11
29

Universidad Nacional Autónoma de México

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGON

DISEÑO DE UN SISTEMA PARA LA ADMINISTRACION DE LOS HORARIOS DE LA CARRERA DE INGENIERIA MECANICA ELECTRICA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A

YOLANDA CUEVAS SALGADO

DIRIGIDA POR ING. ROBERTO BLANCO BAUTISTA

México

 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A mis padres:

Amparo Salgado S.

Luis Cuevas C.

y

hermanos:

Patricia Cuevas S.

Alejandro Cuevas S.

AGRADECIMIENTOS

Al Ing. Roberto Blanco Bautista, quien amablemente aceptó dirigir este trabajo, y por sus valiosos comentarios a lo largo de todo el desarrollo.

A los señores Profesores miembros del jurado, por sus aportaciones para la versión final de la tesis:

Ing. Martín Contreras Soto
Ing. Juan Gastaldi Pérez
Ing. José González Bedolla
Ing. Martín Ordóñez Rosales

Al Físico y Maestro en Ingeniería Alejandro Cuevas S., por sus sugerencias en el diseño y codificación del programa.

CONTENIDO

CAPITULO 2

CAPITULO 3

ALGORITMOS

FIGURAS

INTRODUCCION

La administración de los horarios en la carrera de Ingeniería Mecánica - Eléctrica (ENEP-ARAGON) se lleva a cabo en una computadora de la Corporación Burroughs modelo 600. La computadora se encuentra fuera de la institución y no se cuenta con medios de enlace directo hacia el equipo de cómputo, lo cual ocasiona que el proceso que se sigue para la actualización de la información sea lento y no se tenga disponibilidad inmediata. Bajo este esquema, para el departamento de Ingeniería Mecánica-Eléctrica implica una considerable pérdida de tiempo la actualización, consulta e impresión de los horarios.

Por otro lado, la administración de los horarios es compleja debido al gran número de alumnos, los cuales conforman aproximadamente 340 grupos materia, y de ahí la importancia de contar con un sistema de software que permita administrar la información para la toma de decisiones y para facilitar las actividades académico - administrativo.

Este trabajo de tesis tiene por objetivo encontrar una solución del problema de horarios mencionado, mediante la creación de un Sistema de Software desarrollado exclusiva y específicamente para tal fin. Este Sistema se ajusta a todas las necesidades que involucra el problema de horarios, aprovechando al máximo los recursos de hardware disponibles (tiempo y espacio).

El Sistema ofrece al usuario la posibilidad de actualizar la información en el momento que se necesite, permite consultar datos propios de materias, grupos, profesores, salones, exámenes finales (primera y segunda vuelta) y exámenes extraordinarios (primera y segunda vuelta). La información consultada puede enviarse a impresión; los listados siguen el formato ya establecido por la Coordinación de Ingeniería.

El Sistema es muy amigable al usuario, ya que cuenta con un módulo de ayuda y mensajes de error en todas las opciones del mismo. De esta manera se evita el uso de manuales adicionales; además de que el usuario tiene acceso a toda la ayuda disponible en el mismo Sistema, logrando rapidez y eficiencia.

El Sistema permite transferir, en archivos de texto (ASCII), los datos correspondientes a los grupos de materias impartidas por la Jefatura de Ingeniería Mecánica-Eléctrica. Y siguiendo el protocolo de comunicación respectivo, ésta información es procesada posteriormente en el Sistema Central de Administración Escolar de la UNAM.

El trabajo de tesis se divide en tres capítulos.

El problema de horarios.

En el capítulo 1 se describe ampliamente el problema de la administración de los horarios en la Jefatura de la carrera de Ingeniería Mecánica - Eléctrica, y se analiza la información de los horarios, proporcionada por la Jefatura de carrera. Se consideran los recursos disponibles y se obtiene un planteamiento general.

Solución.

En el capítulo 2 se hace mención de las estructuras de datos como parte fundamental en el desarrollo del Sistema, se describe el diseño completo del sistema de horarios y se plantean los algoritmos de las principales funciones que dan solución al problema de horarios. Se dan algunas características del lenguaje C y las razones de su elección.

El Programa.

En el capítulo 3 se presenta la estructura del Sistema; el uso de los dispositivos de entrada y salida (teclado y video) para la comunicación con el usuario. Además, se listan las principales funciones de biblioteca utilizadas en la codificación. Se describen los tipos de manipulación del video, el modo texto y el modo gráfico, y se justifica la elección del modo gráfico como la forma de comunicación con el usuario.

También se describen los modelos de memoria y se explica cual de ellos se seleccionó. Se describen brevemente los modos de operación del microprocesador con que se trabajó (80386). Por último, se explica la elaboración de un algoritmo de comunicación con la memoria extendida, para poder direccionar toda la memoria física del computador.

Se finaliza con las conclusiones, que describen los resultados obtenidos, así como los principales beneficios obtenidos en la realización del Sistema. Se deja al lector la opción de consultar las referencias del texto en los apéndices.

El apéndice A resume la arquitectura del microprocesador 80386. Muestra todos los registros del 80386 en diagramas de bloques.

El apéndice B contiene la documentación del listado del programa completo. Se describe la estructura general del programa, la forma de compilación y recomendaciones para su correcta ejecución.

CAPITULO 1

EL PROBLEMA DE HORARIOS

1.1. EL PROBLEMA DE LA ADMINISTRACION DE HORARIOS

Desde hace aproximadamente doce años, en la administración escolar se decidió realizar un proyecto que diera solución al problema de la administración de los horarios. Colaboraron diez personas en el proyecto, dos años más tarde el sistema estaba terminado. éste sistema se codificó en ALGOL y se instaló en una red de cómputo utilizando un equipo Burroughs modelo 600, el cual guarda la información en unidades de cinta.

El proceso que se sigue desde entonces para actualizar la información es el siguiente:

Al iniciar un período de clase se realizan modificaciones en la información, éstas modificaciones consisten en agregar, eliminar o cambiar grupos, profesores y salones. Se maneja un listado de cada uno, el cual es marcado manualmente con las modificaciones que se requieran. Una vez que se tienen marcados los listados con todas las modificaciones, son entregados a una persona quien los lleva a un lugar fuera de la institución, en donde se encuentra el sistema que ha de ejecutar la nueva impresión.

Para actualizar el archivo que contiene la información de los listados, el tiempo varía dependiendo de la cantidad de correcciones, si son muchas es de aproximadamente 6 a 8 horas. Por consecuencia, las nuevas impresiones son entregadas a la institución una semana después de que salieron de ella.

Para la reinscripción se requieren cinco listados por cada carrera (listados por grupo, profesor, salón, exámenes finales y/o extraordinarios), en los dos primeros meses del período de clase surgen cambios que son marcados manualmente en los listados; para transferir la información de una Jefatura de carrera a otra, la actualización consiste en prestarse los listados y marcar manualmente las modificaciones. Una vez que los listados estan completos se vuelve al proceso de envío, para obtener los nuevos listados con todas las modificaciones. Si existiera un cambio posterior, vuelven a marcarse los listados.

El procedimiento sigue durante todo el período de clase y continua en los exámenes finales y extraordinarios.

1.2. ANALISIS DE REQUERIMIENTOS

1.2.1. NECESIDADES

Debido a la complejidad del proceso y el tiempo invertido en su realización, se decidió crear un nuevo Sistema, el cual pudiera tenerse dentro de la institución, se accesara a la información en un mínimo de tiempo, de manera eficiente, y se obtuvieran los listados actualizados que se necesitacen en el momento requerido.

Se utilizan por semestre listados que contienen información de los grupos de cada materia. Estos listados son clasificados de la manera siguiente:

- Listado por Grupo
- Listado por Profesor
- Listado por Salón
- Listado de Exámenes Finales
- Listado de Exámenes Extraordinarios

Listado por Grupo, contiene los siguientes datos: número de grupo, nombre y clave de la materia a la que está asignado el grupo, nombre y RFC del profesor que imparte la materia en dicho grupo, número de salón, días y horario en que se imparte la clase y el cupo máximo de alumnos que pueden tomar clase en el salón asignado.

Listado por Profesor, contienen los siguientes datos: nombre y RFC del profesor, nombre y clave de la materia o materias que imparte, los grupos en donde imparte la o las materias, los salones, días y horarios en que imparte clase.

Listado por Salón, contiene los siguientes datos: número de salón, nombre y clave de las materias que se imparten en dicho salón, así como los días y horarios, nombre de los profesores que imparten las materias, y el cupo clasificándolos en cuatro categorías, salones chicos si su cupo máximo es de 30 alumnos, salones grandes si su cupo máximo es de 60 alumnos, salones de dibujo y salones de laboratorio.

Listado de Exámenes Finales, contiene los siguientes datos: nombre y clave de la materia, número de grupo correspondiente, nombre y RFC del profesor que realizará el exámen, salón, día, hora, fecha del exámen final de primera vuelta y fecha del exámen final de segunda vuelta, así como el número de semestre.

Listado de Exámenes Extraordinarios, contiene los siguientes datos: nombre y clave de la materia, número de grupo correspondiente, nombre y RFC del profesor titular que realizará el exámen y el nombre y RFC del profesor suplente, salón, día, hora y fecha del exámen, así como el número de semestre.

Los listados por grupo, de exámenes finales y exámenes extraordinarios deben de tener la opción de que sólo si es requerido el RFC del profesor se imprima, de lo contrario que se omita.

Además de éstos cinco listados se requiere hacer consultas de los datos por materia, grupo (de teoría y laboratorio), profesor, salones y exámenes (final y extraordinario). A excepción de consulta por materia las demás consultas serán similares a los listados.

La consulta por materia contiene los siguientes datos: nombre y clave, si es materia obligatoria u optativa, tipo de laboratorio (opcional, obligatorio o no tiene laboratorio), los grupos en que se imparte y que profesores imparten clase en los grupos asignados a la materia que se esté consultando. También se requiere un listado de los datos de materia.

Se requiere que el Sistema sea capaz de realizar actualizaciones en la información. Las actualizaciones en cuanto a las materias incluyen agregar, eliminar o cambiar los datos de alguna materia como son: nombre, clave, tipo de materia (obligatoria u optativa) y tipo de laboratorio (obligatorio, opcional o sin laboratorio).

Las actualizaciones en cuanto a datos de grupo, profesor, salón y exámenes son similares.

El Sistema debe de cumplir con cierta longitud en los campos de nombre del profesor, RFC del profesor, clave de materia y número de grupo. Esto es con el

propósito de que en el momento que se requiera pueda ser transportada la información en código ASCII, para poder ser procesada en algún otro sistema.

El Sistema procesará datos de la carrera de Ingeniería Mecánica-Eléctrica, cubriendo las tres áreas de dicha carrera, el área mecánica, el área eléctrica y el área de electrónica.

Debido a que el Sistema esta orientado a todos los usuarios que requieran saber éste tipo de información, es imprescindible que el Sistema pueda ser manejado con la mayor sencillez y con el mínimo de conocimientos de software. Se requiere que sea rápido y eficiente en su ejecución.

1.2.2. RECURSOS

Una vez terminado el Sistema se pretende que se instale en una computadora con microprocesador de la serie 80386 ú 80486 con un mínimo de 2 ó 4 Mb de memoria RAM, con monitor VGA color. El Sistema tendrá que ser capaz de ejecutarse en una máquina con las características señaladas. Las impresiones se llevarán a cabo en una impresora estándar en hoja tamaño doble carta.

1.3. DISEÑO DEL SISTEMA

La información que se tiene actualmente esta disponible en listados por grupo, profesor, salón, exámenes finales y exámenes extraordinarios.

Las siguientes tablas muestran los atributos de grupo, profesor, salón, exámenes finales y extraordinarios.

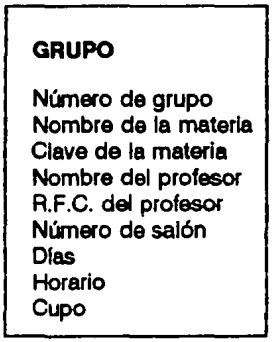


Fig. 1.1. Atributos del listado por Grupo

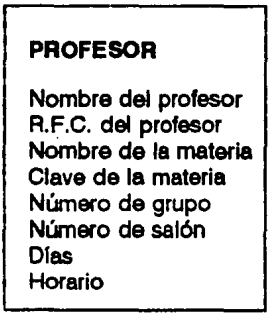


Fig.1.2. Atributos del listado por Profesor

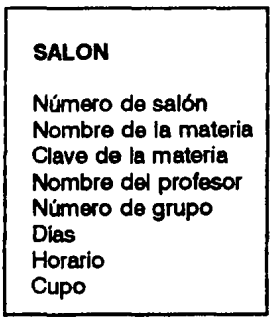


FIG. 1.3. Atributos del listado por por Salón.

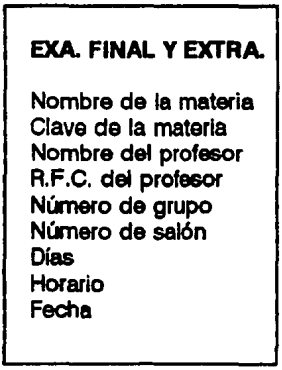


FIG. 1.4. Atributos de los listados de Exámenes Final y Extra.

Los atributos de cada tabla son semejantes. La relación que existe entre ellos es la siguiente, cada materia tiene un número determinado de grupos y cada grupo tiene un profesor asignado y un salón en donde se imparte la clase. Generalmente se abren ocho grupos por materia en un semestre, se abren cuatro grupos para el turno matutino y cuatro para el turno vespertino, con la siguiente identificación:

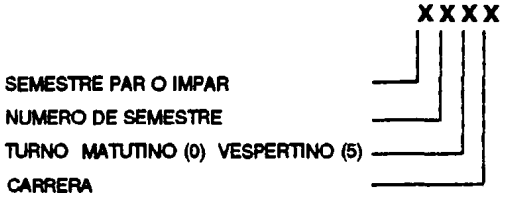


Fig. 1.5. Significado de los cuatro dígitos que componen un grupo.

Hay algunos períodos de clases en los que es necesario abrir más de ocho grupos de alguna materia.

A cada grupo se le asigna un profesor, un salón y un horario. A cada profesor se le asignan una o varias materias con uno o varios grupos por materia y uno o varios salones con sus respectivos horarios.

A diferencia de la tabla de exámen final y extraordinario, las tablas de grupo, profesor y salón tienen atributos que están interrelacionados entre sí, los cuales se refieren a información del período de clases. En la tabla de exámen final y extraordinario, sus atributos se refieren a información del período de exámenes ordinario y extraordinario.

La tabla de exámenes final y extraordinario tienen un atributo más, que es la fecha de exámen. En el caso del exámen final, se requiere de fecha de primera vuelta y fecha de segunda vuelta. En el caso de exámen extraordinario se requiere

de dos datos de profesor, el profesor titular y el profesor suplente,los demás atributos son similares para ambos tipos de exámen.

Debido a lo anterior los datos se jerarquizaron en el siguiente orden:

1. Materias
2. Grupos
    2.1. Profesores
    2.2. Salones
3. Exámenes
    3.1. Profesores
    3.2. Salones

El diagrama de bloques de la fig. 1.6, muestra una forma general de la jerarquización.

El diagrama visualiza el esquema general del problema de horarios, se muestra una lista de materias hasta n materias, se toma como ejemplo MATERIA 1, a la cual se le asignaron ocho grupos, con la variante de que puede asignarse hasta n grupos ó menos, a cada grupo se le asigna un profesor y a cada profesor un salón. Además de asignar grupos a cada materia, también se le asigna exámenes final y/o extraordinario. A cada materia se le asignan un número n de exámenes finales. Pero sólo se asigna un exámen extraordinario por materia. A cada tipo de exámen se le asigna un profesor y un salón.

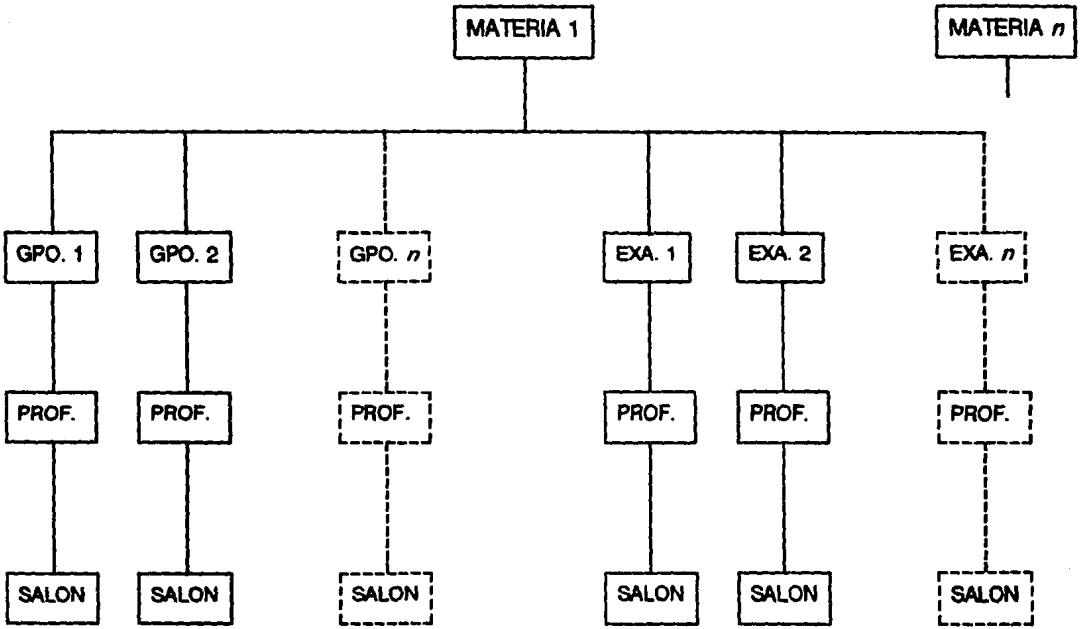


Fig. 1.6. Esquema general de la jerarquización de los atributos de los listados.

CAPITULO 2

SOLUCION

Para poder llevar el planteamiento de un problema a una forma para su realización en un programa, se requiere saber los elementos de programación que pueden utilizarse, éstos elementos son los algoritmos empleados para resolver el problema de programación, así como las estructuras de datos para modelar la información procesada por los programas.

2.1. BASE DE DATOS

Una *base de datos* es una colección de datos interrelacionados. Concretamente, la base de datos es la colección completa de datos, apuntadores, tablas, índices, etc.

Uno de los objetivos primordiales de la base de datos es crear un ambiente para la recuperación de información y para almacenar información nueva en la base de datos.

Un sistema de base de datos se divide en módulos que se encargan de cada una de las tareas del sistema general. Algunas de las funciones del sistema de base de datos pueden ser realizadas por el sistema operativo. En la mayor parte de los casos, el sistema operativo proporciona únicamente los servicios más elementales y la base de datos debe partir de ese fundamento. Así, el diseño de la base datos debe incluir una consideración de la comunicación entre el sistema de base de datos y el sistema operativo.

Un sistema de manejo de base de datos se compone de una serie de datos relacionados entre sí y de un conjunto de programas para tener acceso a esos datos.

Los sistemas de base de datos se diseñan para manejar grandes cantidades de información. El manejo de los datos implica tanto la definición de estructuras para el almacenamiento como la creación de mecanismos para manejar la información. Algunas de las estructuras de datos y mecanismos empleados para manipular la información serán tratados en las siguientes secciones.

2.2. ESTRUCTURAS DE DATOS

Una *estructura de datos* es un modelo lógico de una organización particular de datos, la cual muestra la relación entre los datos procesados y lo que representan.

Para seleccionar la estructura de datos que convenga, se considerá primero que sea capaz de mostrar la relación entre los datos y lo que representan; segundo, deberá de ser una estructura simple tal que los datos puedan ser procesados de manera eficiente cuando se necesite. Un modelo de estructuras de datos nos permite construir funciones que almacenan y accesan a elementos individuales de datos.

En la construcción de cualquier estructura de datos, se reserva espacio en memoria para la estructura de datos y se codifican las funciones de acceso.

A continuación se hace mención de algunas de las estructuras de datos más usuales.

2.2.1. ARREGLOS

Es una lista de un número finito de datos del mismo tipo. El acceso se realiza mediante un índice que especifica la posición del elemento en el arreglo.

La siguiente notación muestra un arreglo designado por la letra A.

A[1], A[2], A[3], ... A[i], ..., A[N]

i: es el índice

A[i]: es la variable subindicada

Hay dos tipos de arreglos:

Los arreglos unidimensionales (ó lineales), a cada elemento del mismo se referencia a través de un sólo índice. Los arreglos bidimensiones, son una colección de datos del mismo tipo donde cada elemento se referencia por dos índices (dos dimensiones), el primer índice referencia a la primera dimensión, el segundo se refiere a la posición del elemento en la segunda dimensión, por ejemplo, en la representación de una matriz:

A[i][j]

/ \

Especifica renglón Epecifica columna

2.2.2. REGISTRO

El registro es una estructura de datos que manejan algunos lenguajes de programación.

Un registro es una colección finita de elementos no necesariamente del mismo tipo, en la cual cada elemento se identifica por medio de su propio campo identificador.

Ejemplo:

2.2.3. PILA

Es una estructura LIFO (primero en entrar, último en salir). Es una colección ordenada de elementos en la cual las inserciones y extracciones tienen lugar en un sólo extremo llamado la parte superior de la pila.

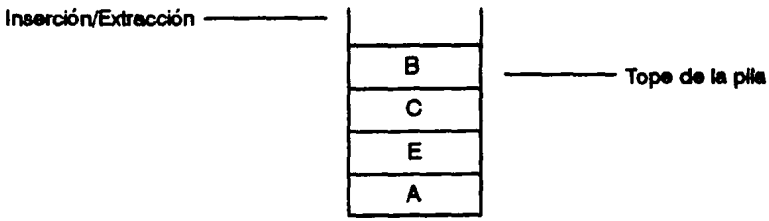


Fig. 2.1. Representación de la Pila

2.2.4. COLA

Es una estructura FIFO (primero en entrar, primero en salir). Es una colección ordenada de elementos, en la cual se pueden eliminar elementos de un extremo llamado el frente de la cola, y también se pueden insertar elementos en el otro extremo llamado el final de la cola.

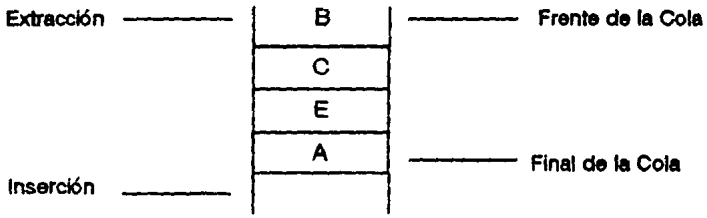


Fig. 2.2. Representación de la Cola

2.2.5. LISTA ENLAZADA

Es una colección de elementos, donde a cada elemento se le denomina *nodo*, el orden se establece mediante apuntadores. Cada nodo consta de dos partes, la parte información, la cual contiene información asociada al elemento, es decir, contiene el elemento actual de la lista; la parte enlace ó campo apuntador al siguiente, que contiene la dirección del siguiente nodo de la lista.

La lista enlazada contiene un apuntador externo al primer nodo de la lista para poder acceder a toda la lista. El apuntador externo quiere decir que no está incluído

dentro del nodo. En su lugar se accesa haciendo referencia a una variable. No se puede accesar aleatoriamente a un nodo particular de la lista enlazada. Se tiene que recorrer la lista desde su inicio hasta llegar al nodo requerido.

El campo apuntador del último nodo, contiene un valor especial denominado nulo, que es una dirección no válida. El apuntador nulo señala el final de la lista. La lista sin nodos se llama lista vacía o nula.

Ejemplo:

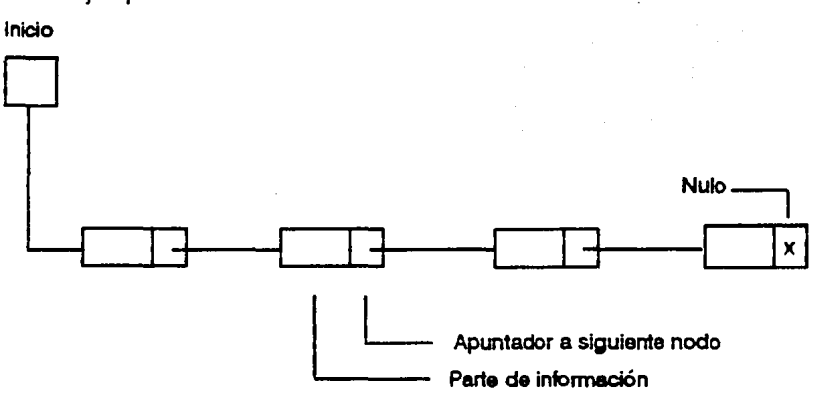


Fig. 2.3. Lista enlazada

2.2.6. ARBOL

Es una estructura de datos que se usa para representar datos con una relación jerárquica entre sus elementos.

Cada árbol tiene un primer elemento llamado *raíz* del árbol, y los elementos restantes son partidos en subconjuntos separados mayores o iguales a cero, cada

uno de los cuales es en sí un árbol. A cada elemento de un árbol se le llama un nodo del árbol. Cada nodo puede ser la raíz de un árbol con un número de subárboles mayor ó igual a cero. Un nodo que no tiene subárboles es una hoja.

El nivel o grado de un nodo se refiere a su distancia a la raíz.

Ejemplo:

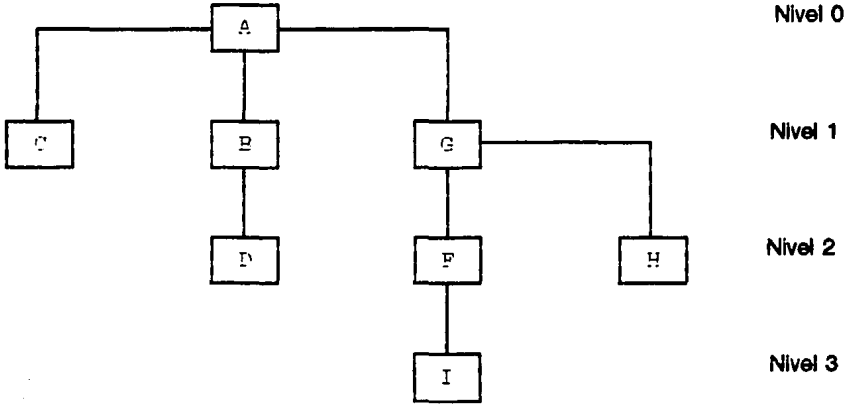


Fig. 2.4. Arbol

La profundidad (o altura) de un árbol es el número máximo de nodos de una rama. Equivale a 1 más que el mayor número de nivel de la rama. La estructura del ejemplo anterior tiene nivel máximo 3 y profundidad de 4.

2.3. EL PROBLEMA DE HORARIOS VISTO A TRAVES DE ESTRUCTURAS DE DATOS

El capítulo 1 se concluyó con el planteamiento general del problema de los horarios en Ingeniería Mecánica-Eléctrica. El siguiente paso es la formulación de los algoritmos que darán solución al problema, así como la elección de las estructuras de datos para su construcción.

Tomando el diagrama de bloques de la fig. 1.6., es fácil identificar a **MATERIA** como el elemento principal de la estructura, ya que de éste se desprende **GRUPO** y **EXAMEN**, y de cada uno se desprende profesores y salones.

Muchos de los programas que dan solución a un problema son creados bajo las condiciones propias en ese instante, y algunas veces no se preveen los futuros cambios que se pueden requerir en un corto período de tiempo. Debido a ello es necesario preveer ciertos parámetros para que el programa no sea obsoleto. Teniendo la posibilidad de ser modificado bajo las condiciones que se requiera y que sea posible.

Uno de los parámetros que es necesario preever es el incremento de estudiantes, lo cual origina un aumento en el número de grupos, por este motivo es conveniente utilizar una estructura dinámica que permita incrementar el número de grupos hasta donde los recursos del computador lo permitan.

Dentro de **GRUPO** es en donde se realizan una mayor cantidad de modificaciones tales como agregación, eliminación o modificación de grupos, y de los datos de grupo (datos de profesor y salón), debido a ello se requiere utilizar una estructura que nos permita accesar a algún elemento por medio de búsquedas, y se pueda eliminar, agregar o modificar algún grupo. Una estructura que nos permite éstas modificaciones y que además es una estructura dinámica, es la lista enlazada.

De ésta manera se seleccionó a la *lista enlazada* como la estructura de datos propia para **GRUPO**, dentro de la cual se definieron los datos de profesor y salón.

En **EXAMEN**, al igual que **GRUPO**, también se realizan las operaciones de agregar, eliminar o modificar datos de exámen. Como el número de exámenes finales varía según el número de grupos creados, la estructura para exámenes es similar a la de grupos. De igual manera se seleccionó a la *lista enlazada* como la estructura de datos adecuada para **EXAMEN**.

Algunas de las modificaciones que pueden ocurrir no precisamente en un corto período de tiempo, es en **MATERIA** en donde también es conveniente utilizar una *lista enlazada*, por las mismas razones que en **GRUPO**. Además, ya que de ésta manera, el sistema puede ser utilizado en otra Jefatura de carrera que tenga necesidades similares, realizando un mínimo de cambios y sin tener la limitante de un cierto número de materias.

La forma de unir la lista enlazada de **GRUPO** y de **EXAMEN** a la lista enlazada de **MATERIA** es la siguiente:

Debido a que **MATERIA** es el bloque base, como ya se había mencionado en el planteamiento del problema, es la lista enlazada de la cual de cada nodo se desprenderá una lista enlazada de **GRUPO** y también una lista enlazada de **EXAMEN**.

Cada nodo de la lista enlazada de **MATERIA** contendrá en su parte información los datos de materia, una estructura de grupos y una estructura de exámenes, éstas estructuras de grupos y exámenes formarán la lista enlazada **GRUPO** y la lista enlazada **EXAMEN**, en la parte información de cada una contendrá los datos de grupo ó de exámen, profesor y salón. En el diagrama de la fig. 2.5., se ejemplifica la forma en como se manejan las lista enlazadas.

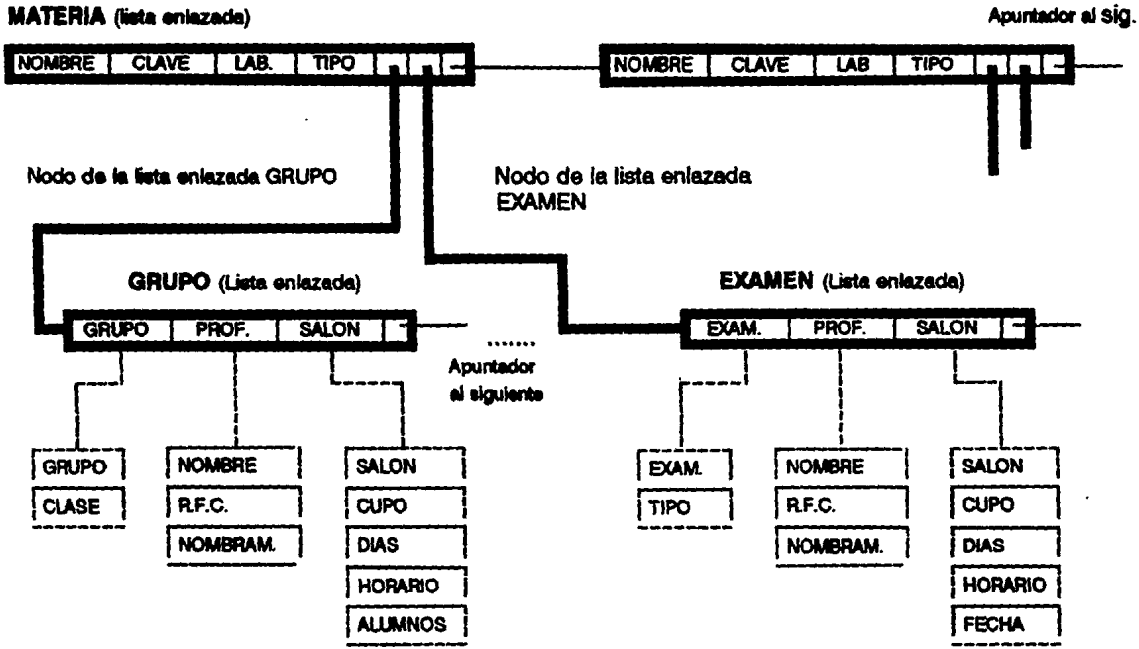


Fig.2.5. Diagrama del planteamiento general como una Estructura de datos.

2.3.1. BUSQUEDA

Además de las operaciones de actualización (agregar, eliminar ó cambiar) se necesitan operaciones de búsqueda y ordenación.

Las búsquedas se realizan recorriendo la lista enlazada en forma lineal, es decir, si queremos localizar **DATO** en la lista **MATERIA**, se compara **DATO** con el contenido de la parte información **INF** de cada nodo de la lista.

El siguiente algoritmo encuentra la posición **POS** del nodo del elemento **DATO** en la lista **MATERIA**, almacenada en memoria, de lo contrario devuelve **POS = NULO**. La lista se recorre utilizando el apuntador **PUNTR** y comparando **DATO** con el contenido **INFO[PUNTR]** de cada nodo.

Busca **DATO** en la lista **MATERIA**

1. **PUNTR** = **INICIO**.
2. Mientras **PUNTR** != **NULO** entonces:
   Si **DATO** = **INFO[PUNTR]**, entonces:
   POS = **PUNTR**; salir.
   Si no:
   PUNTR = **SIGUIENTE[PUNTR]**.
   (Fin del ciclo del paso 2);
3. No se encontró el elemento.
   POS = **NULO**.
4. FIN.

Algoritmo 2.1. Búsqueda Lineal sobre la lista enlazada MATERIA.

Para obtener la posición de un elemento en la lista **GRUPO**, se recorren los elementos uno a uno de la lista **GRUPO** del primer nodo de la lista **MATERIA**, si no

se encontró, se recorren los elementos uno a uno de la lista **GRUPO** del segundo nodo de la lista **MATERIA**, así sucesivamente hasta que sea encontrado el elemento buscado.

El siguiente algoritmo encuentra la posición **POS** del nodo del elemento **DATO** en la lista **GRUPO**, almacenada en memoria, de lo contrario devuelve **POS = NULO**. La lista **MATERIA** se recorre utilizando el apuntador **PUNTR**, y la lista de **GRUPO** se recorre utilizando el apuntador **PUNTERO**, se compara **DATO** con el contenido **INFO[PUNTERO]** de cada nodo de la lista **GRUPO**. Como de cada nodo de la lista **MATERIA** contiene en su parte información una estructura para grupos estará dada por **GPO = INF[PUNTERO]**.

```
Busca DATO en la lista GRUPO

1. PUNTR = INICIO.
2. Mientras PUNTR != NULO, entonces:
       GPO = INF[PUNTR] (Inicio de la lista GRUPO)
3. Mientras GPO != NULO, entonces:
       Si DATO = INFO[PUNTR], entonces:
             POS = PUNTR; salir.
       Si no:
             PUNTERO = SIGUIENTE[PUNTERO].
       (Fin del ciclo del paso 3);
4. PUNTR = SIGUIENTE[PUNTR]
   (Final del ciclo del paso 2);
5. No se encontró el elemento.
   POS = NULO.
6. FIN.
```

Algoritmo 2.2. Búsqueda lineal sobre la lista enlazada GRUPO.

De igual manera se obtendrá la posición de un elemento en la lista de **EXAMEN**. El algoritmo es el siguiente:

```
Busca DATO en la lista EXAMEN

1. PUNTR = INICIO.
2. Mientras PUNTR != NULO, entonces:
       EXAM = INF[PUNTR] (inicio de la lista GRUPO)
3. Mientras EXAM != NULO, entonces:
       Si DATO = INFO[PUNTR], entonces:
             POS = PUNTR; salir.
       Si no:
             PUNTERO = SIGUIENTE[PUNTERO].
       (Fin del ciclo del paso 3);
4. PUNTR = SIGUIENTE[PUNTR]
   (Final del ciclo del paso 2);
5. No se encontró el elemento.
   POS = NULO.
6. FIN.
```

Algoritmo 2.3. Búsqueda lineal sobre la lista enlazada EXAMEN.

2.3.2. ORDENACION

Se seleccionó el método de ordenamiento rápido (quick sort) para el algoritmo de ordenación.

El método de ordenamiento rápido consiste en reorganizar los elementos numéricos ó alfabéticos de una lista de n elementos en un orden lógico, a partir de un conjunto de n elementos se divide para ordenar conjuntos menores.

Se tiene el siguiente arreglo a de n elementos, donde cada a[j] es cualquier número, y x es la posición buscada.

a[1], a[2], a[3], a[4], a[5], .., a[j-1], a[j], a[j+1], .., a[n]

El paso de reducción es buscar la posición final de uno de los elementos de la lista. En este caso se toma el primer elemento a[1], se compara con el último elemento a[n], recorriendo la lista de derecha a izquierda, comparando cada elemento con a[1] y parando en el primer número menor que a[1], se intercambian, colocando a[1] en la posición del elemento menor y colocando a[menor] en la posición a[1].

a[menor], a[2], a[3], a[4], a[5], a[1], ... ,a[n]

Comenzando con a[menor] se sigue recorriendo la lista en sentido contrario, de izquierda a derecha, comparando cada número con a[1] y parando en el primer número mayor que a[1]. Se intercambian a[1] con el elemento mayor a[mayor].

a[menor], a[2], a[3], a[1], a[5], a[mayor], ... , a[n]

Cuando todos los números a la izquierda de a[1] sean menores y a la derecha sean mayores se habrá encontrado la posición del elemento a[1]. Ahora todos los números menores que a[1] forman la sublista de números a la izquierda de a[1], y todos los números mayores de a[1] forman ahora la sublista de números a la derecha de a[1]. Donde a[1] es igual a[j], que es la posición del elemento x.

Lo anterior debe cumplir con las siguientes dos condiciones:

1. Los elementos en posiciones 1 hasta j-1 son menores que o iguales a x.
2. Los elementos en posiciones j+1 hasta n son mayores o iguales a x.

a[1], a[2], a[3], a[4], ..., a[j-1], a[j], a[j+1], ..., a[n]

Repitiendo el proceso con los subarreglos a[1], hasta a[j-1] y a[j+1] hasta a[n], se obtendrá la lista ordenada.

Para construir el método se utilizan dos pilas, una para la lista de menores y otra para la lista de mayores. En cada pila se meten los elementos límite, es decir el elemento primero y último de cada subarreglo.

A continuación se muestra el algoritmo de ordenamiento rápido el cual se divide en dos partes, el procedimiento **PARTICION**, ejecuta el paso de reducción del algoritmo, y la segunda parte **ORDENACION_RAPIDA** ordena la lista completa de elementos. Los procedimientos **QUITA_PILA** y **PON_PILA** realizan las operaciones de quitar y meter a la pila los elementos de los límites superior e inferior de los subarreglos que deben ser ordenados.

```
PARTICION(A,INF,SUP,POSICION)

1. IZQ = INF
   DER = SUP
   POS = INF
2. Mientras IZQ < DER entonces:
3. Mientras  A[POS].DATO <= A[POS].DATO y POS != DER
        Si POS = DER entonces: salir
        Si A[POS].DATO > A[DER].DATO entonces:
              INF = A[POS].INF
              A[POS].INF = A[DER].INF
              POS = DER
4. Mientras A[IZQ].DATO <= A[POS].DATO Y IZQ != POS
        IZQ = IZQ + 1
5. Si POS = IZQ entonces: salir
6. Mientras A[IZQ].DATO > A[POS].DATO entonces:
        INF = A[POS].INF
        A[POS].INF = A[IZQ].INF
        POS = IZQ
   (Fin del condicional del paso 2)
7. POSICION = POS.
8. Fin.
```

```
ORDENACION_RAPIDA(ARREGLO,N)

1. TOPE = NULO.
2. Si N > 1 entonces:
        ELEM.INF = 0
        ELEM.SUP = N
        PON_PILA(PILA,ELEM,N) (Pone la estructura ELEM en pila)
        3. Mientras TOPE  != 0 entonces:
                QUITA_PILA(PILA,ELEM) (Quita ELEM de pila)
                PARTICIONA(ARREGLO,ELEM.INF,ELEM.SUP,POS)
                4. Si (ELEM.INF < POS - 1)  entonces:
                        TEMP = ELEM.SUP
                        ELEM.SUP = POS - 1
                        PON_PILA(PILA,ELEM,N)
                        ELEM.SUP = TEMP
                5. Si (POS + 1 < ELEM.SUP) entonces:
                                ELEM.INFERIOR = POS + 1
                                PON_PILA(PILA,ELEM,N)
          (Fin del condicional del paso 3)
(Fin del paso 2)
6. Fin.
```

Algoritmo 2.4. Método de ordenamiento rápido (Quick Sort).

2.3.3. COMUNICACION CON EL USARIO (ENTRADA Y SALIDA DE DATOS)

La comunicación entre el usuario y el Sistema se realiza por medio de dispositivos periféricos. Algunos de ellos son lectoras ópticas, sensores de marcas que pueden leer tarjetas marcadas con un lápiz especial, teclado, ratón, lectoras de cinta magnética, registradores gráficos, monitor, etc.

En éste Sistema el usuario podrá enviar datos al programa por medio del teclado y los recibirá por medio del monitor e impresora.

Para facilitar el envío de mensajes al usuario se emplean las ventanas, que son la parte activa de la pantalla dentro de la cual se muestra la salida.

Al accesar a algún módulo de un menú de opciones dentro de la ejecución de un programa, aparecerán las ventanas de mensaje, y desaparecerán al salir, a partir de la última ventana colocada hasta la primera que apareció.

Debido a que las ventanas aparecen en un orden, y desaparecen a partir de la última ventana que se haya colocado hasta la primera, la estructura de datos que se requiere para ésta aplicación es la *pila*, ya que es una estructura dinámica en donde se agregan hasta n número de ventanas y la última ventana que se coloque será la primera en quitarse.

El utilizar la pila para almacenar las ventanas permite que la ejecución del programa sea eficiente. Esta estructura almacena una gran cantidad de ventanas, por lo que requiere de mucho espacio en memoria, debido a ello es necesario plantear un mecanismo que permita enviar los apuntadores de la pila de ventanas a la memoria extendida. Al utilizar toda la memoria física del computador permite tener más espacio para los datos. El controlador de memoria extendida se explicará ampliamente en el siguiente capítulo.

2.4. SELECCION DEL LENGUAJE DE PROGRAMACION

Para la codificación del programa hay que considerar las estructuras de datos que se utilizarán en su diseño, seleccionando un lenguaje estructurado que soporte las estructuras utilizadas.

Las estructuras de datos que se requieren en el programa son: arreglos, registros, pilas y listas enlazadas.

Se requiere un lenguaje estructurado de programación que soporte las estructuras de datos estáticas y dinámicas, que permita además la compilación de programas de archivos múltiples.

Cuando en el programa se manejan estructuras de datos dinámicas es imposible saber el espacio que necesitarán en su ejecución, en este caso se trata de un programa de base de datos que utiliza estructuras dinámicas, en el que no se puede predefinir el espacio que se necesitará para su ejecución, es necesario que el lenguaje en que se codifique el programa permita asignar y liberar memoria dinámicamente.

Un lenguaje que soporta las características anteriores es el lenguaje C.

Se seleccionó el lenguaje C debido a que es un lenguaje estructurado, de empleo general, no está especializado en ningún área particular de aplicación, se caracteriza por su portabilidad, además ofrece una velocidad similar a la del ensamblador.

El C permite la compilación separada: cuando se utilizan estructuras dinámicas el C asigna una región de memoria libre que se usa mediante las funciones de asignación dinámica.

Se utilizó el compilador BORLAND C++ versión 3.1. El Borland C++ es un compilador profesional que optimiza las funciones del C++ y C, incluye las últimas caracteríticas de programación que se tienen para:

- C y C++: Borland C++ ofrece la potencia de programación del C y C++, con una herramienta completa del AT&T C++ versión 3.0 y el ANSI C.

- Optimización global: Una serie completa de opciones para optimizar el control completo sobre generación de código.

- Rápida compilación: Borland C++ compila en la mitad de tiempo del C++. Precompila las cabeceras recortando significativamente el tiempo de recompilación.

- Compilación DPMI: La compilación de los programas enormes están limitados sólo por la memoria del sistema. Borland C++ usa la interface DPMI (DOS Protected Mode Interface) para permitir la compilación y ejecución en modo protegido bajo el DOS o en el modo mejorado del 386 para Windows.

- Plataforma de programación:

  - Incluye un editor de archivos múltiples con características tanto en interfaces DPMI y CUA (Common User Access) e interfaces alternas compatibles con versiones previas del Borland C++.

  - Depurador completamente integrado con ejecuciones en DPMI, para depuración de aplicaciones grandes.

  - Soporta código ensamblador en línea.

  - Múltiple solapamiento de ventanas con soporte completo del ratón.

  - Intensidad del color en la sintaxis para ayudar a distinguir visualmente los elementos del código fuente.

CAPITULO 3

EL PROGRAMA

Una vez que se tiene bien definida la estructura general del problema, el planteamiento de los principales algoritmos y se ha seleccionado el lenguaje de programación, el siguiente paso es la creación de los bloques de código que constituyen el programa completo.

La codificación completa del programa se encuentra en el apéndice B.

3.1. MODELOS DE MEMORIA

Para la familia de procesadores 8086, 80286, 80386 y 80486, Borland C++ puede compilar los programas de seis maneras distintas. Cada una organiza de forma diferente a la memoria de la computadora durante la ejecución del programa.

Para saber que se necesita para seleccionar un modelo de memoria, es necesario conocer el procesador con que se trabaja.

Para el desarrollo del programa de aplicación se trabajó con el microprocesador 80386. En el apéndice A se encuentra la arquitectura del 80386.

Los seis modelos son diminuto, pequeño, mediano, compacto, grande y enorme.

3.1.1. MODELO DIMINUTO (TINY)

Este es el más pequeño de los modelos de memoria. El modelo diminuto compila un programa de tal manera que todos los registros de segmento tienen el mismo valor. Esto significa que el código, los datos y la pila deben estar dentro del mismo segmento de 64 K. Esta manera de compilación produce el código menor y más rápido.

3.1.2. MODELO PEQUEÑO (SMALL)

El segmento de código está separado del segmento de datos, pila y extra, que están en su propio segmento. De esta manera se tiene 64 K para el código y 64 K para datos y pila. La velocidad de ejecución es tan rápida como el diminuto, pero el programa puede ser aproximadamente el doble de grande.

3.1.3. MODELO MEDIANO (MEDIUM)

Los datos y la pila están limitados a 64 K, pero el código puede ocupar arriba de 1 MB. Este modelo es bueno para programas grandes con pocos datos. Los programas irán más lentos en las llamadas a funciones, pero los accesos a datos serán tan rápidos como el modelo pequeño.

3.1.4. MODELO COMPACTO (COMPACT)

Este modelo es el inverso del modelo mediano. El código está limitado a 64 K, mientras que los datos pueden ocupar un rango de 1 MB. Este modelo es apropiado para programas que usan gran cantidad de datos, pero cuyo código es pequeño. Los programas compilados con éste modelo se ejecutarán tan rápido como en el modelo pequeño excepto al hacer referencia a datos.

3.1.5. MODELO GRANDE (LARGE)

El modelo grande permite usar varios segmentos tanto al código como a los datos, es decir pueden tener un rango de 1 MB. Sin embargo, el tamaño máximo de una estructura de datos está limitado a 64 K. Este modelo se usa cuando se tiene una gran cantidad de código y datos. Se ejecuta mucho más lento que los modelos anteriores.

3.1.6. MODELO ENORME (HUGE)

Este modelo permite tanto al código como a los datos ocupar un rango de 1 MB. Es similar al modelo grande, pero con la diferencia de que las estructuras de datos pueden superar los 64 K. Esto hace que la velocidad de ejecución sea aún más lenta que en el modelo grande.

Por defecto el Borland C++ compila en el modelo pequeño, pero debido a que el programa de aplicación tiene mucho código y muchos datos, es necesario que la compilación se ejecute en el modelo de memoria apropiado.

De los seis modelos de memoria, hay dos que permiten la compilación de código y datos en un rango de 1 MB, éstos son el modelo grande y el modelo enorme. Como las estructuras de datos del programa de aplicación no sobrepasan los 64 K, entonces se seleccionó el modelo de memoria grande, el cual ejecuta la compilación un poco más rápido que el modelo enorme.

3.2. FUNCIONES DE VIDEO

La mayoría de los programas actuales están diseñados para trabajar con funciones de gráficos ó con funciones de texto, por medio de ventanas. La ventana por defecto es la pantalla completa.

Para la manipulación de la pantalla de texto, así como para las funciones gráficas, es fundamental el concepto de ventana, la parte activa de la pantalla dentro de la cual se muestra la salida. Las funciones de texto se refieren a ventanas; el sistema de gráficos se refiere a ventanas gráficas, pero el concepto es el mismo.

Existen varios tipos de adaptadores de video, o tarjetas gráficas disponibles para las computadoras PC. Borland C++ provee controladores para los siguientes adaptadores gráficos:

- Color Graphics Adapter (CGA, adaptador de gráficos en color).
- Multi-Color Graphics Array (MCGA, array de gráficos en multi-color).
- Enhanced Graphics Adapter (EGA, adptador de gráficos mejorado).
- Video Graphics Array (VGA, array de gráficos de vídeo).
- Hercules Graphics Adapter.
- AT&T 400-line Graphics Adapter.
- 3270 PC Graphics Adapter.
- IBM 8514 Graphics Adapter.

La parte más pequeña de la pantalla, que es direccionable por el usuario en modo texto, es un *carácter*. En el modo gráfico se pueden mostrar en pantalla tanto texto como imágenes gráficas. La parte más pequeña de la pantalla, que es direccionable por el usuario, es un *pixel*.

Tanto en el modo texto como en el gráfico, las posiciones individuales se referencian mediante su fila y columna. En el modo gráfico, a la esquina superior izquierda de la pantalla corresponde la posición (0,0). En el modo texto a la esquina superior izquierda corresponde la posición (1,1).

3.2.1. MODO TEXTO

En el modo texto la pantalla está organizada en 80 columnas por 25 renglones, cada posición tiene una longitud de 2 bytes llamado *textcel* (es la abreviación de

textcells). Cada texcel consiste de un código de carácter y un atributo. El código de carácter es un código ASCII que incluye varios carácteres gráficos. El atributo controla el texto y el color de fondo.

Fig. 3.1. Texcel

El Borland C++ proporciona un amplio conjunto de funciones en modo texto, que pueden dividirse en la forma siguiente:

ENTRADA Y SALIDA BASICA

Estas funciones permiten escribir y leer un carácter o una cadena en la ventana.

MANIPULACION DE LA PANTALLA

Estas funciones permiten establecer el modo texto de la pantalla, limpiar una ventana o una línea de texto, borrar una línea o insertar una en blanco, copiar texto de la pantalla a un buffer y de un buffer a la pantalla, colocar el cursor en una posición específica, etc.

CONTROL DE LOS ATRIBUTOS

Con el control de los atributos, es posible cambiar los modos de video, así como controlar el color del texto y del fondo y poner la pantalla en alta o baja intensidad.

ESTADO DE LA PANTALLA

Estas funciones devuelven información sobre la ventana de texto actual y las coordenadas del cursor.

APARIENCIA DEL CURSOR

Las funciones al habilitarse cambian la apariencia del cursor, ya sea una barra larga o una barra corta.

Estas funciones necesitan que se incluya el archivo cabecera *conio.h*.

3.2.2. MODO GRAFICO

En el modo gráfico la pantalla está organizada en una matriz de por ejemplo, 640 columnas por 480 en un monitor con adaptador de video VGA.

A cada posición se le denomina pixel, el cual tiene una posición (x,y) dentro de la matriz que define a la pantalla.

El término pixel se refería originalmente al menor elemento de fósforo en el monitor de video que podía exitar un haz de electrones. Sin embargo el término se ha generalizado para referirse al menor punto direccionable sobre un monitor gráfico.

Borland C++ provee una biblioteca de más de 70 funciones gráficas.

La biblioteca de gráficos soporta numerosos estilos de líneas, rellenos de figuras, y provee varios tipos de letras a las que puede modificarse su tamaño, justificar y orientar en forma horizontal y vertical.

Las funciones de gráficos de Borland C++ se pueden agrupar dentro de las siguientes categorías:

CONTROL DEL MODO DE LA TARJETA DE VIDEO

Es necesario poner el adaptador de video en alguno de sus modos gráficos antes de poder usar cualquiera de las funciones de gráficos.

GRAFICOS BASICOS

Las funciones fundamentales de gráficos son las que dibujan un punto, una línea y un círculo. Otras funciones permiten rellenar figuras cerradas con líneas de diferente grosor, dibujar arcos, elipses, etc.

SALIDA DE TEXTO

Permite escribir el texto con distintas fuentes, tamaños e incluso direcciones.

ESTADO DE LA PANTALLA

Devuelve la posición en (x,y) de la ventana gráfica, obtiene información del tipo actual de texto gráfico.

MANIPULACION DE LA PANTALLA

Tiene funciones de control de la ventana gráfica, como copiar parte de una ventana gráfica a un buffer y de un buffer a una ventana, crear ó limpiar una ventana gráfica, etc.

Para el funcionamiento de las funciones de gráficos es necesario que se incluya el archivo cabecera graphics.h.

En la mayoría de los programas de aplicación el diseño que se presenta al usuario debe de ser lo suficientemente sencillo, para facilitar la entrada y salida de datos al programa. Para ello es necesario utilizar las funciones de pantalla para crear ventanas, por medio de las cuales sea posible enviar mensajes al usuario.

Como ya se ha dicho, existen dos formas de hacer estas ventanas, utilizando funciones de control de pantalla en modo texto ó funciones de control de pantalla en modo gráfico.

Cuando se trabaja con el modo texto, sólo se pueden usar rutinas estándar de escritura en pantalla, es posible modificar la intensidad del texto; en el modo gráfico existen varios tipos de letras a los cuales es posible modificar su tamaño y cambiar la dirección en la que se imprima el texto, ya sea horizontal o vertical. También es posible justificar el texto y otras funciones que hace que sea atractivo al programador el uso de éstas funciones.

Debido a que es posible una mayor manipulación de funciones en el modo gráfico que en el modo texto y por las características ya señaladas, en el diseño del programa se utilizaron las funciones de control de pantalla en modo gráfico.

3.3. ADMINISTRACION DE LA MEMORIA

Es necesario entender el uso y control de la memoria del sistema en el que se trabaja. A continuación se muestran las áreas de memoria dentro de una PC.

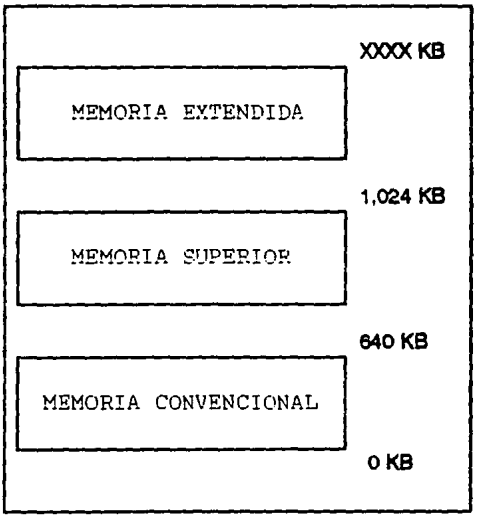


Fig.3.2. Areas de memoria

El área entre 0 y 640 kilobytes es llamada memoria convencional, ya que por el diseño de la PC, es un área reservada para programas en general. El área entre 640 kilobytes y 1,024 kilobytes (1 megabyte) es llamada memoria superior, debido a que reside en el límite superior de las direcciones originales de 8086/8088. El área arriba de 1 megabyte es llamada memoria extendida debido a que es una extensión del espacio de memoria original disponible en una PC.

La memoria convencional y superior pueden ser direccionadas por la PC en modo real. El área extendida puede ser direccionada sólo en modo protegido.

La memoria expandida es un efecto de la necesidad del uso de una PC original para manejar más memoria. Esta emplea un técnica de computadora llamada interruptor de banco, el cual ocupa una porción de un espacio de memoria grande (la parte expandida) en un área de memoria que puede ser facilmente direccionada por la CPU. El interruptor del banco de memoria, está dividido en páginas que pueden ser mapeadas dentro del espacio de memoria convencional.

La siguiente lista resume las áreas de memoria.

- Memoria convencional: de 0 a 640 kilobytes.
- Memoria superior: de 640 kilobytes a 1 megabyte.
- Memoria expandida: memoria adicional intercambiada dentro de la memoria superior.
- Memoria extendia: arriba de 1 megabyte.

3.3.1. MODO REAL Y MODO PROTEGIDO

El 80386 y 80486 operan por defecto en el modo real. Estos CPUs proveen un tercer modo operativo, modo virtual 8086 o V86, el cual permite multiplicar el ambiente del modo real dentro de una máquina simple.

En el modo real, el acceso a un megabyte de RAM requiere una dirección de 20 bits, una dirección está compuesta de un segmento y un desplazamiento. Para calcular una dirección física, se desplaza cuatro bits hacia la izquierda el valor en el registro de segmento y se suma el valor resultante al desplazamiento. Esto construye una dirección de 20 bits. En el apéndice A se describe la arquitectura del 80386.

En el ambiente del modo protegido, los registros de segmento no funcionan como en el modo real. Para evitar confusión, se les llama *registros selectores.* Así como en el modo real, ellos también tienen una longitud de 16 bits, pero los bits tienen un significado especial.

Los registros selectores contienen valores especiales llamados *selectores*, los cuales apuntan a los recursos del sistema por medio de una tabla llamada *tabla descriptora*. Esta tabla es interpretada por el CPU, pero esta mantenida por el sistema operativo.

Un selector consta de 3 componentes, RPL (Requested Privilel Level), TI (Table Indicator) y INDEX.

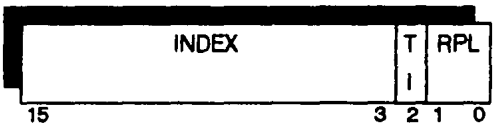


Fig. 3.3. Componentes de un Selector

El **RPL** soporta 4 niveles privilegiados, del 0 al 3. El nivel privilegiado 0 tiene acceso a todos los recursos en el computador, y los niveles del 1 al 3 tienen acceso reducido a los recursos. Los niveles privilegiados son usados para guardar las tareas de bajo nivel en la computadora.

El **TI** indica el segmento para el espacio de direcciones local ó el espacio de direcciones global. 0 para el selector de dirección global y 1 para el selector de dirección local.

INDEX es el valor índice, el cual tiene un rango de 0 a 8,191, es usado como un apuntador dentro de la tabla descriptora, tanto para la GDT ó LDT, dependiendo de TI. Contiene información acerca de los segmentos incluyendo las direcciones de inicio, indica en todo caso, que el valor índice es un apuntador dentro de GDT cuando el bit indicador es 0, ó la LDT cuando el bit indicador es 1.

Hay 3 tipos de tablas descriptoras, a continuación se muestra la lista de tablas descriptoras y sus registros correspondientes:

TABLA DESCRIPTORA	REGISTRO
GDT (Global Descriptor Table)	GDTR
LDT (Local Descriptor Table)	LDTR
IDT (Interrupt Descriptor Table)	IDTR

Fig. 3.4. Tipos de Tablas descriptoras

Generalmente la GDT, define el espacio de direcciones global de la máquina cuando está en modo protegido. El tamaño y direcciones de inicio de la GDT están definidas por valores en un registro especial, el GDTR, el cual deberá de estar inicializado antes de entrar al modo protegido. La LDT, contiene descriptores aplicados para localizar el ambiente de los programas. El tercer tipo de tabla, la IDT, es usado para el manejo de interruptores y no podrá ser accesado por un selector directamente. El registro IDTR contiene las direcciones de inicio y el tamaño del IDT.

Un sistema puede tener sólo una GDT y una IDT, pero puede tener diferentes LDT.

Cada tabla descriptora contiene *descriptores* individuales. Cada descriptor tiene una longitud de 8 bytes, los bits dentro de un descriptor tienen un significado diferente. Hay diferentes tipos de descriptores:

DESCRIPTORES DE SEGMENTO
DESCRIPTORES DE SEGMENTO DEL SISTEMA
DESCRIPTORES DE ENTRADA DEL SISTEMA

Fig. 3.5. Tipos de Descriptores

Los descriptores de segmento, son usados para controlar el acceso a una porción de memoria. Tienen los siguientes campos: Dirección base, Límite y Derecho de acceso.

Los descriptores de segmento del sistema son usados para indicar las direcciones del segmento para varias tablas usadas en memoria y manejo de tareas.

Los descriptores de entrada al sistema son usados para facilitar la transferencia del control entre procesos.

Los descriptores son el corazón de las operaciones en modo protegido, ya que ellos lo describen completamente y controlan todos los aspectos correspondientes a bloques de memoria, ó segmentos. Cada descriptor contiene una dirección de memoria física base para éstos segmentos, la longitud de los segmentos, el nivel privilegido requerido para accesar a los segmentos, y algunos bits definen los atributos usados de los segmentos.

El modelo en modo protegido está definido por el GDT y IDT, contiene descriptores que definen los segmentos de código y datos. Sin embargo éstos no

son suficientes para soportar las multitareas y el espacio de direcciones local para el uso de cada tarea. Para ello son necesarios los descriptores del sistema.

3.3.2. MEMORIA EXTENDIDA

Se usa el término de memoria extendida de RAM para almacenar direcciones por encima de 1 megabyte basado en PCs 80286, 80386 y 80486.

La memoria extendida es el área de memoria arriba de 1 megabyte. Este tipo de memoria no puede ser accesada por la ejecución de un programa en modo real, incluyendo MS-DOS. Para accesar a la memoria extendida el programa tiene que operar en modo protegido. Cuando el programa está operando en modo protegido no puede accesar al BIOS, DOS ú otras funciones del sistema que son fácilmente accesadas en el modo real.

En modo real, el CPU selecciona una localidad particular de memoria por el desplazamiento a la izquierda de cuatro bits del contenido de un registro de segmentos y añadiendo 16 bits de desplazamiento, formando una dirección física de 20 bits. Pero la dirección de la memoria extendida por definición está arriba de 1 megabyte, así todas las direcciones físicas corresponden a la memoria extendida que tiene el bit 21 como menos significativo. En otras palabras, los programas en modo real no pueden operar en memoria extendida, ya que no pueden generar las direcciones apropiadas.

Para poder accesar a la memoria extendida existen diferentes interfaces de software, algunas de ellas son las siguientes:

- La función de acceso de la memoria extendida ROM BIOS (Int 15H función 87H).
- El VDISK ó método de disco virtual.
- La interrupción 15H función 89H ó el método de localización arriba-abajo.
- La interface de software XMS.
- La interface de software VCPI.
- La interface de software DPMI.

La primera cosa que hay que entender acerca de la memoria extendida es que un programa necesita ser ejecutado en modo protegido para leer y escribir en localidades de memoria arriba de 1 megabyte.

El primer paso es pasar del modo real al modo protegido. Los registros se ponen en una GDT (Global Descriptor Table) ésta controla el mapeo de la memoria en modo protegido, tipos de segmento y derechos de accesos; se cargan las direcciones de la tabla en el CPU, y finalmente se cargan todos los registros de segmentos con selectores válidos para referirse al GDT.

En las máquinas 80286, 80386 y 80486, hay dos funciones del BIOS disponibles para accesar a la memoria extendida del modo real. La interrupción 15H, función 88H permite determinar la cantidad que se tiene de memoria extendida, y la interrupción 15H, función 87H la *cual* puede mover un bloque de memoria de algún lugar dentro de la memoria convencional ó memoria extendida a alguna otra localidad.

Para poder desarrollar una interface de software que permita la ejecución de nuestro programa de aplicación se emplearon las funciones del BIOS.

En la siguiente sección se describe el mecanismo de comunicación empleado en el programa de aplicación, para poder operar en la memoria extendida.

3.3.3. CONTROLADOR DE MEMORIA EXTENDIDA

El ROM BIOS PC/AT provee dos funciones las cuales dan acceso a la memoria extendida: Int 15H función 87H, la cual copia un bloque de datos de una localidad en memoria convencional ó extendida a alguna otra localidad, y la interrupción 15H función 88H, regresa la cantidad de memoria extendida instalada en el sistema.

Cuando se llama a la interrupción 15H función 87H, los registros de segmento y desplazamiento apuntan a la tabla GDT, en base a seis descriptores. Los descriptores son los siguientes:

DUMMY
TABLA DE DATOS FUENTE
MOVIMIENTO DE DATOS FUENTE
MOVIMIENTO DE DATOS DESTINO
CODIGO DEL ROM BIOS
PILA DEL ROM BIOS

Fig.3.6. Tabla descriptora usada por el ROM BIOS en la Int 15H función 87H

El primer descriptor dummy corresponde a un selector nulo en el rango 0000 - 0003H. Los selectores nulos obtienen un tratamiento especial del hardware en modo protegido; son valores válidos que siempre se cargan dentro de un registro de segmento.

Dos de los descriptores soportan las direcciones fuente y destino de los bloques de memoria que el programa está pidiendo al ROM BIOS para moverlos.

Los descriptores se inicializan con direcciones base, una longitud apropiada y un acceso correcto del byte 93H. Los tres descriptores restantes son usados por el ROM BIOS para proveer direccionamiento en el código, datos y pila mientras se está ejecutando en modo protegido.

La tabla descriptora tiene direcciones lineales al contrario de los registros de segmento y desplazamiento. Para convertir los registros de segmento y desplazamiento en direcciones lineales, se desplaza el valor en el registro de segmento cuatro bits a la izquierda y se añade el desplazamiento. Los tres bytes de una dirección lineal son almacenados en su orden natural, con el byte menos significativo en la dirección menos significativa.

A continuación se muestra el algoritmo para transferir datos entre memoria convencional y extendida.

Los registros de segmento y desplazamiento apuntan a la GDT.

1. Almacenar los bytes de acceso correcto.
2. Almacenar las direcciones destino en el descriptor.
3. Obtener direcciones lineales:
   [segmento fuente * 16] + [desplazamiento]
4. Almacenar las direcciones fuente en el descriptor.
5. Almacenar el tamaño del bloque de memoria en los descriptores fuente y destino.
6. Convertir el tamaño en palabras.
7. Ejecutar la interrupción 15H función 87H:
   bloque de memoria transferido = Int 15H función 87H
8. Almacenar registros.

Algoritmo 3.1. Copia datos de memoria convencional a memoria extendida.

La codificación del programa que construye la interface para accesar a la memoria extendida, se encuentra en el apéndice B.

3.4. PROGRAMA DE ARCHIVOS MULTIPLES

Debido a todas las herramientas de software empleadas para la codificación del Sistema, el código fuente es tan largo que resulta incoveniente manejarlo en un sólo programa. Un cambio requiere la recompilación de todo el programa.

Para solucionar éste problema se crearon varios programas o archivos pequeños, así se compila cada archivo y se enlazan juntos. Este proceso se conoce como compilación separada y enlazamiento.

En BORLAND C ++, los programas de archivos múltiples se llaman *Project*. Cada project se asocia con un archivo de project que determina qué archivos forman parte del project.

El proceso que se sigue para la compilación es el siguiente, se crea un archivo con extensión .PRJ, dentro de él se crearán los archivos que formen parte del project. Inicialmente BORLAND C++ lee el contenido del archivo .PRJ y cada archivo que se necesita compilar se compila a un .OBJ. A continuación se enlazan estos archivos para su ejecución.

Si hubo modificaciones en alguno de los archivos del project, BORLAND C++ sólo compila los archivos que realmente lo necesitan. Determina esto comprobando la hora y fecha asociada con cada archivo fuente y su archivo .OBJ. Si el archivo en C es más reciente que el archivo .OBJ, BORLAND C++ sabe que el archivo fuente ha sido modificado y lo recompila. Pasa lo mismo con el archivo .EXE. Mientras el archivo .EXE sea más reciente que los archivos .OBJ, no se recompila nada. En cualquier otro caso, los archivos que lo necesitan se recompilan y se reenlaza el project.

Además de comprobar las fechas de los archivos .CPP (extensión de los archivos en BORLAND C++), .OBJ y .EXE, BORLAND C++ comprueba si se ha

cambiado alguno de los archivos de cabecera usados en el programa. Si es éste el caso, cualquier archivo que use un archivo cabecera modificado se recompila automáticamente.

3.5. BIBLIOTECA DE FUNCIONES

Debido a que el lenguaje C no proporciona ningún método para llevar a cabo las operaciones de Entrada/Salida, los programas incluyen llamadas a varias funciones contenidas en la biblioteca estándar del lenguaje C.

Una biblioteca es una colección de funciones. Un archivo de biblioteca guarda el nombre de cada función, el código objeto de la función y la información de reubicación necesaria para el proceso de enlace. Cuando un programa se refiere a una función contenida en una biblioteca, el ligador toma esa función y añade el código objeto en el programa. Así, sólo se añadirán al archivo ejecutable aquellas funciones que realmente se utilicen en el programa.

Muchas de las funciones que se encuentran en la biblioteca del C trabajan con sus propios tipos de datos y variables a los que el programa debe acceder. Estas variables y tipos están definidas en archivos cabecera (utilizando #include), proporcionados junto con el compilador, y deben incluirse en cualquier archivo que utilice funciones específicas que hagan referencia a ellos.

A continuación se muestran los archivos cabecera estándar que utiliza Borland C++ y archivos cabecera específicos de Borland C++, que se emplearon en la codificación del programa.

- **ALLOC.H:** Funciones de asignación dinámica (ANSIC).

- **BIOS.H:** Funciones ROM-BIOS.

- **CONIO.H:** Funciones de manejo de pantalla.

- **CTYPE.H:** Funciones de manejo de caracteres (ANSIC).

- **DIR.H:** Funciones de manejo de directorio.

- **DOS.H:** Funciones de interfaz con el DOS.

- **GRAPHICS.H:** Funciones de gráficos.

- **PROCESS.H:** Contiene estructuras y declaraciones para las funciones spaw y exec.

- **STDIO.H:** Definiciones y declaraciones para secuencias de E/S estándar (ANSI C). Este archivo contiene información necesaria por el programa para asegurar la correcta operación de la función de E/S de la biblioteca de funciones de Borland C++ estándar.

- **STDLIB.H:** Declaraciones variadas (ANSI C) como, rutinas de conversión, rutinas de búsqueda y ordenación, y otras.

- **STRING.H:** Soporte a las funciones de cadena (ANSI C).

CONCLUSIONES

Como resultado de este trabajo de tesis se obtuvo un Sistema, que permite solucionar la problemática en la administración de los horarios en la Jefatura de Ingeniería Mecánica-Eléctrica en la E.N.E.P. Aragón. De esta forma se tiene toda la información de horarios disponible dentro de la Escuela, y así, la manipulación de la información puede efectuarse en un mínimo de tiempo.

El desarrollo de este Sistema permite a la Jefatura de carrera y a la Escuela, obtener una solución a su problema de administración de horarios a un bajo costo, comparado con la inversión que hubiese tenido que hacer si hubiera contratado los servicios de algún programador o empresa de software.

Además, la Escuela poseerá el código fuente y el código ejecutable, de esta manera puede dar mantenimiento al Sistema en el momento que se necesite.

El Sistema se elaboró para la Jefatura de Ingeniería Mecánica-Eléctrica, pero es posible utilizarlo para cualquier otra carrera que tenga requerimientos similares.

El Sistema está diseñado para ser instalado en máquinas PC con microprocesador 80286, 80386 u 80486, con un mínimo de 2 Megabytes en memoria RAM y monitor VGA cromático. Utiliza estructuras de datos dinámicas que permiten una rápida ejecución. Cuenta con un módulo de funciones logrando direccionar la memoria física por encima de 1 Megabyte. De esta forma se obtiene un considerable ahorro en tiempo.

El Sistema esta distribuido en varios programas, esto, ahorra tiempo en la compilación y detección de errores. Así, se facilita el mantenimiento al programa. Para poder ejecutar el código fuente es necesario utilizar el compilador BORLAND C++ versión 3.1, o alguna versión posterior.

Este Sistema puede ser ejecutado por usuarios con un mínimo de conocimientos de software. Cuenta con mensajes de información constante, de ayuda y error en todos los módulos. De esta manera no se requiere el uso de manuales adicionales para su operación.

El usuario puede consultar la información de horarios en un mínimo de tiempo y con eficiencia, también puede obtener la impresión de los datos consultados en el formato establecido por la Jefatura de carrera.

En ocasiones es necesario enviar parte de la información al Sistema Central de Administración Escolar de la UNAM, para ello se creó un módulo que guarda, en código ASCII, esta información siguiendo un protocolo de comunicación ya establecido por la UNAM.

APENDICE A

A.1. ARQUITECTURA DEL 80386

El microprocesador 80386 es un procesador de 32 bits diseñado para soportar aquellos sistemas operativos optimizados para multitarea. El 80386 soporta registros, direcciones y tipos de datos de 32 bits.

El microprocesador 80386 es capaz de direccionar hasta 4 gigabytes de memoria física y 64 tetrabytes, de memoria virtual. La introducción de memoria integrada y arquitectura de protección incluye registros de traducción de direcciones y mecanismos de protección para soportar sistemas operativos y hardware avanzado de multitarea.

Características adicionales incluyen mecanismos de autoprueba, acceso directo a memoria intermedia (caché) donde se realiza la traducción de página, además es capaz de ejecutar instrucciones a una frecuencia de 3 a 4 millones por segundo. El 80386 tiene el código objeto compatible con el 8086/8088/80286.

El 80386 soporta varios tipos de datos además de los soportados por el 8086/80286. El microprocesador 80386 soporta enteros con signo y sin signo de 32 bits y campos de bits de 1 a 32 bits de longitud. El microprocesador 80386 soporta los tipos de apuntadores estándar, definidos para la familia 8086/80286, así como un apuntador de desplazamiento de 32 bits y un apuntador completo de 48 bits.

Además de los operandos inmediatos de 8 y 16 bits soportados por el 8086/80286/80386, el 80386 también soporta operandos de 32 bits, con la indicación general que el campo de operando inmediato de 16 bits se extiende a 32 bits.

Cuando se accesa a un segmento mayor de 64 K, las direcciones pueden ser de 32 bits o de 16 bits. La dirección se forma sumando un registro base opcional, un registro índice opcional y un desplazamiento. Además, los modos de direccionamiento de 32 bits han sido expandidos para permitir el uso de registros de propósito general como registro base o índice.

El microprocesador 80386 tiene dos modos de operación que son compatibles con el conjunto de instrucciones 8086/80286. Se suministra para ejecutar el código objeto del 8086. El modo real se suministra como el del 80286, y es el modo utilizado por el microprocesador después de un *RESET*. El modo virtual del 8086 es un subconjunto del modo protegido del 80386 y permite que el código del 8086 se ejecute en modo protegido y en el entorno de operación paginado suministrado por el 80386.

Los miembros de la familia Intel tienen áreas especiales de memoria llamados registros, los cuales son construídos dentro del microprocesador. El 80386 proporciona 32 registros. Estos 32 registros pueden ser divididos en seis categorías principales.

- Registros de propósito general.
- Registros de segmentos.
- Apuntadores de instrucción y señalizadores.
- Registros de control.
- Registros de direcciones del sistema.
- Registros de prueba.

Todos los registros de 16 bits del 8086 y 80286 están contenidos en el microprocesador de 32 bits.

A.1.1. REGISTROS DE PROPOSITO GENERAL

Los registros de propósito general son capaces de soportar operandos de datos de 1, 8, 16 y 32 bits y campos de bits de 1 a 32 bits. Los registros son los siguientes: EAX (acumulador); EBX (base); ECX (contador); EDX (datos); ESP (apuntador de pila); EBP (apuntador base), ESI (índice fuente) y EDI (índice destino).

A.1.2. REGISTROS DE SEGMENTO

El 80386 contiene seis registros de segmento de 16 bits, los cuales retienen los valores en el selector en las posiciones de memoria actualmente direccionables. En modo de dirección real, un segmento puede variar desde 1 byte hasta un tamaño de segmento máximo de 64 K bytes. El direccionamiento en modo protegido habilita rangos de segmento desde 1 byte hasta un máximo de 4 gigabytes.

Los registros de segmento son el CS (segmento de código), DS (segmento de datos), SS (segmento de pila), ES (segmento extra) y FS y GS.

A.1.3. APUNTADORES DE INSTRUCCION Y SEÑALIZADORES

El 80386 contiene un registro apuntador de instrucción de 32 bits llamado EIP. El EIP contiene el desplazamiento de la siguiente instrucción que se va a ejecutar. Este desplazamiento es siempre relativo a la base del CS actualmente activo (segmento código). Los 16 bits de orden inferior del EIP pueden ser accedidos separadamente. A estos bits se les denomina registro IP y se utilizan en direccionamientos de 16 bits.

El registro EFLAGS se utiliza para controlar ciertas operaciones e indicar el status del mismo 80386. El registro EFLAGS contiene dos nuevos señilizadores: VM (Modo virtual 8086) y RF (señalizador de Resumen). Los 16 bits de orden inferior del registro EFLAGS, llamado registro FLAGS, contienen el mismo control de operaciones y los mismos señalizadores de status asociados a los microprocesadores 8086/80286.

El señalizador VM (señalizador Modo virtual 8086) localizado en el registro EFLAGS habilita el modo virtual del 8086. Si VM está a 1 y el 80386 está en modo protegido, el microprocesador conmutará a la operación en modo virtual del 8086, haciendo que todas las operaciones de segmento se ejecuten como si se estuviese corriendo en un 8086. Mientras se emula el 8086, el 80386 también genera fallos para códigos de operación privilegiados.

A.1.4. REGISTROS DE CONTROL

El 80386 contiene cuatro registros de control de 32 bits, CR0, CR1, CR2 Y CR3. Estos registros contienen información sobre el status no dependiente de la tarea de la máquina. El acceso a los registros de control se realiza a través de instrucciones de carga y almacenamiento.

El registro CR0 contiene seis señalizadores predefinidos usados para el control del microprocesador y propósitos de status. Los bits 0 a 15 del registro CR0 también son conocidos como el MSW (palabra de status de la máquina), haciendo así al 80386 compatible con el 80286 en modo protegido.

PE (protection enable). La habilitación de protección se utiliza para activar el modo protegido del microprocesador. Si PE está a 0, el procesador opera en modo de direccionamiento real. Si PE está a 1, se activa el modo protegido.

MP (monitor coprocessor). El coprocesador monitor es utilizado, junto al bit TS, para determinar si el código de operación WAIT generará un fallo de coprocesador no disponible cuando TS = 1.

EM (emulate). El coprocesador de emulación se inicializa a 1 para hacer que todos los códigos de operación del coprocesador generen un fallo de coprocesador no disponible. Si EM está a 0, todos los códigos de operación del coprocesador serán ejecutados en un coprocesador real 80387.

TS (task switched). El conmutador de tareas se inicializa automáticamente a 1 cuando se realiza una operación de conmutación de tareas. Con TS a 1, un código de operación del coprocesador provocará una trampa de coprocesador no disponible.

PG (paging). El bit PG es habilitado, siempre que el mecanismo de paginación esta activado.

CR1 se reserva para futuros microprocesadores Intel. CR2 contiene la dirección lineal de 32 bits que provoca la detección de fallo de la última página. CR3 contiene la dirección base física de la tabla del directorio de página. Ya que la tabla de direcctorio de página del 80386 es siempre de página-alineada, los 12 bits de orden inferior del CR3 se ignoran cuendo escribe.

El 8086 admite dos tipos básicos de memoria de bloques: segmentos y páginas. Mientras un segmento puede tener alguna longitud, cada página tiene 4 K de longitud. Una descomposición de la memoria en páginas puede ser independiente de la descomposición en segmentos: un segmento podrá residir en una página, mientras que una página podrá contener partes de varios segmentos. La información completa de páginas almacenadas en estructuras de datos especiales es llamada directorio de páginas y tablas de páginas.

A.1.5. REGISTROS DE DIRECCIONES DEL SISTEMA

El microprocesador 80386 contiene cuatro registros de propósito especial que son usados para referenciar las tablas o segmentos soportados por el modo de protección de 80286/80386. Las direcciones de la GDT (tabla descriptor global), IDT (tabla de descriptor de interrupción), LDT (tabla de descriptor local) y TSS (tabla de segmento de estado de tarea) se almacenan en estos registros especiales. Los registros segmento del sistema y de direcciones del sistema se denominan GDTR, IDTR, LDTR Y TR.

Los registros GDTR e IDTR son registros de 32 bits que mantienen la dirección base lineal de 32 bits y el límite de 16 bits del GDT e IDT. Los registros LDTR y TR son registros de 16 bits que contienen el selector de 16 bits para los segmentos LDT y TSS.

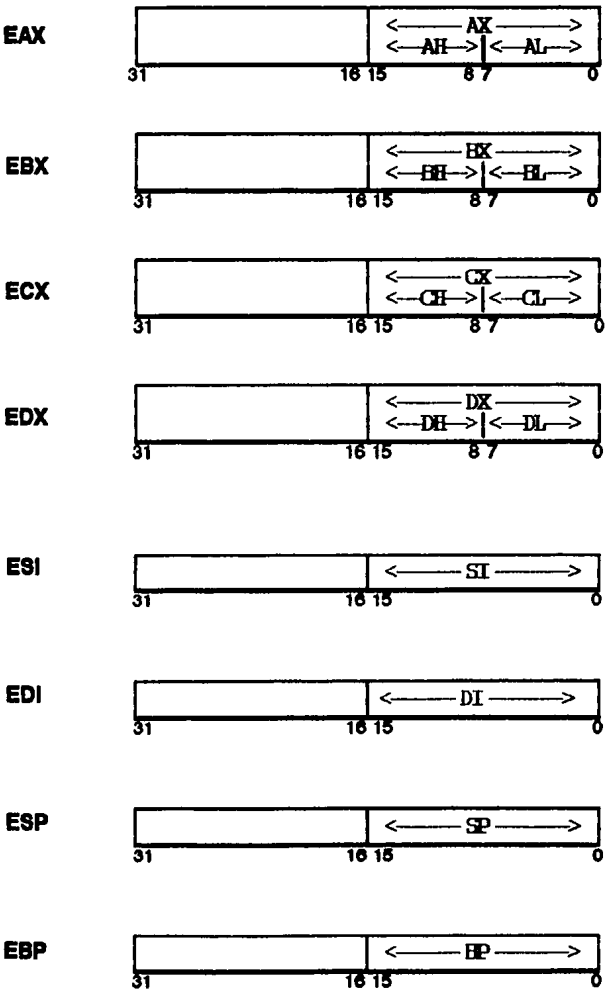
A.1.6. REGISTROS DE DEPURACION Y PRUEBA

El microprocesador permite accesar a seis registros de depuración de 32 bits DR0, DR1, DR2, DR3, DR6, DR7. DR4 y DR5 están reservados para Intel. El registro de control de depuración DR6 se utiliza para inicializar los breakpoints. DR7 visualiza el estado actual de los breakpoints.

El microprocesador 80386 contiene dos registros de prueba de 32 bits que son utilizados para controlar la comprobación de las RAM y CAD (memorias de contenido direccionable) en el buffer de traducciones previas. TR6 se utiliza como registro de comprobación de órdenes y TR7 funciona como registro de datos que mantiene los datos contenidos en la prueba del buffer de traducciones previas.

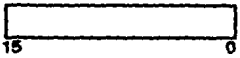
A.2. REGISTROS DEL 80386 EN DIAGRAMAS DE BLOQUES

A.2.1. REGISTROS DE PROPOSITO GENERAL

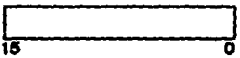


A.2.2. REGISTROS DE SEGMENTO

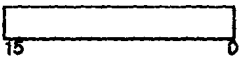
CS



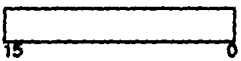
DS



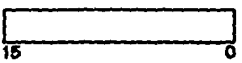
ES



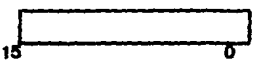
FS



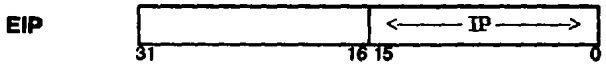
GS



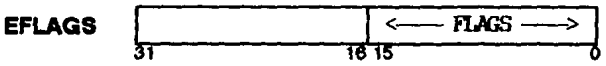
SS



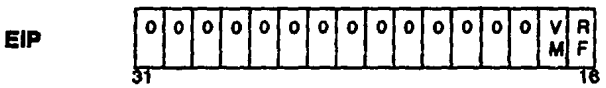
A.2.3. APUNTADOR DE INSTRUCCIONES



A.2.4. REGISTROS EFLAGS



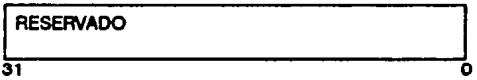
DETALLE DE LOS REGISTROS EFLAGS



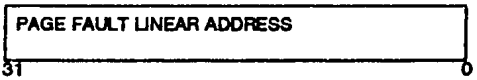
A.2.5. REGISTROS DE CONTROL

CR0

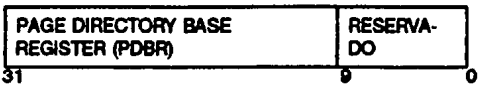
CR1



CR2

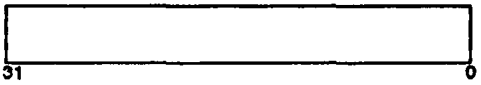


CR3

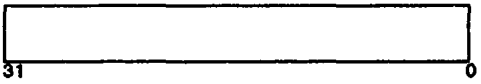


A.2.6. REGISTROS DE DIRECCION DEL SISTEMA

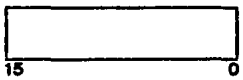
GDTR



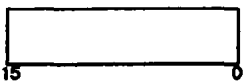
IDTR



LDTR

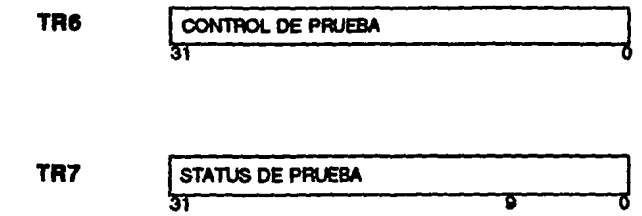


TR



A.2.7. REGISTROS DE DEPURACION

A.2.8. REGISTROS DE PRUEBA



APENDICE B

B.1. SISTEMA PARA LA ADMINISTRACION DE LOS HORARIOS DE LA CARRERA DE INGENIERIA MECANICA ELECTRICA

El Sistema permite manipular la información de los horarios de la carrera de Ingeniería Mecánica Eléctrica. Cuenta con las siguientes características:

- Permite actualizar la información de los horarios en un mínimo de tiempo.

- Permite la consulta de datos por Materia. El usuario puede consultar en pantalla todas las materias de la carrera de Ingeniería Mecanica Eléctrica. Al seleccionar una materia de la lista, visualizará en pantalla, la clave y nombre de la materia seleccionada, el tipo de materia (optativa ú obligatoria), tipo de laboratorio (laboratorio obligatorio, optativo ó sin laboratorio), la lista de grupos asignados a la materia y el nombre del profesor que imparte clase en cada uno de los grupos.

- Permite la consulta de datos por Grupo. El usuario puede consultar la lista de grupos de teoría, grupos de laboratorio ó ambos, para una materia ó para todas las materias de la lista. La selección de un grupo muestra en pantalla el número de grupo, el tipo (grupo de teoría o de laboratorio), el nombre de la materia a la que está asignado el grupo, el nombre y R.F.C. del profesor que imparte clase en el grupo, el número de salón, el total de alumnos inscritos, los días y horario en que se imparte clase en el grupo.

- Permite la consulta de datos por Profesor. El usuario puede consultar la lista de profesores titulares, ayudantes de profesor, profesores sin nombramiento ó todos los profesores, para una materia ó para todas las materias. Al seleccionar un profesor se visualizará en pantalla, el R.F.C. y nombre del profesor seleccionado, la clave y nombre de la materia en la cual imparte clase y la lista de los grupos que tiene asignados. El usario puede desplegar por páginas la lista de grupos de una materia. También puede cambiar a la siguiente materia o a la materia anterior.

- Permite la consulta de datos por Salón. El usuario puede consultar la lista de salones de dibujo, laboratorio, chicos ó grandes, para una ó todas las materias. Al seleccionar un salón se visualizará en pantalla el número de salón seleccionado, el tipo (salón de dibujo, laboratorio, chico ó grande), la clave y nombre de la materia y la lista de grupos asignados que se imparte en el salón. El usuario puede desplegar por páginas la lista de grupos de una materia. También puede cambiar a la siguiente materia ó a la materia anterior.

- Permite la consulta de datos de Exámenes Final y Extraordinario. Al seleccionar un exámen se desplegará en pantalla el nombre y clave de la materia, el nombre y R.F.C. de profesor que aplicará el exámen, en el caso de exámen extraordinario aparecerá el nombre y R.F.C. de profesor titular y profesor suplente, el salón, fecha y horario en que se aplicará el exámen.

- Tiene el módulo Semestre por separado para actualizar la información ya existente.

- El Sistema cuenta con un módulo de ayuda, mensajes informativos y mensajes de error en todos los módulos.

- La comunicación entre el usuario y el Sistema se lleva a cabo por medio de ventanas gráficas, utilizando distintos tipos de fonts y colores, para facilitar la operación al usuario.

- El Sistema cuenta con un mecanismo de comunicación entre la memoria convencional y la memoria extendida.

B.1.1. COMPONENTES DE SOFTWARE DEL SISTEMA

El Sistema de horarios consiste de un archivo ejecutable LISTA.EXE y de un sudirectorio AYUDA.

El sudirectorio AYUDA contiene los archivos que proporcionan toda la información necesaria de cada módulo del Sistema. Durante la ejecución del Sistema el usuario puede llamar al módulo AYUDA presionando la tecla F1.

B.1.2. REQUERIMIENTOS DEL SISTEMA

El Sistema está diseñado para ejecutarse en una computadora con microprocesador 80286, 80386 ú 80486, con un mínimo de 2 Mbytes de memoria RAM, con monitor VGA color, un disco duro y una impresora estándar de 15 pulgadas. Las impresiones requieren de hojas tamaño doble carta.

B.2. RECOMENDACIONES

Para evitar que el Sistema tenga fallas durante su ejecución es necesario verificar lo siguiente:

- **Disco RAM.** Si el disco RAM está creado, es necesario eliminarlo, esto se logra editando el archivo CONFIG.SYS. Se escribe la instrucción *rem* en la línea que contiene la instrucción RAMDRIVE, se guarda el archivo en el disco duro y se inicializa nuevamente el computador (apagar y volver a encender el computador).

- **Programas Residentes.** Si se tienen cargados programas residentes tales como Windows, Norton, etc., es necesario no ejecutarlos, para tener el espacio de memoria convencional libre; para lograrlo se edita el archivo CONFIG.SYS, se escribe *rem* en la línea que contenga la instrucción de ejecución de algún programa residente, se guarda el archivo en el disco duro y se inicializa nuevamente el computador (apagar y volver a encender el computador).

B.3. CODIFICACION DEL SISTEMA DE HORARIOS

El Sistema de horarios está codificado en lenguaje C, utilizando para ello el compilador BORLAND C++ versión 3.1. Pueden utilizarse versiones posteriores.

El Sistema es muy extenso, debido a ello se crearon varios bloques incluídos en un PROJECT. Cada bloque corresponde a un programa. Se crearon 16 programas:

- ARCHIVOS.CPP
- AYUDA.CPP
- CONSULTA.CPP
- CUADROS.CPP
- ERRORES.CPP
- EXAMEN.CPP
- E_TODOS.CPP
- GRUPOS.CPP
- G_TODOS.CPP
- IMPRIME.CPP
- MENU.CPP
- OPER.CPP
- P_TODOS.CPP
- VENTANAS.CPP
- VENTDOS.CPP
- EXTMEM.ASM

Es necesario enlazar juntos los distintos programas, con el fin de acelerar la ompilación, para ello se crearon archivos cabecera que contienen las variables globales requeridas por el programa. Se crearon 15 archivos cabecera.

- ARCHIVOS.H
- AYUDA.H
- CONSULTA.H
- CUADROS.H
- ERRORES.H
- EXAMEN.H
- E_TODOS.H
- GRAFICO.H
- GRUPO.H
- G_TODOS.H
- IMPRIME.H
- OPER.H
- P_TODOS.H
- VENTANAS.H
- VENTDOS.H

B.3.1. ARCHIVOS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\consulta.h"
#include "d:\errores.h"
#include "d:\ayuda.h"
#define  then


void cambia_elementos_nombre(struct MATERIA *p, int k, struct ARREGLO *arreglo)
{
 struct MATERIA *s,m;

 m.clave = (*p).clave;
 strcpy(m.nombre,(*p).nombre);
 m.lab = (*p).lab;
 m.grupos = (*p).grupos;
 m.examen = (*p).examen;
 s = p;
 while (strcmp((*s).nombre,arreglo[k].nombre) && (s != NULL)) do
     s = (*s).materias;

 (*p).clave = (*s).clave;
 strcpy((*p).nombre,(*s).nombre);
 (*p).lab = (*s).lab;
 (*p).grupos = (*s).grupos;
 (*p).examen = (*s).examen;
 while ((*p).num != ((*s).num)) do
   p = (*p).materias;
 (*p).clave = m.clave;
 strcpy((*p).nombre,m.nombre);
 (*p).lab = m.lab;
 (*p).grupos = m.grupos;
 (*p).examen = m.examen;
}


//     Coloca los elementos de la lista enlazada MATERIA en orden creciente.
void ordena_materias_clave(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 int k;

 r = p;
 k = 0;
 while ((arreglo[k].numero != -2) && (r != NULL)) do
   {
     while (strcmp(arreglo[k].nombre,(*r).nombre)) do
           cambia_elementos_nombre(r,k,arreglo);
```

```
    k = k + 1;
    r = (*r).materias;
   }
}

/*   Compara el dato cadena introducido desde el teclado con cada dato  del mismo tipo contenido
  en la lista enlazada MATERIA  Si  existe un dato en la lista enlazada igual a la cadena introducida
  desde el teclado, se  envía un mensaje de error.*/
int error_clave_igual(struct MATERIA *p, char cadena[100])
{
 struct MATERIA *r;
 int valor, numero,k,n;

 valor = 0;
 for (k=0; cadena[k]; k++)                          // Verifica que el dato sea de tipo ente-
 {                                                  // ro, de lo contrario envía un  mensaje
  if (isdigit(cadena[k]) == 0) then                 // de error.
   {
    valor = 1;
    n = 18;
   }
 }
 numero = atoi(cadena);
 r = p;
 while (r != NULL) do
 {
  if ((*r).clave == numero) then                    // Compara cada dato de la lista  con  el
   {                                                // entero numero.  Si  se encuentra  un
    valor = 1;                                      // encuentra un dato igual a  numero  se
    n = 20;                                         // envía un mensaje de error.
   }
  r = (*r).materias;
 }
 if (valor == 1) then
  llama_error(n);
 return(valor);
}

/*   Compara el dato cadena introducido desde el teclado con cada dato del mismo tipo contenido en
 la lista enlazada MATERIA.  Si existe un dato en la lista enlazada igual a la cadena introducida desde
 el teclado, se  envía un mesaje de error.*/
int error_nombre_igual(struct MATERIA *p, char cadena[100])
{
 struct MATERIA *r;
 int valor,i;

 r = p;
 for (i = 0; cadena[i]; i++)
  cadena[i] = toupper(cadena[i]);
 valor = 0;
 while (r != NULL) do                               // Se compara cada dato de la lista  con
 {                                                  // cadena. Si se encuentra un dato igual
  if (!strcmp((*r).nombre,cadena)) then             // a  cadena  se  envía  un  mensaje  de
   valor = 1;                                       // error.
  r = (*r).materias;
 }
 if (valor == 1) then
  llama_error(19);
 return(valor);
}
```

```
/*    Compara el dato cadena introducido desde el teclado con cada dato  del mismo tipo contenido
  en la lista enlazada MATERIA.  Si existe un dato en la lista enlazada igual a la cadena  introducida
  desde el teclado, se  envía un mensaje de error.  */
int error_clave_cambiar(struct MATERIA *p, char cadena[100], int dato)
{
 struct MATERIA *r;
 int valor, numero,n,k;

 valor = 0;
 for (k=0; cadena[k]; k++)                          // Verifica que el dato sea de tipo  ente-
 {                                                  // ro, de lo contrario envía  un  mensaje
  if (!isdigit(cadena[k]) == 0) then                // de error.
  {
    valor = 1;
    n = 18;
  }
 }
 numero = atoi(cadena);
 r = p;
 while ((*p).num != dato) do
    p = (*p).materias;
 (*p).clave = 0;
 while (r != NULL) do
 {
  if ((*r).clave == numero) then                    // Compara cada dato de la lista con  el
  {                                                 // entero numero.  Si  se  encuentra  un
    valor = 1;                                      // dato igual a numero se envía un men-
    n = 20;                                         // saje de error.
  }
  r = (*r).materias;
 }
 if (valor == 1) then
  llama_error(n);
 return(valor);
}


/*    Compara el dato cadena introducido desde el teclado con cada dato del mismo tipo contenido
  en la lista enlazada MATERIA.  Si existe un dato en la lista enlazada igual a la cadena introducida
  desde el teclado, se  envía un mensaje de error.*/
int error_nombre_cambiar(struct MATERIA *p, char cadena[100], int dato)
{
 struct MATERIA *r;
 int valor,i;

 r =p;
 for (i = 0; cadena[i]; i++)
  cadena[i] = toupper(cadena[i]);
 while ((*p).num != dato) do
    p = (*p).materias;
 strcpy((*p).nombre,"vacio");
 valor = 0;
 while (r != NULL) do                               // Compara cada dato del mismo tipo de
 {                                                  // la lista con cadena.  Si se encuentra un
  if (!strcmp((*r).nombre,cadena)) then             // dato igual a cadena se envía  un  men-
    valor = 1;                                      // saje de error.
  r = (*r).materias;
 }
 if (valor == 1) then
  llama_error(19);
 return(valor);
}
```

```
/*    Confirma la operación a realizar. Si la respuesta es SI, entonces se realizará la operación de
  eliminación. Si la respuesta es  NO,  entonces se regresará a la función que efectuó la llamada.*/
int confirmar(int n)
{
 int i, valor;

 pon(tope,352,COPY_PUT);                                    // Pone ventana 352
 switch(n)                                                  // n selecciona el texto dentro de la ven-
  {                                                         // tana.
   case 1:texto("Al crear eliminará la lista existente. Desea continuar ...",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,3);
          break;
   case 2:texto("Esta seguro de querer eliminar ...",352,LIGHTRED,SANS_SERIF_FONT,1,2,1,1);
          break;
  }
 pon(tope,253,COPY_PUT);                            // Pone ventana gráfica 253
 pon(tope,254,COPY_PUT);                            // Pone ventana gráfica 254
 pon(tope,150,COPY_PUT);                            // Pone ventana gráfica 150
 texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);      // Pone texto en la ventana gráfica 253
 texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);      // Pone texto en la ventana gráfica 254
 cuadro(150,BLUE);                                  // Rellena la ventana 150 con color azul
 texto("Presione < ESC > para salir ....",248,LIGHTBLUE,SMALL_FONT,3,2,2,1);
 pon(tope,253,NOT_PUT);                             // Pone cursor en la ventana 253
 i = opciones1(253,253,254,9);                      // Mueve la pila de ventanas
 quita(tope);                                       // Quita cursor
 quita(tope);                                       // Quita ventana 150
 quita(tope);                                       // Quita ventana 254
 quita(tope);                                       // Quita ventana 253
 quita(tope);                                       // Quita ventana 352
 valor = 0;
 if (i == 254) then
  {
   valor = 1;
  }
 return(valor);
}

//  Pone el texto formateado en las ventanas del menú ARCHIVOS
void ventanas_archivos(void)
{
 cuadro(30,GREEN);
 cuadrito(31,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"LEER");
 cuadrito(32,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"ESCRIBIR");
 cuadrito(33,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CREAR");
 cuadrito(34,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"AGREGAR");
 cuadrito(35,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"ELIMINAR");
 cuadrito(36,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CAMBIAR");
}

/*    Selecciona un archivo de entre todos los que aparecen en pantalla. Regresa -1  si  no  se  quiso
  leer; regresa -2 si no hubo archivo que leer; regresa 0, 1, 2, ..., según la posición que se ocupe en
  la cadena de nombres de archivos; pone los nombres de los archivos  en  la  variable archivos.*/
int selecciona_archivo(char archivos[20][15], char *cadena, int *n)
{
 struct ffblk      r;
 int               k, i, j, m;
                                                            // Busca archivos con la cadena seleccio-
 k = 0;                                                     // nada
 i = 71;
 j = findfirst(cadena,&r,FA_ARCH);
 if (j == 0) then
   {
    cuadrito(i,RED,SMALL_FONT,1,1,2,1,r.ff_name);
    strcpy(archivos[k],r.ff_name);
    j = -1;                                                 // Busca en archivos siguientes
    while (j == -1)
```

```
    {
        while (!(m = findnext(&r)))
          {
           i++;
           cuadrito(i,RED,SMALL_FONT,1,1,2,1,r.ff_name);
           k++;
           strcpy(archivos[k],r.ff_name);
           if (i == 90) then break;
          }
        if ((i == 70) && (m == -1))
          return(-1);
        *n) = i;                                    // Indica cual fué la última ventana de ar-
        pon(tope,71,NOT_PUT);                       // chivos.
        j = opciones1(71,71,i,8);
        quita(tope);
        if (j == -1) then
         {
          for (j = 71; j <= *n; j++)
                {
                 setfillstyle(SOLID_FILL,GREEN);
                 bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                }
          return(-3);
         }
        if (j >= 71) then
          return(j - 71);                           // Se eligió una opción.
        else
         {
                                                    // Quita las ventanas anteriores
          for (j =71; j <= *n; j++);
                {
                 setfillstyle(SOLID_FILL,GREEN);
                 bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                }
          k = 0;
          i = 70;
          j = -1;
          if (i < 90) then
            return(-1);
         }
    }
  }
return(-2);
}


//    Lee un archivo en modo binario en disco según la trayectoria que se haya establecido.
void leer(struct MATERIA *p, char semestre[10])
{
int       i, j, n, k;
char      directorio[20], cadena[100], nombre[20], archivos[20][15];
char      drive[15], dir[15], name[15], ext[15], *s;

i = getdisk();                                      // Obtiene la trayectoria del subdirectorio
switch(i)                                           // actual.
 {
  case 0: strcpy(directorio,"A:\\"); break;
  case 1: strcpy(directorio,"B:\\"); break;
  case 2: strcpy(directorio,"C:\\"); break;
  case 3: strcpy(directorio,"D:\\"); break;
  case 4: strcpy(directorio,"E:\\"); break;
  case 5: strcpy(directorio,"F:\\"); break;
 }
                                                    // Pone la ventanas gráficas para el sub-
getcurdir(i+1,cadena);                              // menú LEER.
```

```
strcat(directorio,cadena);
cuadrito(63,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"LECTURA");
cuadrito(64,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"DIRECTORIO");
cuadrito(65,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"BUSCAR");
cuadrito(66,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"SALIR");
cuadrito(67,BLUE,SMALL_FONT,1,1,2,1,directorio);
i = 63;
pon(tope,i,NOT_PUT);
while ((i != 66) && (i != -1))
 {
  i = opciones1(i,63,66,7);                    // Mueve la pila de ventanas.
  switch(i)
   {
                                               // El case 63 realiza la operación de LECTURA
        case 63:
                strcpy(nombre,"*.tp");         // Indica la cadena a leer.
                strcpy(cadena,"Leyendo ");
                strcat(cadena,nombre);
                strcat(cadena,". . . ");
                cuadrito(62,RED,SMALL_FONT,2,1,2,1,cadena);
                j = selecciona_archivo(archivos,nombre,&n);
                if (j == -3) then
                 {
                  k = 62;
                  setfillstyle(SOLID_FILL,GREEN);
                  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                 }
                if (j == -2) then
                  {                            // No hubo archivo a seleccionar
                   pon(tope,68,COPY_PUT);      // Pone ventana gráfica 68
                   cuadrito(68,BLACK,SMALL_FONT,1,1,2,1,"NO HAY ARCHIVOS A SELECCIONAR");
                   pon(tope,69,COPY_PUT);      // Pone ventana gráfica 69
                   cuadrito(69,LIGHTGRAY,SMALL_FONT,1,1,2,1,"PRESIONE UNA TECLA PARA SALIR");
                   blockey(0);
                   quita(tope);                // Quita ventana gráfica 69
                   quita(tope);                // Quita ventana gráfica 68
                    k = 62;
                        setfillstyle(SOLID_FILL,GREEN);
                        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                  }
                if (j >= -1) then
                  {                            // Lee el archivo con la trayectoria establecida
                   strcpy(cadena,"Leyendo archivo ");
                   strcat(cadena,archivos[j]);
                   strcat(cadena,"...");
                   pon(tope,62,COPY_PUT);      // Pone ventana gráfica 62
                   cuadrito(62,RED,SMALL_FONT,2,1,2,1,cadena);
                   pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                   texto("Presione una tecla para continuar . . .",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                   blockey(0);
                   quita(tope);                // Quita ventana gráfica 179
                   quita(tope);                // Quita ventana gráfica 62
                   s=archivos[j];
                   lee_lista_materias(p,s,semestre);
                        // Quita todas las ventanas de los archivos desplegados
                   for (j = 71; j <= n; j++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                        }
                   j = 62;
                   bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                  }
                break;
```

```
                // El case 64 establece el NUEVO DIRECTORIO
        case 64: pon(tope,66,COPY_PUT);                 // Pone ventana gráfica 66
                 cuadrito(66,BLUE,SANS_SERIF_FONT,1,2,1,1,"NUEVO DIRECTORIO . . .");
                 pon(tope,69,COPY_PUT);                 // Pone ventana gráfica 69
                 lee_cadena(cadena,25,69,RED,SANS_SERIF_FONT,1,3,1,1);
                 quita(tope);                           // Quita ventana gráfica 69
                 quita(tope);                           // Quita ventana gráfica 66
                                                        // Cambia a mayúsculas la trayectoria
                   n = strlen(cadena);
                   for (j = 0; j < n; j++)
                         cadena[j] = toupper(cadena[j]);
                                                        // Establece el nuevo disco
                 fnsplit(cadena,drive,dir,name,ext);
                 switch(drive[0])
                  {
                    case 'A': j = 0; break;
                    case 'B': j = 1; break;
                    case 'C': j = 2; break;
                    case 'D': j = 3; break;
                    case 'E': j = 4; break;
                    case 'F': j = 5; break;
                  }
                 j = setdisk(j);
                                                        // Establece el nuevo directorio
                 if (chdir(cadena)) then
                   {
                    llama_error(15);                    // No se estableció el nuevo directorio
                   }
                 else
                   {
                                                        // Cambia a mayúsculas la trayectoria
                    n = strlen(cadena);
                    for (j = 0; j < n; j++)
                          cadena[j] = toupper(cadena[j]);
                                                        // Pone nuevo directorio
                    cuadrito(67,BLUE,SMALL_FONT,1,1,2,1,cadena);
                   }
                 break;

// El case 65 realiza la operación de NUEVA CADENA A BUSCAR PARA LECTURA DE //ARCHIVOS *.*
        case 65: pon(tope,68,COPY_PUT);                 // Pone ventana gráfica 68
                 cuadrito(68,BLUE,SANS_SERIF_FONT,1,2,1,1,"CADENA A BUSCAR . . .");
                 pon(tope,69,COPY_PUT);                 // Pone ventana gráfica 69
                 lee_cadena(cadena,12,69,RED,TRIPLEX_FONT,9,10,1,1);
                 quita(tope);                           // Quita ventana gráfica 69
                 quita(tope);                           // Quita ventana gráfica 69
                                                        // Indica la nueva cadena a leer
                 strcpy(nombre,cadena);
                 strcpy(cadena,"Leyendo ");
                 strcat(cadena,nombre);
                 strcat(cadena," . . . ");
                 cuadrito(62,RED,SMALL_FONT,2,1,2,1,cadena);
                 j = selecciona_archivo(archivos,nombre,&n);
                 if (j == -2) then
                   {
                                                        // No hubo archivo a seleccionar
                    setfillstyle(SOLID_FILL,GREEN);
                    k = 62;
                    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                    pon(tope,66,COPY_PUT);              // Pone ventana gráfica 66
                    cuadrito(66,BLACK,SMALL_FONT,1,1,2,1,"NO HAY ARCHIVOS A SELECCIONAR!");
                    pon(tope,69,COPY_PUT);              // Pone ventana gráfica 69
                    cuadrito(69,LIGHTGRAY,SMALL_FONT,1,1,2,1,"PRESIONE UNA TECLA PARA SALIR!");
                    bioskey(0);
                    quita(tope);                        // Quita ventana gráfica 69
```

```
                    quita(tope);                        // Quita ventana gráfica 69
                   }
                 if (j >= -1) then
                   {
                    strcpy(cadena,"Leyendo archivo ");
                    strcat(cadena,archivos[j]);
                    strcat(cadena," . . .");
                    pon(tope,62,COPY_PUT);              // Pone ventana gráfica 62
                    cuadrito(62,RED,SMALL_FONT,2,1,2,1,cadena);
                    pon(tope,179,COPY_PUT);             // Pone ventana gráfica 179
                    texto("Presione una tecla para continuar . . .",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    bioskey(0);
                    quita(tope);                        // Quita ventana gráfica 179
                    quita(tope);                        // Quita ventana gráfica 62
                    s=archivos[j];
                    lee_lista_materias(p,s,semestre);
                        // Quita todas las ventanas de los archivos desplegados
                    for (j = 71; j <= n; j++)
                          {
                           setfillstyle(SOLID_FILL,GREEN);
                           bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                          }
                    j = 62;
                    bar(vent[j].x1,vent[j].y1,vent[j].x2,vent[j].y2);
                   }
                    k = 62;
                        setfillstyle(SOLID_FILL,GREEN);
                        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                 break;
                // El case 66 realiza la operación SALIR DE LA OPCION 'LEER'
        case 66: pon(tope,68,COPY_PUT);         // Pone ventana gráfica 68
                 cuadrito(68,BLUE,SANS_SERIF_FONT,1,2,1,1,"SALIENDO . . .");
                 delay(1000);                   // Detiene la ejecución en tiempo de un segundo
                 quita(tope);                   // Quita ventana gráfica 68
                 break;
    }
  }
quita(tope);                                    // Quita ventana gráfica 63
}


/*    Escribe en disco un archivo en modo binario ó un archivo en modo texto, según la variable
   tipo. Sigue la trayectoria ya establecida.*/
void escribir(struct MATERIA *p, char semestre[10], int tipo)
{
 struct MATERIA *r;
 struct ARREGLO *arreglo;
 char    nombre[30];
 int ultimo,k;

 pon(tope,100,COPY_PUT);                        // Pone ventana gráfica 100
 texto("ESCRIBIENDO UNA LISTA DE MATERIAS",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
if ((*p).num == 0) then                         // Si no se ha leído algún archivo se
  {                                             // envía mensaje de error
  llama_error(4);
  quita(tope);                                  // Quita ventana gráfica 100
  return;
  }
 if (tipo == 2) then
  {
   llama_ayuda(57);                             // Llama a la función AYUDA
   pon(tope,352,COPY_PUT);                      // Confirma la salida de un archivo en modo texto
   texto("Confirme la salida en código ASCII . . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
   pon(tope,253,COPY_PUT);
   pon(tope,254,COPY_PUT);
```

```
    texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
    texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
    pon(tope,253,NOT_PUT);
    k = opciones1(253,253,254,9);
    quita(tope);                                        // Quita cursor
    quita(tope);                                        // Quita ventana gráfica 254
    quita(tope);                                        // Quita ventana gráfica 253
    quita(tope);                                        // Quita ventana gráfica 352
    if (k == 253 || k == -1) then
          tipo = 0;
  }
 else
   texto("Escriba el archivo con extensión .TP  . . .",115,RED,SANS_SERIF_FONT,1,2,1,1);
 if (tipo != 0) then
  {
    pon(tope,101,COPY_PUT);                    // Pone ventana gráfica 101
    pon(tope,102,COPY_PUT);                    // Pone ventana gráfica 102
    texto("Nombre del Archivo . . .",101,RED,SANS_SERIF_FONT,1,2,1,1);
    lee_cadena(nombre,24,102,RED,SANS_SERIF_FONT,1,3,1,1);  // Lee el nombre del archivo para almacenarlo en disco
    texto(nombre,102,RED,SANS_SERIF_FONT,1,2,1,1);
    quita(tope);                               // Quita ventana gráfica 102
    quita(tope);                               // Quita ventana gráfica 101
    r = p;                                     // Ordena la lista enlazada MATERIA por orden alfabético
    ultimo = 0;
    while (r != NULL) do
    {
          ultimo = ultimo + 1;
          r = (*r).materias;
    }
    arreglo = (struct ARREGLO *) farcalloc(ultimo+1,sizeof(struct ARREGLO));
          if (!arreglo)
            {
             printf("Error.");
             exit(1);
            }
    r = p;
    k = 0;
    while (r != NULL) do
          {
           arreglo[k].numero = (*r).clave;
           arreglo[k].grupo = (*r).clave;
           strcpy(arreglo[k].nombre,(*r).nombre);
           arreglo[k].materia = (*r).num;
           k = k + 1;
           r = (*r).materias;
          }
    arreglo[k].numero = -2;
    arreglo[k].grupo = -2;
    strcpy(arreglo[k].nombre,"-2");
    arreglo[k].materia = -2;
    ultimo = ultimo - 1;
    if (ultimo == -1) then
          ultimo = 0;
    ordena_nombre_arreglo(arreglo,ultimo);
    ordena_materias_clave(p,arreglo);
    farfree(arreglo);
  }
switch(tipo)                                   // Escribe la lista enlazada MATERIA en disco
  {
        case 1: guarda_lista_materias(p,nombre,semestre);
                  break;
        case 2: guarda_lista_materias_texto(p,nombre);
                  break;
  }
                                   // Envía mensaje que avisa presionar una tecla para continuar
```

```
pon(tope,118,COPY_PUT);                          // Pone ventana gráfica 115
texto("Presiona un tecla para continuar . . .",115,RED,SANS_SERIF_FONT,1,2,1,1);
bioskey(0);
quita(tope);                                     // Quita ventana gráfica 115
quita(tope);                                     // Quita ventana gráfica 100
}

//    Lee los campos de un nodo de la lista enlazada MATERIA desde el teclado
void lee_nodo_materia(struct MATERIA *p, struct MATERIA *s)
{
 char     nombre[100], cadena[100];
 int      clave, i, laboratorio, tipomat, valor;
                                                 // Lee el NOMBRE DE LA MATERIA
 pon(tope,101,COPY_PUT);                         // Pone ventana gráfica 101
 pon(tope,102,COPY_PUT);                         // Pone ventana gráfica 102
 texto("Nombre de la Materia . . .",101,RED,SANS_SERIF_FONT,1,2,1,1);
 valor = 1;
 while (valor == 1) do                           // Verifica que el nuevo nombre exista, si
 {                                               // existe envía un mensaje de error
  lee_cadena(nombre,28,102,RED,SANS_SERIF_FONT,1,3,1,1);
  valor = error_nombre_igual(s,nombre);
 }
 texto(nombre,102,RED,SANS_SERIF_FONT,1,2,1,1);
                                                 // Lee la CLAVE de la materia
 pon(tope,103,COPY_PUT);                         // Pone la ventana gráfica 103
 pon(tope,104,COPY_PUT);                         // Pone la ventana gráfica 104
 texto("Clave de la Materia . . .",103,RED,SANS_SERIF_FONT,1,2,1,1);
 valor = 1;
 while (valor == 1) do                           // Verifica que la nueva clave exista, si
 {                                               // existe envía un mensaje de error
  lee_cadena(cadena,4,104,RED,SANS_SERIF_FONT,1,2,1,1);
  valor = error_clave_igual(s,cadena);
 }
 texto(cadena,104,RED,SANS_SERIF_FONT,1,1,1,1);
 clave = atoi(cadena);
 laboratorio = 3;                                // Lee la opción de LABORATORIO
 pon(tope,105,COPY_PUT);                         // Pone ventana gráfica 105
 texto("Laboratorio . . .",105,RED,SANS_SERIF_FONT,1,2,1,1);
 pon(tope,107,COPY_PUT);                         // Pone ventana gráfica 107
 pon(tope,108,COPY_PUT);                         // Pone ventana gráfica 108
 pon(tope,109,COPY_PUT);                         // Pone ventana gráfica 109
 texto("Lopt",107,BLUE,SMALL_FONT,2,1,2,1);
 texto("L",108,BLUE,SMALL_FONT,2,1,2,1);
 texto("S/Lab",109,BLUE,SMALL_FONT,2,1,2,1);
 pon(tope,107,NOT_PUT);                          // Pone ventana gráfica del cursor
 i = opciones1(107,107,109,10);                  // Mueve la pila de ventanas
 quita(tope);                                    // Quita ventana gráfica del cursor
 quita(tope);                                    // Quita ventana gráfica 109
 quita(tope);                                    // Quita ventana gráfica 108
 quita(tope);                                    // Quita ventana gráfica 107
 pon(tope,106,COPY_PUT);                         // Pone ventana gráfica 106
 switch(i)
   {
   case 107: texto("Laboratorio Opcional",106,RED,SANS_SERIF_FONT,2,3,1,1);
             laboratorio = 1;
             break;
   case 108: texto("Laboratorio Obligatorio",106,RED,SANS_SERIF_FONT,2,3,1,1);
             laboratorio = 2;
             break;
   case 109: texto("Sin Laboratorio",106,RED,SANS_SERIF_FONT,2,3,1,1);
             laboratorio = 3;
             break;
   }                                             // Opción TIPO MATERIA
 tipomat = 1;
 pon(tope,110,COPY_PUT);                         // Pone ventana gráfica 110
```

```
texto("Tipo de Materia . . .",110,RED,SANS_SERIF_FONT,1,2,1,1);
pon(tope,111,COPY_PUT);                                  // Pone ventana gráfica 111
pon(tope,112,COPY_PUT);                                  // Pone ventana gráfica 112
texto("Oblig.",111,BLUE,SMALL_FONT,2,1,2,1);
texto("Opta.",112,BLUE,SMALL_FONT,2,1,2,1);
pon(tope,111,NOT_PUT);                                   // Pone ventana gráfica del cursor
i = opciones1(111,111,112,11);                           // Mueve pila de ventanas
quita(tope);                                             // Quita ventana gráfica del cursor
quita(tope);                                             // Quita ventana gráfica 112
quita(tope);                                             // Quita ventana gráfica 111
pon(tope,113,COPY_PUT);                                  // Pone ventana gráfica 113
switch(i)
  {
  case 111: texto("Materia Obligatoria",113,RED,SANS_SERIF_FONT,2,3,1,1);
            tipomat = 1;
            break;
  case 112: texto("Materia Optativa",113,RED,SANS_SERIF_FONT,2,3,1,1);
            tipomat = 2;
            break;
  }
        // Envía el mensaje de presionar una tecla para continuar
pon(tope,115,COPY_PUT);
texto("Presiona una tecla para continuar . . .",115,RED,SANS_SERIF_FONT,1,2,1,1);
bioskey(0);
quita(tope);                    // Quita ventana gráfica 115
quita(tope);                    // Quita ventana gráfica 113
quita(tope);                    // Quita ventana gráfica 110
quita(tope);                    // Quita ventana gráfica 106
quita(tope);                    // Quita ventana gráfica 105
quita(tope);                    // Quita ventana gráfica 104
quita(tope);                    // Quita ventana gráfica 103
quita(tope);                    // Quita ventana gráfica 102
quita(tope);                    // Quita ventana gráfica 101
                                // Asigna campos al nodo *p
for (i = 0; nombre[i]; i++)
  nombre[i] = toupper(nombre[i]);
strcpy((*p).nombre,nombre);
(*p).clave = clave;
(*p).lab = laboratorio;
(*p).tipomat = tipomat;
(*p).grupos = NULL;
(*p).examen = NULL;
}


//    Agregar un nuevo nodo a la lista enlazada MATERIA
int menu_agregar(void)
{
 int c,j;

pon(tope,253,COPY_PUT);                                  // Pone ventana gráfica 253
texto("AGREGAR UNO MAS",253,RED,SANS_SERIF_FONT,1,2,1,2);
pon(tope,254,COPY_PUT);                                  // Pone ventana gráfica 254
texto("SALIR",254,RED,SANS_SERIF_FONT,1,2,1,2);
c=0;
j=-1;
pon(tope,253,NOT_PUT);                                   // Pone ventana gráfica del cursor
while (c == 0)
{
  while (j == -1) do
  j=opciones1(253,253,254,12);
  switch (j)
    {
        case  253 : c = 1;
                    break;
```

```
            case  254 : c = 2;
                        break;
        }
    }
    quita(tope);                                    // Quita ventana gráfica del cursor
    quita(tope);                                    // Pone ventana gráfica 253
    quita(tope);                                    // Pone ventana gráfica 254
    return(c);
}

//      Pone menú de opciones
void ventana_nodo_lista_materia(void)
{
    pon(tope,150,COPY_PUT);
    cuadro(150,BLUE);
    texto("SALIR",152,LIGHTRED,SMALL_FONT,2,1,2,1);
    texto("<F1> AYUDA",153,LIGHTRED,SMALL_FONT,2,1,2,1);
    texto("<TAB> CAMBIA MENU",154,LIGHTRED,SMALL_FONT,3,2,2,1);
}

//      Pone cada campo de cada nodo de la lista enlazada MATERIA. Devuelve el número de la
    última ventana.
int ventanas_ponen_materias(struct MATERIA *p, int inicio, int fin, int *max)
{
    struct MATERIA *r;
    int i,c,k,j;
    char numero[20];

    r=p;
    i = 130;
    while ((*r).num != inicio) do                   // Mueve el apuntador r hasta el nodo
            r=(*r).materias;                        // definido por inicio.
j=*max;
for (k=inicio; k <= fin; k++)                       // Despliega a partir de inicio hasta fin
    {
            cuadro(i,CYAN);
            pon(tope,i,COPY_PUT);
            itoa((*r).num,numero,10);
            texto (numero,i,RED,SMALL_FONT,1,1,2,1);
            i=i+1;
            pon(tope,i,COPY_PUT);
            itoa((*r).clave,numero,10);
            texto (numero,i,RED,SMALL_FONT,1,1,2,1);
            i=i+1;
            pon(tope,i,COPY_PUT);
            texto ((*r).nombre,i,RED,SMALL_FONT,1,1,2,1);
            i=i+1;
            pon(tope,i,COPY_PUT);
            c = (*r).lab;
            switch(c)
             {
                    case 1: texto ("Lopt",i,RED,SMALL_FONT,1,1,2,1);
                            break;
                    case 2: texto ("L",i,RED,SMALL_FONT,1,1,2,1);
                            break;
                    case 3: texto ("S/Lab",i,RED,SMALL_FONT,1,1,2,1);
                            break;
             }
        i=i+1;
        *max=j;
        j=j+1;
        r = (*r).materias;
    }
    return(i);
}
```

```
/*    Permite el desplazamiento del cursor sobre las ventanas de los datos de la lista enlazada MATERIA */
int mover_ventana_menu(int i, int min, int max, int regresa)
{
 int c;

 c=ABAJO;
 pon(tope,i,NOT_PUT);
 while (c != ENTER) then
 {
  c=biosKey(0);
  switch(c)
  {
        case DERECHA:   quita(tope);          // Mueve la pila de ventanas en orden creciente
                        i++;
                        if (i == (max + 1)) then
                          i = min;
                        pon(tope,i,NOT_PUT);
                        break;
        case IZQUIERDA: quita(tope);                  // Mueve la pila de ventanas en orden decreciente
                        i--;
                        if (i == (min - 1)) then
                          i = max;
                        pon(tope,i,NOT_PUT);
                        break;
        case TAB:       if ((i >= min) && (i <= max)) then  // Cambia al submenú inferior
                        {
                         i=regresa;
                         c=ENTER;
                        }
                        break;
  }
 }
 quita(tope);
 return(i);
}


//    Pone ventanas para menú de opciones
void ventana_cambia_datos_materia(void)
{
 pon(tope,150,COPY_PUT);
 cuadro(150,BLUE);
 cuadrito(152,LIGHTRED,SMALL_FONT,2,1,2,1,"SALIR");
 cuadrito(153,LIGHTRED,SMALL_FONT,2,1,2,1,"<F1> AYUDA");
 cuadrito(154,LIGHTRED,SMALL_FONT,3,2,2,1,"<TAB> CAMBIA MENU");
}


/*   Cambia la información de la lista enlazada MATERIA por los datos introducidos desde el
     teclado.*/
void cambia_datos_materia(struct MATERIA *p, int dato)
{
 struct MATERIA *r;
 char nombre[100],numero[100];
 int clave,tipo,lab,j,j,c,regresa,k;

 if (dato == 0) then
  {
   llama_error(10);
   return;
  }
 r = p;
 while ((*r).num != dato) do  // Encuentra el nodo que ha de ser modificado
          r = (*r).materias;
                              // Menú de opciones de los datos de materia que van a modificarse
```

```
ventana_cambia_datos_materia();
texto("Seleccione el módulo que desee cambiar . . .",210,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("MATERIA",215,LIGHTRED,SMALL_FONT,1,1,2,1);
texto("CLAVE",216,LIGHTRED,SMALL_FONT,1,1,2,1);
texto("LABORATORIO",217,LIGHTRED,SMALL_FONT,1,1,2,1);
texto("TIPO",218,LIGHTRED,SMALL_FONT,1,1,2,1);
i = 215;
j = 1;
pon(tope,i,NOT_PUT);                                    // Pone ventana gráfica del cursor
while (j == 1) do
 {
  c = bioskey(0);
  switch(c)
   {
        case F1     : llama_ayuda(15);                  // Llama a la función de AYUDA
                      break;
        case DERECHA : quita(tope);          // Mueve la pila de ventanas en orden creciente
                       i++;
                       if (i == 219) then
                          i = 215;
                       pon(tope,i,NOT_PUT);
                       break;
        case IZQUIERDA: quita(tope);      // Mueve la pila de ventanas en orden decreciente
                       i--;
                       if (i == 214) then
                          i = 218;
                       pon(tope,i,NOT_PUT);
                       break;
        case TAB:     if ((i >= 215) && (i <= 218)) then   // Cambia al submenú inferior
                        {
                         regresa = i;
                         i = 152;
                         i = mover_ventana_menu(i,i,152,regresa);
                         if (i == 152) then
                               j = 0;
                        }
                       break;
        case ENTER:    quita(tope);                     // Quita ventana gráfica del cursor
                       switch(i)
                       {//Despliega en pantalla el campo actual NOMBRE DE LA MATERIA
                         case 215:
                                  pon(tope,171,COPY_PUT);          // Pone ventana gráfica 171
                                  pon(tope,172,COPY_PUT);  // Pone ventana gráfica 172
                                  texto("Nombre de la materia actual ...",171,BLUE,SMALL_FONT,1,1,2,1);
                                  texto((*r).nombre,172,BLUE,SMALL_FONT,1,1,2,1);
                              // Lee del teclado el nuevo campo NOMBRE DE LA MATERIA
                                  pon(tope,173,COPY_PUT);  // Pone ventana gráfica 173
                                  pon(tope,174,COPY_PUT);  // Pone ventana gráfica 174
                                  texto("Nuevo nombre de la materia ...",173,RED,SMALL_FONT,1,1,2,1);
                                  k = 1;
                                  while (k == 1) do
                                       {
                                        lee_cadena(nombre,28,174,RED,SMALL_FONT,1,1,2,1);
                                        k = error_nombre_cambiar(p,nombre,dato);
                                       }
                                  texto(nombre,174,RED,SMALL_FONT,1,1,2,1);
                                  pon(tope,179,COPY_PUT);   // Pone ventana gráfica 179
                                  texto("Presione una tecla para continuar ...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                  bioskey(0);
                                  quita(tope);      // Quita ventana gráfica 179
                                  quita(tope);      // Quita ventana gráfica 174
                                  quita(tope);      // Quita ventana gráfica 173
                                  quita(tope);      // Quita ventana gráfica 172
                                  quita(tope);      // Quita ventana gráfica 171
                                  strcpy((*r).nombre,nombre);
```

```
                i = 215;
                pon(tope,i,NOT_PUT); // Pone ventana gráfica del cursor
                break;

        case 216: // Despliega en pantalla el campo actual CLAVE
                pon(tope,171,COPY_PUT);  // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);  // Pone ventana gráfica 172
                texto("Clave de la materia actual ...",171,BLUE,SMALL_FONT,1,1,2,1);
                itoa((*r).clave,numero,10);
                texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee del teclado el nuevo campo CLAVE
                pon(tope,173,COPY_PUT);  // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);  // Pone ventana gráfica 174
                texto("Nueva clave de la materia ...",173,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) do
                 {
                      lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                      k = error_clave_cambiar(p,numero,dato);
                 }
                clave=atoi(numero);
                pon(tope,179,COPY_PUT);          // Pone ventana gráfica 179
                texto("Presione una tecla para continuar ...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);     // Quita ventana gráfica 179
                quita(tope);     // Quita ventana gráfica 174
                quita(tope);     // Quita ventana gráfica 173
                quita(tope);     // Quita ventana gráfica 172
                quita(tope);     // Quita ventana gráfica 171
                (*r).clave = clave;
                i = 216;
                pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                break;

        case 217: // Despliega en pantalla el campo actual OPCION
                //LABORATORIO
                pon(tope,171,COPY_PUT);
                pon(tope,172,COPY_PUT);
                texto("Opcion de laboratorio actual ...",171,BLUE,SMALL_FONT,1,1,2,1);
                c = (*r).lab;
                switch(c)
                     {
                      case 1: texto("Laboratorio Opcional",172,BLUE,SMALL_FONT,1,1,2,1);
                              break;
                      case 2: texto("Laboratorio Obligatorio",172,BLUE,SMALL_FONT,1,1,2,1);
                              break;
                      case 3: texto("Sin Laboratorio",172,BLUE,SMALL_FONT,1,1,2,1);
                              break;
                     }
              // Lee del teclado el nuevo campo OPCION LABORATORIO
                pon(tope,173,COPY_PUT); // Pone ventana gráfica 173
                pon(tope,176,COPY_PUT); // Pone ventana gráfica 176
                pon(tope,177,COPY_PUT); // Pone ventana gráfica 177
                pon(tope,178,COPY_PUT); // Pone ventana gráfica 178
                texto("Nueva opcion de laboratorio ...",173,RED,SMALL_FONT,1,1,2,1);
                texto("Lapt",176,RED,SMALL_FONT,1,1,2,1);
                texto("L",177,RED,SMALL_FONT,1,1,2,1);
                texto("S/Lab",178,RED,SMALL_FONT,1,1,2,1);
                pon(tope,176,NOT_PUT);  // Pone ventana gráfica del cursor
                i = opciones1(176,176,178,7); // Mueve la pila de ventanas
                quita(tope);     // Quita ventana gráfica del cursor
                quita(tope);     // Quita ventana gráfica 178
                quita(tope);     // Quita ventana gráfica 177
                quita(tope);     // Quita ventana gráfica 176
                pon(tope,174,COPY_PUT); // Pone ventana gráfica 174
```

```
                switch(i)
                    {
                     case 176: texto("Laboratorio Optativo",174,RED,SMALL_FONT,1,1,2,1);
                               lab = 1;
                               break;
                     case 177: texto("Laboratorio Obligatorio",174,RED,SMALL_FONT,1,1,2,1);
                               lab = 2;
                               break;
                     case 178: texto("Sin Laboratorio",174,RED,SMALL_FONT,1,1,2,1);
                               lab = 3;
                               break;
                    }
                pon(tope,179,COPY_PUT);  // Pone ventana gráfica 179
                texto("Presione una tecla para continuar ...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);     // Quita ventana gráfica 179
                quita(tope);     // Quita ventana gráfica 174
                quita(tope);     // Quita ventana gráfica 173
                quita(tope);     // Quita ventana gráfica 172
                quita(tope);     // Quita ventana gráfica 171
                (*r).lab = lab;
                l = 217;
                pon(tope,l,NOT_PUT); // Quita ventana gráfica del cursor
                break;

        case 216: // Despliega en pantalla campo actual TIPO DE MATERIA
                pon(tope,171,COPY_PUT);  // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);  // Pone ventana gráfica 172
                texto("Tipo de materia actual ...",171,BLUE , SMALL_FONT,1,1,2,1);
                c = (*r).tipomat;
                switch(c)
                    {
                     case 1: texto("Materia Obligatoria",172,BLUE,SMALL_FONT,1,1,2,1);
                             break;
                     case 2: texto("Materia Optativa",172,BLUE,SMALL_FONT,1,1,2,1);
                             break;
                    }
                // Lee del teclado el nuevo campo TIPO DE MATERIA
                pon(tope,173,COPY_PUT); // Pone ventana gráfica 173
                pon(tope,193,COPY_PUT); // Pone ventana gráfica 193
                pon(tope,194,COPY_PUT); // Pone ventana gráfica 194
                texto("Nuevo tipo de materia ...",173,RED,   SMALL_FONT,1,1,2,1);
                texto("Materia Obligatoria",193,RED,SMALL_FONT,1,1,2,1);
                texto("Materia Optativa",194,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,NOT_PUT);  // Pone ventana gráfica del cursor
                i = opciones1(193,193,194,7); // Mueve la pila de ventanas
                quita(tope);     // Quita ventana gráfica del cursor
                quita(tope);     // Quita ventana gráfica 194
                quita(tope);     // Quita ventana gráfica 193
                pon(tope,174,COPY_PUT);
                switch(i)
                    {
                     case 193: texto("Materia Obligatoria",174,RED,SMALL_FONT,1,1,2,1);
                               tipo = 1;
                               break;
                     case 194: texto("Materia Optativa",174,RED,SMALL_FONT,1,1,2,1);
                               tipo = 2;
                               break;
                    }
                pon(tope,179,COPY_PUT); // Pone ventana gráfica 179
                texto("Presione una tecla para continuar ...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);     // Quita ventana gráfica 179
                quita(tope);     // Quita ventana gráfica 174
                quita(tope);     // Quita ventana gráfica 173
```

```
                                        quita(tope);     // Quita ventana gráfica 172
                                        quita(tope);     // Quita ventana gráfica 171
                                        (*r).tipomat = tipo;
                                        i = 218;
                                        pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                                        break;
                                }
     }
   }
 quita(tope);                                    // Quita ventana gráfica del cursor
 k = 210;                                                // Quita las ventanas gráficas restantes
 setfillstyle(SOLID_FILL,GREEN);
 bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 for (k=215; k<=218; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 for (k=152; k<=154; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 quita(tope);                            // Quita ventana gráfica 150
}


// Crea una lista de MATERIAS. Lee el primer nodo de la lista de materias
void crear(struct MATERIA *p)
{
 struct MATERIA r,*s;
 int k;

 pon(tope,100,COPY_PUT);                                 // Pone ventana gráfica 100
 texto("CREANDO UNA NUEVA LISTA DE MATERIAS",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 k = 1;
 if ((*p).num > 0) then
   k = confirmar(1);
                                        // Asignar al primer nodo en la lista de materias
 if (k == 1) then
  {
   lee_nodo_materia(&r,s);
   crea_lista_materias(p);
   pon_lista_materias(p,r);
  }
 quita(tope);                            // Quita ventana gráfica 100
}


//   Añade un nuevo nodo a la lista enlazada MATERIA
void agregar(struct MATERIA *p)
{
 struct MATERIA r,*s;
 int d;

 pon(tope,100,COPY_PUT);
 texto("AGREGANDO UNA NUEVA MATERIA A LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
  {
   llama_error(5);
   quita(tope); // 100
   return;
  }
                                                // Asignar al primer nodo en la lista de materias
 s = p;
```

```
d=1;
while (d != 2)
{
 lee_nodo_materia(&r,e);                    // Lee del teclado los campos de MATERIA
 pon_lista_materias(p,r);                   // Añade un nodo en la lista enlazada MATERIA
 d=menu_agregar();                          // Opción para añadir más nodos
}
quita(tope);                                // Quita ventana gráfica 100
}


//    Elimina físicamente un nodo de la lista enlazada MATERIA
void eliminar(struct MATERIA *p)
{
struct MATERIA *r,*s;
int dato,i,ultimo,valor;

pon(tope,100,COPY_PUT);                     // Pon ventana gráfica 100
texto("ELIMINANDO UNA NUEVA MATERIA A LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
if ((*p).num == 0) then
 {
 llama_error(6);
 quita(tope);                               // Quita ventana gráfica 100
 return;
 }
i = 0;
dato = 0;
while (i != -1) do
  {
   r = p;
   dato = escoje_consulta_materia(r,&i);    // Selecciona el nodo a eliminar
   if (i != -1) then
    {
        if (i != 351) then
          {
           valor = confirmar(2);

           if (valor == 1) then
            {
                  dato=quita_lista_materias(p,dato);  // Elimina si y sólo si hay más de uno nodo

             if (dato == 0) then
                  {
                   crea_lista_materias(p); // Crea una nueva lista cuando se elimina el último nodo
                   i = -1;
                  }
            }
          }
        else
         {
          s = p;
          ultimo = 0;
          while (s != NULL) do
           {
                 ultimo = ultimo + 1;
                 s = (*s).materias;
           }
          while (dato != -2) do
          {
           r = p;   // Despliega por ventanas la información de cada nodo de MATERIA
           dato=mover_pantallas_consulta(120,120,231,p,ultimo);
           if (dato > 0) then
                  {
                   valor = confirmar(2);
                   if (valor == 1) then
```

```
                    {                   // Elimina si y sólo si hay más de uno nodo
                     dato=quita_lista_materias(p,dato);
                     if (dato == 0) then
                      {                 // Crea una nueva lista cuando se elimina el último nodo
                          crea_lista_materias(p);
                          i = -1;
                          dato = -2;
                      }
                    }
                  }
                }
              }
            }
          }
        quita(tope);                                        // Quita ventana gráfica 100
}


/*  Llama a la función que ha de leer los nuevos datos para cada nodo de la lista enlazada
   MATERIA */
void cambiar(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct ARREGLO *arreglo;
 int dato,valor,j,clave,k,ultimo;
 char cadena[100];

 pon(tope,100,COPY_PUT);                                    // Pon ventana gráfica 100
 texto("CAMBIANDO UNA NUEVA MATERIA A LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
  {
   llama_error(7);
   quita(tope);                                             // Quita ventana gráfica 100
   return;
  }
  r = p;
  i = 0;
  dato = 0;
  while (i != -1) do
    {
     dato = escoje_consulta_materia(r,&i);                  // Selecciona el nodo a eliminar
     if (i != -1) then
      {
          if (i != 351) then
              cambia_datos_materia(r,dato);                 // Cambia los datos actuales por los nuevos datos
          else
            {
             s = p;
             ultimo = 0;
             while (s != NULL) do
              {
                    ultimo = ultimo + 1;
                    s = (*s).materias;
              }
             while (dato != -2) do
             {      // Despliega por ventanas la información de cada nodo de MATERIA
              dato=mover_pantallas_consulta(120,120,231,p,ultimo);
              if (dato > 0) then                    // Cambia los datos actuales por los nuevos datos
                  cambia_datos_materia(r,dato);
             }
            }
      }
    }
  }
 quita(tope);                                   // Quita ventana gráfica 100
                                                // Ordena por orden alfabético la lista enlazada MATERIA
```

```
  r = p;
  ultimo = 0;
  while (r != NULL) do
   {
    ultimo = ultimo + 1;
    r = (*r).materias;
   }
  arreglo = (struct ARREGLO *) farcalloc(ultimo+1,sizeof(struct ARREGLO));
   if (!arreglo)
     {
      printf("Error.");
      exit(1);
     }
  r = p;
  k = 0;
  while (r != NULL) do
   {
     arreglo[k].numero = (*r).clave;
     arreglo[k].grupo = (*r).clave;
     strcpy(arreglo[k].nombre,(*r).nombre);
     arreglo[k].materia = (*r).num;
     k = k + 1;
     r = (*r).materias;
   }
  arreglo[k].numero = -2;
  arreglo[k].grupo = -2;
  strcpy(arreglo[k].nombre,"-2");
  arreglo[k].materia = -2;
  ultimo = ultimo - 1;
  if (ultimo == -1) then
    ultimo = 0;
  ordena_nombre_arreglo(arreglo,ultimo);
  ordena_materias_clave(p,arreglo);
  farfree(arreglo);
}

// Hace la llamada a las funciones según el valor de i
void archivos(int i, struct MATERIA *lista, char semestre[10])
{

 cuadro(61,GREEN);
 switch(i)
   {
    case 31: leer(lista,semestre);          // Lee archivos en disco y establece directorios
             break;
    case 32: escribir(lista,semestre,1);   // Guarda archivos en disco según la trayectoria establecida
             break;
    case 33: crear(lista);                 // Crea una nueva lista enlazada MATERIA
             break;
    case 34: agregar(lista);               // Agrega nodos a la lista enlazada MATERIA
             break;
    case 35: eliminar(lista);              // Elimina nodos de la lista enlazada MATERIA
             break;
    case 36: cambiar(lista);               // Modifica la información de la lista enlazada MATERIA
             break;
   }
 cuadro(61,LIGHTBLUE);
}
```

B.3.2. AYUDA.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\errores.h"
#define  then



// Centra un texto dentro de una ventana
void cuadro_texto(char *p, int n, int color, int tipo, int x1, int x2, int y1, int y2)
{
int       x, y, i, j;

setfillstyle(SOLID_FILL,color);
bar(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);

settextstyle(tipo,HORIZ_DIR,USER_CHAR_SIZE);
setusercharsize(x1,x2,y1,y2);

y = (unsigned int)(1.2*textheight(p));
x = textwidth(p);

i = vent[n].x1 + div(vent[n].x2 - vent[n].x1 - x,2).quot;
j = vent[n].y1 + div(vent[n].y2 - vent[n].y1 - y,2).quot;

outtextxy(i,j,p);
}



/*    Crea la ventana para el módulo de ayuda. Centra el texto del arreglo cadena según las di -
  mensiones definidas por:  x1, y1, x2, y2.  */
void ventana_ayuda(char cadena[20][100], int inicio, int fin)
{
int x1, y1, x2, y2, x, y, i, j, m, k, n, mul;

x1 = 1;
x2 = 1;
y1 = 3;
y2 = 2;
n = 379;
setfillstyle(SOLID_FILL,LIGHTBLUE);
bar(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);

setcolor(WHITE);
rectangle(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);

settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
setusercharsize(x1,x2,y1,y2);
```

```
m = div(vent[n].y2 - vent[n].y1,8).quot;          // Se divide el tamaño de la ventana en ocho,
                                                  // para tener un máximo de ocho líneas de texto

for (k=inicio, mul=0;  k <= fin;  k++, mul++)
  {
   y = (unsigned int)(1.2*textheight(cadena[k]));
   x = textwidth(cadena[k]);
   i = vent[n].x1 + div(vent[n].x2 - vent[n].x1 - x,2).quot;
   j = (vent[n].y1 + div(m - y,2).quot) + m*mul;
   outtextxy(i,j,cadena[k]);
  }
}



/*   Llama a la función que establece el número de líneas dentro de la ventana de ayuda. Lee
  el archivo AYUDA y selecciona el texto que ha de ser desplegado en pantalla.*/
void llama_ayuda(int num)
{
 FILE *archivo;
 char cadena[50][100], cad[100], sig[100];
 int c,quot,rem,ultimo,j,grupo,inicio,fin,k,a;
 struct POS
      {
          int inicio;
          int fin;
      }pos[50];

 strcpy(cad,"AYUDA\\h");
 itoa(num,sig,10);
 strcat(cad,sig);
 if ((archivo = fopen(cad,"rb")) == NULL) then    // Abre el archivo definido por num
   {
    llama_error(53);
    return;
   }

 while (!feof(archivo))
   {
    if (!fread(cadena,4000,1,archivo)) then
      break;
   }
 ultimo = 0;
 for (k=0; strcmp(cadena[k],""); k++)
   ultimo = ultimo + 1;

 quot = ultimo/8;
 rem = ultimo%8;
 if (rem != 0) then
   grupo = quot + 1;
 else
   grupo = quot;
 k = 0;

 for (j=1; j <= grupo; j++)
   {
    a = 0;
    while ((strcmp(cadena[k],"")) && (a < 8)) do
      {
           fin = k;
           if (a == 0) then
             inicio = fin;
           a = a + 1;
           k = k + 1;
      }
```

```
    pos[].inicio = inicio;
    pos[].fin = fin;
   }
 j = 1;
 inicio = pos[].inicio;
 fin = pos[].fin;
 pon(tope,378,COPY_PUT);                          // Despliega la ventana del módulo AYUDA
 cuadro_texto(" ",376,MAGENTA,SMALL_FONT,1,1,1,1);
 setfillstyle(SOLID_FILL,MAGENTA);
 rectangle(102,182,536,378);
 rectangle(104,184,536,376);
 cuadro_texto(" MENSAJE ",377,MAGENTA,COMPLEX_FONT,1,2,1,2);
 ventana_ayuda(cadena,inicio,fin);
 cuadro_texto("Presione <ESC> para salir . . .",378,DARKGRAY,TRIPLEX_SCR_FONT,1,2,1,2);
 if (grupo > 1) then
  {
   strcpy(cad," PAG. 1 ");
   cuadro_texto(cad,360,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
   cuadro_texto("PgUp/PgDn",361,GREEN,SIMPLEX_FONT,2,5,1,2);
   c = END;
   while (c != ESC) do
        {
         c = bioskey(0);
         switch(c)
          {
                case PGDN: j++;                          // Cambia a la siguiente página
                        if (j == (grupo + 1)) then
                           j = 1;
                        inicio = pos[j].inicio;
                        fin = pos[j].fin;
                        ventana_ayuda(cadena,inicio,fin);
                        itoa(j,sig,10);
                        strcpy(cad,"PAG. ");
                        strcat(cad,sig);
                        cuadro_texto(cad,360,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
                        break;

                case PGUP: j--;                                  // Cambia a la página anterior
                        if (j == 0) then
                           j = grupo;
                        inicio = pos[j].inicio;
                        fin = pos[j].fin;
                        ventana_ayuda(cadena,inicio,fin);
                        itoa(j,sig,10);
                        strcpy(cad,"PAG. ");
                        strcat(cad,sig);
                        cuadro_texto(cad,360,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
                        break;
          }
        }
  }
 else
  {

   c = END;
   while (c != ESC) do
        c = bioskey(0);
  }

 fclose(archivo);                                        // Cierra el archivo
 quita(tope);                                            // Quita ventana gráfica 376
}
```

B.3.3. CONSULTA.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\grupos.h"
#include "d:\consulta.h"
#include "d:\g_todos.h"
#include "d:\p_todos.h"
#include "d:\e_todos.h"
#include "d:\imprime.h"
#include "d:\errores.h"
#include "d:\ayuda.h"
#define  then


/*   Estructura de la pila para ordenar elementos construyendo el algoritmo Quick Sort. */
 struct ELEMENTO
   {
    int inferior;
    int superior;
   };

 struct STACK
   {
    int top;
    struct ELEMENTO *elemento;
   };


//   Pone elementos en la pila
void pon_stack(struct STACK *ps,struct ELEMENTO el, int N)
{
 if ((*ps).top == N) then
   exit(1);
 (*ps).top = (*ps).top + 1;
 (*ps).elemento[(*ps).top].inferior = el.inferior;
 (*ps).elemento[(*ps).top].superior = el.superior;
}


//   Quita elementos de la pila
void quita_stack(struct STACK *ps,struct ELEMENTO *el)
{
 if ((*ps).top == 0) then
   exit(1);
 (*el) = (*ps).elemento[(*ps).top];
 (*ps).top--;
}
```

```
/*    Ordena por número de grupo el arreglo ARREGLO. Los parámetros inferior y superior contienen los valores  límite
  de la sublista de ARREGLO a la que se le aplica el procedimiento. j sigue la pista a la posición del primer elemento de
  la sublista durante el procedimiento. Las variables locales izq y der contendrán los valores límite de la lista de elemen-
  tos que no han sido examinados. */
void particion_arreglo(struct ARREGLO *arreglo, long int inferior, long int superior, long int *j)
{
 int grupo, materia;
 long int numero, izq, der, pos;

 izq = inferior;
 der = superior;
 pos = inferior;
 while (izq < der) do
   {
    while ((arreglo[pos].numero <= arreglo[der].numero) && (pos != der)) do
              der = der - 1;
    if (pos == der) then
      break;
    if (arreglo[pos].numero > arreglo[der].numero) then
     {
          grupo = arreglo[pos].grupo;
          materia = arreglo[pos].materia;
          numero = arreglo[pos].numero;
          arreglo[pos].grupo = arreglo[der].grupo;
          arreglo[pos].materia = arreglo[der].materia;
          arreglo[pos].numero = arreglo[der].numero;
          arreglo[der].grupo = grupo;
          arreglo[der].materia = materia;
          arreglo[der].numero = numero;
          pos = der;
     }
    while ((arreglo[izq].numero <= arreglo[pos].numero) && (izq != pos)) do
            izq = izq + 1;
    if (pos == izq) then
      break;
    if (arreglo[izq].numero > arreglo[pos].numero) then
     {
      grupo = arreglo[pos].grupo;
      materia = arreglo[pos].materia;
      numero = arreglo[pos].numero;
      arreglo[pos].grupo = arreglo[izq].grupo;
      arreglo[pos].materia = arreglo[izq].materia;
      arreglo[pos].numero = arreglo[izq].numero;
      arreglo[izq].grupo = grupo;
      arreglo[izq].materia = materia;
      arreglo[izq].numero = numero;
      pos = izq;
     }
   }
 *j = pos;
}

//    Construye el método de Quick Sort. Ordena la lista de elementos completa.
void ordena_arreglo(struct ARREGLO *arreglo, int n)
{
 long int pos, temp;
 struct STACK stack;
 struct ELEMENTO elemen;

 stack.elemento = (ELEMENTO *) farcalloc(n+1,sizeof(ELEMENTO));
 if (!stack.elemento)
   {
    printf("Error.");
    exit(1);
   }
```

```
 pos = 0;
 stack.top = 0;
 elemen = stack.elemento[stack.top];
 if (n >= 1) then
  {
   elemen.inferior = 0;
   elemen.superior = n;
   pon_stack(&stack,elemen,n);
   while (stack.top != 0) do
     {
      quita_stack(&stack,&elemen);
      particion_arreglo(arreglo,elemen.inferior,elemen.superior,&pos);
      if (elemen.inferior < pos - 1) then
            {
             temp = elemen.superior;
             elemen.superior = pos - 1;
             pon_stack(&stack,elemen,n);
             elemen.superior = temp;
            }
      if (pos + 1 < elemen.superior) then
            {
             elemen.inferior = pos + 1;
             pon_stack(&stack,elemen,n);
            }
     }
  }
farfree(stack.elemento);
}


/*     Ordena por número de grupo el arreglo SALIDA. Los parámetros inferior y superior contienen
  los valores límite de la sublista de ARREGLO a la que se le aplica el procedimiento. Sigue la pista
  a la posición del primer elemento de la sublista durante el procedimiento. Las variables locales izq
  y der contendrán los valores límite de la lista de elementos que no han sido  examinados. */
void particion_salida(struct SALIDA *salida, long int inferior, long int superior, long int *j)
{
int grupo, materia;
long int numero, izq, der, pos;

izq = inferior;
der = superior;
pos = inferior;
while (izq < der) do
  {
   while ((salida[pos].numero <= salida[der].numero) && (pos != der)) do
            der = der - 1;
   if (pos == der) then
     break;
   if (salida[pos].numero > salida[der].numero) then
     {
          grupo = salida[pos].grupo;
          materia = salida[pos].materia;
          numero = salida[pos].numero;
          salida[pos].grupo = salida[der].grupo;
          salida[pos].materia = salida[der].materia;
          salida[pos].numero = salida[der].numero;
          salida[der].grupo = grupo;
          salida[der].materia = materia;
          salida[der].numero = numero;
          pos = der;
     }
   while ((salida[izq].numero <= salida[pos].numero) && (izq != pos)) do
            izq = izq + 1;
   if (pos == izq) then
     break;
```

```
    if (salida[izq].numero > salida[pos].numero) then
      {
       grupo = salida[pos].grupo;
       materia = salida[pos].materia;
       numero = salida[pos].numero;
       salida[pos].grupo = salida[izq].grupo;
       salida[pos].materia = salida[izq].materia;
       salida[pos].numero = salida[izq].numero;
       salida[izq].grupo = grupo;
       salida[izq].materia = materia;
       salida[izq].numero = numero;
       pos = izq;
      }
    }
  *j = pos;
}


//     Construye el método de Quick Sort. Ordena la lista de elementos completa.
void ordena_salida(struct SALIDA *salida, int n)
{
 long int pos, temp;
 struct STACK stack;
 struct ELEMENTO elemen;

 stack.elemento = (ELEMENTO *) farcalloc(n+1,sizeof(ELEMENTO));
 if (!stack.elemento)
    {
    printf("Error.");
    exit(1);
    }
 pos = 0;
 stack.top = 0;
 elemen = stack.elemento[stack.top];
 if (n >= 1) then
  {
   elemen.inferior = 0;
   elemen.superior = n;
   pon_stack(&stack,elemen,n);
   while (stack.top != 0) do
     {
      quita_stack(&stack,&elemen);
      particion_salida(salida,elemen.inferior,elemen.superior,&pos);
      if (elemen.inferior < pos - 1) then
             {
              temp = elemen.superior;
              elemen.superior = pos - 1;
              pon_stack(&stack,elemen,n);
              elemen.superior = temp;
             }
      if (pos + 1 < elemen.superior) then
             {
              elemen.inferior = pos + 1;
              pon_stack(&stack,elemen,n);
             }
     }
  }
farfree(stack.elemento);
}


/*      Ordena cadenas para el arreglo ARREGLO. Los parámetros inferior y superior contienen los
  valores límite de la sublista de ARREGLO a la que se le aplica el procedimiento. j sigue la pista a
  la posicion del primer elemento de la sublista durante el procedimiento. Las variables  locales izq
y der contendrán los valores límite de la lista de elementos que no han  sido examinados. */
```

```
void particion_cadena_arreglo(struct ARREGLO *arreglo, long int inferior, long int superior, long int *j)
{
 int grupo,materia;
 long int numero, izq, der, pos;
 char nombre[100];

 izq = inferior;
 der = superior;
 pos = inferior;
 while (izq < der) do
  {
   while ((stromp(arreglo[pos].nombre,arreglo[der].nombre) <= 0) && (pos != der)) do
          der = der - 1;
   if (pos == der) then
     break;
   if (stromp(arreglo[pos].nombre,arreglo[der].nombre) > 0) then
    {
          grupo = arreglo[pos].grupo;
          materia = arreglo[pos].materia;
          numero = arreglo[pos].numero;
          strcpy(nombre,arreglo[pos].nombre);
          arreglo[pos].grupo = arreglo[der].grupo;
          arreglo[pos].materia = arreglo[der].materia;
          arreglo[pos].numero = arreglo[der].numero;
          strcpy(arreglo[pos].nombre,arreglo[der].nombre);
          arreglo[der].grupo = grupo;
          arreglo[der].materia = materia;
          arreglo[der].numero = numero;
          strcpy(arreglo[der].nombre,nombre);
          pos = der;
    }
   while ((stromp(arreglo[izq].nombre,arreglo[pos].nombre) <= 0) && (izq != pos)) do
          izq = izq + 1;
   if (pos == izq) then
     break;
   if (stromp(arreglo[izq].nombre,arreglo[pos].nombre) > 0) then
    {
    grupo = arreglo[pos].grupo;
    materia = arreglo[pos].materia;
    numero = arreglo[pos].numero;
    strcpy(nombre,arreglo[pos].nombre);
    arreglo[pos].grupo = arreglo[izq].grupo;
    arreglo[pos].materia = arreglo[izq].materia;
    arreglo[pos].numero = arreglo[izq].numero;
    strcpy(arreglo[pos].nombre,arreglo[izq].nombre);
    arreglo[izq].grupo = grupo;
    arreglo[izq].materia = materia;
    arreglo[izq].numero = numero;
    strcpy(arreglo[izq].nombre,nombre);
    pos = izq;
    }
  }
 *j = pos;
}

//    Construye el método de Quick Sort. Ordena los elementos de la lista completa.
void ordena_nombre_arreglo(struct ARREGLO *arreglo, int n)
{
 long int pos, temp;
 struct STACK stack;
 struct ELEMENTO elemen;

 stack.elemento = (ELEMENTO *) farcalloc(n+1,sizeof(ELEMENTO));
 if (!stack.elemento)
   {
```

```
    printf("Error.");
    exit(1);
    }
  pos = 0;
  stack.top = 0;
  elemen = stack.elemento[stack.top];
  if (n >= 1) then
   {
    elemen.inferior = 0;
    elemen.superior = n;
    pon_stack(&stack,elemen,n);
    while (stack.top != 0) do
     {
      quita_stack(&stack,&elemen);
      particion_cadena_arreglo(arreglo,elemen.inferior,elemen.superior,&pos);
      if (elemen.inferior < pos - 1) then
            {
             temp = elemen.superior;
             elemen.superior = pos - 1;
             pon_stack(&stack,elemen,n);
             elemen.superior = temp;
            }
      if (pos + 1 < elemen.superior) then
            {
             elemen.inferior = pos + 1;
             pon_stack(&stack,elemen,n);
            }
     }
   }
  farfree(stack.elemento);
  }

  /*    Ordena cadenas para el arreglo SALIDA. Los parámetros inferior y superior contienen los valores límite de la
    sublista de ARREGLO a la que se le aplica el procedimiento. j sigue la pista a la posición del primer elemento de
    la sublista durante el procedimiento. Las variables locales izq y der contendrán los valores límite de la lista de
    elementos que no han sido examinados.*/
  void particion_cadena_salida(struct SALIDA *salida, long int inferior, long int superior, long int *j)
  {
  int grupo,materia;
  long int numero, izq, der, pos;
  char nombre[100];

  izq = inferior;
  der = superior;
  pos = inferior;
  while (izq < der) do
   {
    while ((strcmp(salida[pos].nombre,salida[der].nombre) <= 0) && (pos != der)) do
            der = der - 1;
    if (pos == der) then
      break;
    if (strcmp(salida[pos].nombre,salida[der].nombre) > 0) then
     {
            grupo = salida[pos].grupo;
            materia = salida[pos].materia;
            numero = salida[pos].numero;
            strcpy(nombre,salida[pos].nombre);
            salida[pos].grupo = salida[der].grupo;
            salida[pos].materia = salida[der].materia;
            salida[pos].numero = salida[der].numero;
            strcpy(salida[pos].nombre,salida[der].nombre);
            salida[der].grupo = grupo;
            salida[der].materia = materia;
            salida[der].numero = numero;
            strcpy(salida[der].nombre,nombre);
```

```
            pos = der;
        }
      while ((strcmp(salida[izq].nombre,salida[pos].nombre) <= 0) && (izq != pos)) do
            izq = izq + 1;
      if (pos == izq) then
        break;
      if (strcmp(salida[izq].nombre,salida[pos].nombre) > 0) then
       {
        grupo = salida[pos].grupo;
        materia = salida[pos].materia;
        numero = salida[pos].numero;
        strcpy(nombre,salida[pos].nombre);
        salida[pos].grupo = salida[izq].grupo;
        salida[pos].materia = salida[izq].materia;
        salida[pos].numero = salida[izq].numero;
        strcpy(salida[pos].nombre,salida[izq].nombre);
        salida[izq].grupo = grupo;
        salida[izq].materia = materia;
        salida[izq].numero = numero;
        strcpy(salida[izq].nombre,nombre);
        pos = izq;
       }
     }
   *j = pos;
}

//     Construye el método Quick Sort. Ordena la lista de elementos completa.
void ordena_nombre_salida(struct SALIDA *salida, int n)
{
long int pos, temp;
struct STACK stack;
struct ELEMENTO elemen;

stack.elemento = (ELEMENTO *) farcalloc(n+1,sizeof(ELEMENTO));
if (!stack.elemento)
   {
   printf("Error.");
   exit(1);
   }
pos = 0;
stack.top = 0;
elemen = stack.elemento[stack.top];
if (n >= 1) then
 {
  elemen.inferior = 0;
  elemen.superior = n;
  pon_stack(&stack,elemen,n);
  while (stack.top != 0) do
   {
    quita_stack(&stack,&elemen);
    particion_cadena_salida(salida,elemen.inferior,elemen.superior,&pos);
    if (elemen.inferior < pos - 1) then
          {
           temp = elemen.superior;
           elemen.superior = pos - 1;
           pon_stack(&stack,elemen,n);
           elemen.superior = temp;
          }
    if (pos + 1 < elemen.superior) then
         {
          elemen.inferior = pos + 1;
          pon_stack(&stack,elemen,n);
         }
   }
 }
```

```
farfree(stack.elemento);
}

// Pone submenú de opciones
void ventana_nodo_lista_consulta(void)
{
 pon(tope,150,COPY_PUT);
 cuadro(150,BLUE);
 texto("SALIR",231,LIGHTRED,SMALL_FONT,2,1,2,1);
 texto("<F1> AYUDA",232,LIGHTRED,SMALL_FONT,2,1,2,1);
 texto("<TAB> CAMBIA MENU",233,LIGHTRED,SMALL_FONT,4,3,2,1);
 texto("PAG. 1",234,LIGHTRED,SMALL_FONT,2,1,2,1);
}

/*    Muestra en pantalla los elementos de la lista enlazada MATERIA.  Devuelve el número de
   elemento seleccionado, a la función que hizo la llamada.*/
int mover_pantallas_consulta(int l, int min, int salir, struct MATERIA *p, int cuenta)
{
 struct MATERIA *r,*s;
 struct POS{
             int inicio;
             int fin;
            };
 struct POS *pos;
 int c,regresa,inicio,fin,grupo,ultimo,j,quot,rem,numc,k,dato,max;
 char sig[20],cad[20];
        // Reserva espacio en memoria para pos
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
  {
   printf("Error... Espacio insuficiente");
   exit(1);
  }
 r=p;
 s=p;
 dato=0;
 while (r != NULL) do
 {
        ultimo=(*r).num;
        r=(*r).materias;
 }
 quot=(ultimo/5);
 rem=(ultimo%5);
 if (rem != 0) then
  grupo=quot + 1;
 else
  grupo=quot;
 fin=0;
 for (j=1; j <= grupo; j++)
 {
  inicio=fin+1;
  while (( s != NULL) && (fin < (inicio + 4))) do
  {
   fin=(*s).num;
   s=(*s).materias;
  }
  pos[j].inicio=inicio;
  pos[j].fin=fin;
 }
 ventana_nodo_lista_consulta();
 max=120;
 j=1;
 inicio=pos[j].inicio;
 fin=pos[j].fin;
 numc=ventanas_ponen_materias(p,inicio,fin,&max);
```

```
c=END;
regresa=0;
pon(tope,i,NOT_PUT);
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1:    llama_ayuda(14);            // Llama al módulo de AYUDA
                    break;
        case ABAJO: quita(tope);                // Mueve el cursor sobre los elementos de la pantalla
                    i++;
                    if (i == (max + 1)) then
                          i = min;
                    pon(tope,i,NOT_PUT);
                    break;
        case ARRIBA: quita(tope);
                    i--;
                    if (i == (min - 1)) then
                          i = max;
                    pon(tope,i,NOT_PUT);
                    break;
        case TAB:   if ((i >= min) && (i <= max)) then  // Cambia al menú inferior
                    {
                     regresa=i;
                     i=salir;
                     i=mover_ventana_menu(i,i,salir,regresa);
                     if (i == 231) then
                           {
                            c = ENTER;
                            dato = -2;
                           }
                    }
                    break;
        case PGDN:  quita(tope);                        // Cambia a la siguiente página
                    numc--;
                    for (k=130; k <= numc; k++)
                          quita(tope);

                    for (k=120; k <= max; k++)
                          {
                           setfillstyle(SOLID_FILL,GREEN);
                           bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                          }
                    j++;
                    if (j == (grupo+1)) then
                     j=1;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=120;
                    numc=ventanas_ponen_materias(p,inicio,fin,&max);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,234,LIGHTRED,SMALL_FONT,2,1,2,1);
                    if (i > max) then
                    {
                          i=max;
                          pon(tope,i,NOT_PUT);
                    }
                    else
                     pon(tope,i,NOT_PUT);
                    break;
        case PGUP:  quita(tope);                        // Cambia a la página anterior
                    numc--;
```

```
                    for(k=130; k <= numc; k++)
                        quita(tope);
                    for (k=120; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                    j--;
                    if (j == 0) then
                     j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=120;
                    numc=ventanas_ponen_materias(p,inicio,fin,&max);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,234,LIGHTRED,SMALL_FONT,2,1,2,1);
                    if (i > max) then
                     {
                        i=max;
                        pon(tope,max,NOT_PUT);
                     }
                    else
                     pon(tope,i,NOT_PUT);
                    break;
   }
  }
  if (dato != -2) then          // Obtiene la posición del elemento seleccionado
   {
    dato = i - 119;
    dato=dato + 5*(j-1);
   }
  quita(tope);                                  // Quita la ventana gráfica del cursor
  numc--;
  for (k=130; k <= numc; k++)                   // Quita las ventanas gráficas restantes
    quita(tope);
  for (k=120; k <= max; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  for (k=230; k <= 234; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  quita(tope);                                  // Quita la ventana gráfica 150
  farfree(pos);                                 // Libera el espacio reservado para pos
  return(dato);                                 // Devuelve la posición del elemento seleccionado
}

//      Crea un menú de opciones
void ventana_menu_consulta(void)
{
 texto("<F1> AYUDA",236,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 texto("SALIR",237,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 texto("IMPRIMIR",238,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 texto("TODO",239,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 texto("MATERIA SELECCIONADA",240,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
}


/*    Despliega en pantalla la información de la lista enlazada GRUPO, apartir de inicio hasta
  que llegue a fin.  */
```

```
int ventanas_datos_mat(struct GRUPO *g, int inicio, int fin)
{
 struct GRUPO *s;
 int i,k;
 char numero[100],cadena[100];

 s = g;
 i = 362;
 for (k = inicio; k <= fin; k++)
  {
   pon(tope,i,COPY_PUT);
   itoa((*s).numgrupo,numero,10);
   strcpy(cadena,numero);
   if ((*s).clasegrupo == 1) then
     strcat(cadena," Gpo. Teoria");
   if ((*s).clasegrupo == 2) then
     strcat(cadena," Gpo. Lab. ");
   texto(cadena,i,RED,SMALL_FONT,1,1,2,1);
   i = i + 1;
   pon(tope,i,COPY_PUT);
   texto((*s).nombreprofe,i,RED,SMALL_FONT,1,1,2,1);
   i = i + 1;
   s = (*s).apun;
  }
 return(i);                          // Devuelve el valor de la última ventana colocada
}


/*    Muestra en pantalla los elementos de la lista enlazada GRUPO.
void mover_materias_datos(struct GRUPO *g, int cuenta)
{
 struct GRUPO *r,*s;
 struct POS{
            int inicio;
            int fin;
           };
 struct POS *pos;
 int c,inicio,fin,grupo,ultimo,j,quot,rem,numc,k,a;
 char sig[20], cad[20];
                  // Reserva espacio en memoria para pos
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
  {
   printf("Error.");
   exit(1);
  }
 r=g;
 s=g;
 ultimo = 0;
 while (r != NULL) do
 {
   ultimo=ultimo + 1;
   r=(*r).apun;
 }
 quot=(ultimo/3);
 rem=(ultimo%3);
 if (rem != 0) then
  grupo=quot + 1;
 else
  grupo=quot;
 if (grupo == 0) then
 {
   llama_error(46);
   return;
 }
```

```
 fin=0;
 inicio = 0;
 for (j=1; j <= grupo; j++)
 {
  a = 0;
  while (( s != NULL) && (a < 3)) do
  {
    fin=(*s).num;
    if (a == 0) then
        inicio = fin;
    a = a + 1;
    s=(*s).apun;
  }
  pos[j].inicio=inicio;
  pos[j].fin=fin;
 }
 j=1;
 inicio=pos[j].inicio;
 fin=pos[j].fin;
 numc=ventanas_datos_mat(g,inicio,fin);     // Despliega los datos de la lista enlazada GRUPO
 c=END;
 while (c != ESC) do
 {
  c=bioskey(0);
  switch(c)
  {
        case F1: llama_ayuda(51);                    // Llama al módulo de AYUDA
                 break;
        case PGDN:  numc--;                          // Cambia a la siguiente página
                    for (k=362; k <= numc; k++)
                        quita(tope);
                    j++;
                    if (j == (grupo+1)) then
                      j=1;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_datos_mat(g,inicio,fin);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,359,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;
        case PGUP:  numc--;                                  // Cambia a la página anterior
                    for(k=362; k <= numc; k++)
                        quita(tope);
                    j--;
                    if (j == 0) then
                      j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_datos_mat(g,inicio,fin);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,359,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;
  }
 }
 numc--;
 for (k=362; k <= numc; k++)                 // Quita todas las ventanas gráficas
   quita(tope);
 farfree(pos);                                          // Libera el espacio reservado para pos
}
```

```
/*    Despliega en pantalla todos los datos del nodo seleccionado por dato de la lista enlazada  MATERIA. Incluye los datos de
la lista enlazada GRUPO. */
void consulta_datos_materia(struct MATERIA *p, int dato, char semestre[10])
{
 struct MATERIA *r;
 struct GRUPO *g;
 char numero[100];
 int c,k,ultimo;

 if (dato == 0) then
 {
  llama_error(10);
  return;
 }
 r = p;
 while ((*r).num != dato) do
        r = (*r).materias;
 g = (*r).grupos;
 ultimo = 0;
 while (g != NULL) do
   {
    ultimo = ultimo + 1;
    g = (*g).apun;
   }
 pon(tope,150,COPY_PUT);                                    // Pone ventana gráfica 150
 cuadro(150,BLUE);
 texto("Presione <ESC> para salir . . . .",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                                            // Despliega en pantalla los datos de Materia
 itoa((*r).clave,numero,10);
 grupo_correcto(numero);
 texto(numero,355,LIGHTRED,SMALL_FONT,3,2,2,1);
 texto((*r).nombre,356,LIGHTRED,SMALL_FONT,3,2,2,1);
 c = (*r).tipomat;
 switch(c)
  {
        case 1: texto("OBLIGATORIA",357,LIGHTRED,SMALL_FONT,3,2,2,1);
                break;
        case 2: texto("OPTATIVA",357,LIGHTRED,SMALL_FONT,3,2,2,1);
                break;
  }
 cuadro(286,CYAN);
 c = (*r).lab;
 switch(c)
  {
        case 1: strcpy(numero,"Materia con laboratorio opcional");
                strcat(numero,"    Semestre ... ");
                strcat(numero,semestre);
                texto(numero,358,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                break;
        case 2: strcpy(numero,"Materia con laboratorio obligatorio");
                strcat(numero,"    Semestre ... ");
                strcat(numero,semestre);
                texto(numero,358,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                break;
        case 3: strcpy(numero,"Materia sin laboratorio");
                strcat(numero,"    Semestre ... ");
                strcat(numero,semestre);
                texto(numero,358,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                break;
  }
 texto("PAG. 1",359,RED,SANS_SERIF_FONT,1,2,1,2);
                                                            // Pone ventanas de etiquetas
 texto("GRUPOS DE LA MATERIA",360,BLUE,SMALL_FONT,1,1,2,1);
 texto("PROFESORES",361,BLUE,SMALL_FONT,1,1,2,1);
 mover_materias_datos((*r).grupos,ultimo);    // Pone en ventanas los datos de la estructura de
```

```
                                        // GRUPO a donde apunta r
  for (k = 355; k <= 361; k++)          // Quita todas las ventanas gráficas
     {
      setfillstyle(SOLID_FILL,GREEN);
      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  k = 286;
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  quita(tope);
}


/*    Permite seleccionar la forma de consultar los datos de la lista enlazada MATERIA. Devuel-
 ve el nodo seleccionado.*/
int escoje_consulta_materia(struct MATERIA *p, int *i)
{
  struct MATERIA *r;
  int j,k,dato, clave;
  char cadena[100];

  r = p;
  dato = 0;
     pon(tope,150,COPY_PUT);                    // Pone ventana gráfica 150
     cuadro(150,BLUE);
     pon(tope,246,COPY_PUT);                    // Pone ventana gráfica 246
     texto("Presione <ENTER> para seleccionar o <ESC> para salir ....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
     pon(tope,165,COPY_PUT);
     texto("Seleccione la forma de consultar materias ...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
     pon(tope,349,COPY_PUT);                    // Pone ventana gráfica 349
     pon(tope,350,COPY_PUT);                    // Pone ventana gráfica 350
     pon(tope,351,COPY_PUT);                    // Pone ventana gráfica 351
     texto("POR CLAVE",349,BLUE,SMALL_FONT,3,2,2,1);
     texto("POR NOMBRE",350,BLUE,SMALL_FONT,3,2,2,1);
     texto("CONSULTAR TODAS",351,BLUE,SMALL_FONT,3,2,2,1);
     pon(tope,349,NOT_PUT);                     // Pone ventana gráfica del cursor
     j = opciones1(349,349,351,13);             // Mueve la pila de ventanas
     quita(tope);                               // Quita ventana gráfica del cursor
     quita(tope);                               // Quita ventana gráfica 351
     quita(tope);                               // Quita ventana gráfica 350
     quita(tope);                               // Quita ventana gráfica 349
     if (j != -1) then
      {
          if (j != 351) then
           {
            texto("Presione <ENTER> cuando termine de escribir la cadena ....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
            texto("Se ha seleccionado la opcion de ...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
            switch(j)
              {
                    case 349: pon(tope,352,COPY_PUT);
                              texto("CONSULTA DE MATERIA ESCRIBIENDO LA CLAVE",352,BLUE,SMALL_FONT,3,2,2,1);
                              texto("CLAVE . . .",359,BLUE,SMALL_FONT,3,2,2,1);
                              k = 1;
                              while (k == 1) do
                               {
                                lee_cadena(cadena,4,354,BLUE,SMALL_FONT,3,2,2,1);
                                k = error_numero_entero(cadena);
                               }
                              texto(cadena,354,BLUE,SMALL_FONT,3,2,2,1);
                              clave = atoi(cadena);
                              dato = busca_materia_clave(r,clave);
                              if (dato == 0) then
                              {
                                llama_error(10);
                                j = -1;
```

```
                    }
                    break;

            case 360: pon(tope,352,COPY_PUT);
                    texto("CONSULTA DE MATERIA ESCRIBIENDO EL NOMBRE",352,BLUE,SMALL_FONT,3,2,2,1);
                    texto("NOMBRE ...",353,BLUE,SMALL_FONT,3,2,2,1);
                    lee_cadena(cadena,28,354,BLUE,SMALL_FONT,3,2,2,1);
                    texto(cadena,354,BLUE,SMALL_FONT,3,2,2,1);
                    for (k=0; cadena[k]; k++)
                     cadena[k] = toupper(cadena[k]);
                    dato = busca_materia_nombre(r,cadena);
                    if (dato == 0) then
                     {
                      llama_error(10);
                      j = -1;
                     }
                    break;
              }
            for (k = 353; k <= 354; k++)
              {
                  setfillstyle(SOLID_FILL,GREEN);
                  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
              }
            quita(tope);                          // Quita ventana gráfica 352
           }
         }
    quita(tope);                                  // Quita ventana gráfica 165
    quita(tope);                                  // Quita ventana gráfica 246
    quita(tope);                                  // Quita ventana gráfica 150
  *i = j;
  return(dato);                                         // Devuelve el nodo de la lista MATERIA seleccionado
}


//    Llama a las funciones para la consulta de datos de la lista enlazada MATERIA.
int consulta_materias(struct MATERIA *p, char semestre[10])
{
  struct ARREGLO *arreglo;
  struct MATERIA *r,*s;
  int dato,c,j,k,ultimo,m;

  pon(tope,100,COPY_PUT);                         // Pone ventana gráfica 100
  texto("CONSULTA DE LA LISTA DE MATERIAS",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
  if ((*p).num == 0) then
  {
   llama_error(49);
   quita(tope);                                   // Quita ventana gráfica 100
   return(0);
  }
  c = 0;
  m = 0;
  while (m != -1) do
   {
                    // Seleccionar el tipo de operacion sobre la lista de materias
     pon(tope,352,COPY_PUT);
     texto("Seleccione la operación de materias que desea realizar ...",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
     pon(tope,253,COPY_PUT);
     pon(tope,254,COPY_PUT);
     texto("CONSULTAR",253,BLUE,SANS_SERIF_FONT,1,2,2,3);
     texto("IMPRIMIR",254,BLUE,SANS_SERIF_FONT,1,2,2,3);
     pon(tope,150,COPY_PUT);
     cuadro(150,BLUE);
     texto("Presione <ENTER> para seleccionar o <ESC> para salir ....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
     pon(tope,253,NOT_PUT);
     m = opciones1(253,253,254,36);
```

```
quita(tope);                                    // Quita ventana gráfica del cursor
quita(tope);                                    // Quita ventana gráfica 150
quita(tope);                                    // Quita ventana gráfica 254
quita(tope);                                    // Quita ventana gráfica 253
quita(tope);                                    // Quita ventana gráfica 352
switch(m)
    {
     case 253: // Operación de consulta
                r = p;
                i = 0;
                dato = 0;
                while (i != -1) do
                     {
                      dato = escoje_consulta_materia(r,&i);
                      if (dato != 0) then
                        c = dato;
                      if (i != -1) then
                       {
                        if (i != 351) then
                              consulta_datos_materia(r,dato,semestre);
                        else
                         {
                              s = p;
                              ultimo = 0;
                              while (s != NULL) do
                               {
                                 ultimo = ultimo + 1;
                                 s = (*s).materias;
                               }
                              while (dato != -2) do
                              {
                                dato=mover_pantallas_consulta(120,120,231,p,ultimo);
                                if (dato > 0) then
                                 {
                                   consulta_datos_materia(r,dato,semestre);
                                   c = dato;
                                 }
                              }
                         }
                       }
                     }
                break;

     case 254:// Manda a impresión la lista enlazada MATERIA
                r = p;
                ultimo = 0;
                while (r != NULL) do
                 {
                      ultimo = ultimo + 1;
                      r = (*r).materias;
                 }
                arreglo = (ARREGLO *) farcalloc(ultimo+1,sizeof(ARREGLO));
                if (!arreglo)
                 {
                      printf("Error.");
                      exit(1);
                 }
                 // Ordenar la lista enlazada MATERIA por el campo CLAVE
                 r = p;
                 k = 0;
                 while (r != NULL) do
                      {
                       arreglo[k].numero = (*r).clave;
                       arreglo[k].grupo = (*r).lab;
                       strcpy(arreglo[k].nombre,(*r).nombre);
```

```
                    arreglo[k].materia = (*r).num;
                    k = k + 1;
                    r = (*r).materias;
                   }
                 arreglo[k].numero = -2;
                 arreglo[k].grupo = -2;
                 strcpy(arreglo[k].nombre,"-2");
                 arreglo[k].materia = -2;
                 ultimo = ultimo - 1;
                 pon(tope,352,COPY_PUT);          // Pone ventana gráfica 352
                 texto("Seleccione la forma de impresión de materias ...",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
                 pon(tope,253,COPY_PUT);          // Pone ventana gráfica 353
                 pon(tope,254,COPY_PUT);          // Pone ventana gráfica 354
                 texto("ORDENADA POR CLAVE",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
                 texto("ORDENADA POR NOMBRE",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
                 pon(tope,253,NOT_PUT);                    // Pone ventana gráfica del cursor
                 k = opciones1(253,253,254,37);   // Mueve la pila de ventanas
                 quita(tope);                     // Quita ventana gráfica del cursor
                 quita(tope);                     // Quita ventana gráfica 254
                 quita(tope);                     // Quita ventana gráfica 253
                 quita(tope);                     // Quita ventana gráfica 352
                 switch(k)
                      {
                      case 253: ordena_arreglo(arreglo,ultimo); // Ordena por el campo clave
                                break;  // Ordena por el campo nombre
                      case 254: ordena_nombre_arreglo(arreglo,ultimo);
                                break;
                      }
                 imprimir(p,arreglo,2,semestre);          // Manda a impresión el archivo
                 farfree(arreglo);
                 m = 0;
                break;
        }
  }
  quita(tope);                                    // Quita ventana gráfica 100
  return(c);
}

//    Llama a las funciones de consulta de los datos de la lista enlazada GRUPO.
void consulta_grupos(struct MATERIA *p, int *d, char semestre[10])
{
  struct MATERIA *r;
  struct ARREGLO *arreglo;
  int i,j,c,dato,k,numgrupo,b,a,cuenta,tipo;
  char cadena[10];

  cuenta = 0;
  dato = *d;
  pon(tope,100,COPY_PUT);                         // Pone ventana gráfica 100
  texto("CONSULTA DE GRUPOS",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
  if ((*p).num == 0) then
   {
    llama_error(44);
    quita(tope);                                          // Quita ventana gráfica 100
    return;
   }
  ventana_menu_consulta();                        // Pone el menú de opciones
  i = 237;
  j = 1;
  pon(tope,i,NOT_PUT);                            // Pone ventana gráfica del cursor
  while (j == 1)
   {
     c = biosKey(0);
     switch(c)
      {
```

```
        case F1: llama_ayuda(36);       // Llama al módulo AYUDA
                 break;
        case DERECHA:  quita(tope);   // Desplaza el cursor sobre el submenú de opciones
                       i++;
                       if (i == 241) then
                         i = 237;
                       pon(tope,i,NOT_PUT);
                       break;

        case IZQUIERDA: quita(tope);
                       i--;
                       if (i == 236) then
                         i = 240;
                       pon(tope,i,NOT_PUT);
                       break;

        case ENTER: quita(tope);                // Quita ventana gráfica del cursor
                    r = p;
                    switch(i)
                     {
                          case 237:                         // Opción SALIR
                                    j = 0;
                                    i = 237;
                                    pon(tope,i,NOT_PUT);   // Pone ventana gráfica del cursor
                                    break;

                          case 238:                         // Opción IMPRIMIR
                                    if (cuenta < 1) then
                                      llama_error(52);
                                    else
                                      imprimir(r,arreglo,1,semestre);
                                    i = 238;
                                    pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                                    break;

                          case 239:                         // Opción "."
                                    pon(tope,150,COPY_PUT);      // Pone ventana gráfica 150
                                    cuadro(150,BLUE);
                                    pon(tope,246,COPY_PUT);      // Pone ventana gráfica 246
                                    texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,165,COPY_PUT);      // Pone ventana gráfica 165
                                    texto("Seleccione la clase de grupos que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,310,COPY_PUT);      // Pone ventana gráfica 310
                                    pon(tope,311,COPY_PUT);      // Pone ventana gráfica 311
                                    pon(tope,312,COPY_PUT);      // Pone ventana gráfica 312
                                    texto("GRUPOS DE TEORIA",310,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("GRUPOS DE LABORATORIO",311,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("GRUPOS DE TEOR. Y LAB.",312,BLUE,SMALL_FONT,2,2,2,1);

                                    pon(tope,310,NOT_PUT);    // Pone ventana gráfica del cursor
                                    i = opciones1(310,310,312,39);  // Mueve la pila de ventanas

                                    quita(tope);          // Quita ventana gráfica del cursor
                                    quita(tope);          // Quita ventana gráfica 312
                                    quita(tope);      // Quita ventana gráfica 311
                                    quita(tope);      // Quita ventana gráfica 310
                                    quita(tope);      // Quita ventana gráfica 165
                                    if (i != -1) then
                                     {
                                       quita(tope);   // Quita ventana gráfica 246
                                       pon(tope,246,COPY_PUT);     // Pone ventana gráfica 246
                                       texto("Presione <ENTER> cuando termine de escribir la cadena
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
```

```
                pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                texto("Se ha seleccionado la opcion de ...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);

                switch(l)
                    {                            // Pone ventana gráfica 313
                     case 310: pon(tope,313,COPY_PUT);
                               texto("CONSULTA DE GRUPOS DE
TEORIA",313,BLUE,SMALL_FONT,3,2,2,1);
                               break;
                     case 311: pon(tope,313,COPY_PUT);
                               texto("CONSULTA DE GRUPOS DE
LABORATORIO",313,BLUE,SMALL_FONT,3,2,2,1);
                               break;
                     case 312: pon(tope,313,COPY_PUT);
                               texto("CONSULTA DE GRUPOS DE TEORIA Y
LABORATORIO",313,BLUE,SMALL_FONT,3,2,2,1);
                               break;
                    }
                texto("Escriba el numero de grupo que desea consultar o escriba los primeros digitos
seguidos",314,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                texto("de un * (ej. _*, __*, ___*), o solo un * para consultar todos los grupos
existentes.",315,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                texto("GRUPO . . . .",316,BLUE,SMALL_FONT,3,2,2,1);
                lee_cadena(cadena,4,317,BLUE,SMALL_FONT,3,2,2,1);  // Lee desde el teclado
                texto(cadena,317,BLUE,SMALL_FONT,3,2,2,1);
                numgrupo = atoi(cadena);          // Convierte cadena en un entero
                for (k = 314; k <= 317; k++)
                    {
                     setfillstyle(SOLID_FILL,GREEN);
                     bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                    }
                    quita(tope);     // Quita ventana gráfica 313
                    quita(tope);     // Quita ventana gráfica 165
                b = 0;
                if (!strcmp(cadena,"*")) then
                  b = 2;
                else
                 {
                  for (k=0; cadena[k]; k++)
                      {
                       if (!isdigit(cadena[k]))
                          b = 1;
                      }
                 }
                                 // Busca el dato según el tipo de cadena
                tipo = l;
                 switch(tipo)
                  {
                        case 310: if (b == 0) then
                                        busca_grupo_tipo_cuenta(r,numgrupo,1,&cuenta);
                                  if (b == 1) then
                                      busca_grupo_cadena_tipo_cuenta(r,numgrupo,1,&cuenta);
                                  if (b == 2) then
                                        busca_grupo_todo_tipo_cuenta(r,1,&cuenta);
                                  if (b == 3) then
                                        busca_grupo_materia_tipo_cuenta(r,1,&cuenta);
                                  break;

                        case 311: if (b == 0) then
                                        busca_grupo_tipo_cuenta(r,numgrupo,2,&cuenta);
                                  if (b == 1) then
                                      busca_grupo_cadena_tipo_cuenta(r,numgrupo,2,&cuenta);
                                  if (b == 2) then
                                        busca_grupo_todo_tipo_cuenta(r,2,&cuenta);
                                  if (b == 3) then
```

```
                                        busca_grupo_materia_tipo_cuenta(r,2,&cuenta);
                                    break;

                                case 312: if (b == 0) then
                                        busca_grupo_teolab_cuenta(r,numgrupo,&cuenta);
                                    if (b == 1) then
                                        busca_grupo_cadena_teolab_cuenta(r,
numgrupo,&cuenta);

                                    if (b == 2) then
                                        busca_gteolab_todo_cuenta(r,&cuenta);
                                    if (b == 3) then
                                        busca_gteolab_materia_cuenta(r,&cuenta);
                                    break;
                            }
                                    // Asigna espacio a arreglo
                            arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                            if (!arreglo)
                                {
                                printf("Error.");
                                exit(1);
                                }
                        // Despliega en pantalla los datos de la lista enlazada GRUPO
                            a = 0;
                            while (a != -2) do
                                {
                                 a = grupos_teoria(r,numgrupo,arreglo,b,tipo,cuenta);
                                 if (a >= 0) then
                                   consulta_datos_grupos_todos(r,arreglo,a,b,semestre);
                                }
                            farfree(arreglo);
                        }
                        quita(tope); /* 246 */
                        quita(tope); /* 150 */
                        i = 239;
                        pon(tope,i,NOT_PUT);
                        break;

                case 240: // Opción MATERIA SELECCIONADA
                        if (dato != 0) then
                        {
                         while ((*r).num != dato)
                                r = (*r).materias;
                         if ((*r).grupos == NULL) then
                          llama_error(46);
                         else
                         {
                          pon(tope,150,COPY_PUT);  // Pon ventana gráfica 150
                          cuadro(150,BLUE);
                          pon(tope,240,COPY_PUT);  // Pon ventana gráfica 240
                          texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
                          pon(tope,246,COPY_PUT);  // Pon ventana gráfica 246
                          texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                          pon(tope,165,COPY_PUT);  // Pon ventana gráfica 165
                          texto("Seleccione la clase de grupos que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                          pon(tope,310,COPY_PUT);  // Pon ventana gráfica 310
                          pon(tope,311,COPY_PUT);  // Pon ventana gráfica 311
                          pon(tope,312,COPY_PUT);  // Pon ventana gráfica 312
                          texto("GRUPOS DE TEORIA",310,BLUE,SMALL_FONT,2,2,2,1);
                          texto("GRUPOS DE LABORATORIO",311,BLUE,SMALL_FONT,2,2,2,1);
                          texto("GRUPOS DE TEOR. Y LAB.",312,BLUE,SMALL_FONT,2,2,2,1);
                          pon(tope,310,NOT_PUT); // Pon ventana gráfica 310
                          i = opciones1(310,310,312,40);  // Mueve la pila de ventanas
                          quita(tope);       // Quita ventana gráfica del cursor
```

```
                                        quita(tope);      // Quita ventana gráfica 312
                                        quita(tope);      // Quita ventana gráfica 311
                                        quita(tope);      // Quita ventana gráfica
                                        quita(tope);      // Quita ventana gráfica 165
                                        if (i != -1) then
                                        {
                                             quita(tope);      // Quita ventana gráfica 246
                                             pon(tope,246,COPY_PUT);   // Pone ventana gráfica 246
                                             texto("Presione <ENTER> cuando termine de escribir la cadena
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                             // Busca el dato según el tipo de cadena
                                             tipo = i;
                                             b = 3;
                                             switch(tipo)
                                             {
                                                  case 310: busca_grupo_materia_tipo_cuenta(r,1,&cuenta);
                                                            break;
                                                  case 311: busca_grupo_materia_tipo_cuenta(r,2,&cuenta);
                                                            break;
                                                  case 312: busca_gtsolab_materia_cuenta(r,&cuenta);
                                                            break;
                                             }
                                                  // Asigna espacio en memoria a arreglo
                                             arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                                             if (!arreglo)
                                               {
                                               printf("Error.");
                                               exit(1);
                                               }
                                             // Despliega en pantalla los datos de la lista enlazada GRUPO
                                             a = 0;
                                             while (a != -2) do
                                               {
                                                 a = grupos_teoria(r,numgrupo,arreglo,b,tipo,cuenta);
                                                 if (a >= 0) then
                                                   consulta_datos_grupos_todos(r,arreglo,a,b,semestre);
                                               }
                                             farfree(arreglo);
                                        }
                                        quita(tope);      // Quita ventana gráfica 246
                                        quita(tope);      // Quita ventana gráfica 240
                                        quita(tope);      // Quita ventana gráfica 150
                                      }
                                       while (r != NULL)
                                             r = (*r).materias;
                                    }
                                    else
                                      llama_error(43);
                                    i = 240;
                                    pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                                    break;
                          }
     }
   }
  quita(tope);                    // Quita ventana gráfica del cursor
  quita(tope);                    // Quita ventana gráfica 100
}


//   Despliega en pantalla los datos correspondientes a Profesores de la lista enlazada GRUPO.
void consulta_profesores(struct MATERIA *p, int *d, char semestre[10])
{
  struct MATERIA *r;
  struct ARREGLO *arreglo;
  int i,j,c,dato,a,b,k,tipo,cuenta;
```

```
char profesor[100],cadena[100],cad[50];

cuenta = 0;
dato = *d;
pon(tope,100,COPY_PUT);                    // Pone ventana gráfica 100
texto("CONSULTA DE PROFESORES",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
if ((*p).num == 0) then
 {
  llama_error(45);
  quita(tope);                             // Quita ventana gráfica 100
  return;
 }
ventana_menu_consulta();                   // Pone submenú de opciones
i = 237;
j = 1;
pon(tope,i,NOT_PUT);                       // Pone ventana gráfica del cursor
while (j == 1)
 {
  c = bioskey(0);
  switch(c)
   {
       case F1: llama_ayuda(41);           // Llama al módulo AYUDA
                break;
       case DERECHA:  quita(tope);         // Permite el desplazamiento del cursor sobre
                      i++;                 // el submenú de opciones.
                      if (i == 241) then
                        i = 237;
                      pon(tope,i,NOT_PUT);
                      break;
       case IZQUIERDA: quita(tope);
                      i-;
                      if (i == 236) then
                        i = 240;
                      pon(tope,i,NOT_PUT);
                      break;
       case ENTER: quita(tope);                    // Quita ventana gráfica del cursor
                   r = p;
                   switch(i)
                    {
                        case 237:                  // Opción SALIR
                                  j = 0;
                                  i = 237;
                                  pon(tope,i,NOT_PUT); // Pone ventana gráfica del cursor
                                  break;

                        case 238:                  // Opción IMPRIMIR
                                  if (cuenta < 1) then
                                    llama_error(52);
                                  else
                                   imprimir_profe_salida(r,semestre,b,tipo,profesor,cuenta);
                                  i = 238;
                                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                                  break;

                        case 239:                  // Opción *.*
                                  pon(tope,150,COPY_PUT);   // Pone ventana gráfica 150
                                  cuadro(150,BLUE);
                                  pon(tope,246,COPY_PUT);   // Pone ventana gráfica 246
                                  texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                  pon(tope,165,COPY_PUT);   // Pone ventana gráfica 165
                                  texto("Seleccione el tipo de profesor que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                  pon(tope,342,COPY_PUT);   // Pone ventana gráfica 342
                                  pon(tope,343,COPY_PUT);   // Pone ventana gráfica 343
```

```
                                pon(tope,344,COPY_PUT);   // Pone ventana gráfica 344
                                pon(tope,345,COPY_PUT);   // Pone ventana gráfica 345
                                texto("TITULAR",342,BLUE,SMALL_FONT,2,2,2,1);
                                texto("AYUDANTE",343,BLUE,SMALL_FONT,2,2,2,1);
                                texto("SIN NOMBRAMIENTO",344,BLUE,SMALL_FONT,2,2,2,1);
                                texto("TODOS",345,BLUE,SMALL_FONT,2,2,2,1);
                                pon(tope,342,NOT_PUT);    // Pone ventana gráfica 342
                                i = opciones1(342,342,345,42);  // Mueve la pila de ventanas
                                quita(tope);          // Quita ventana gráfica del cursor
                                quita(tope);          // Quita ventana gráfica 345
                                quita(tope);          // Quita ventana gráfica 344
                                quita(tope);          // Quita ventana gráfica 343
                                quita(tope);          // Quita ventana gráfica 345
                                quita(tope);          // Quita ventana gráfica 165
                                if (i != -1) then
                                 {
                                  quita(tope); // Quita ventana gráfica 246
                                  pon(tope,246,COPY_PUT);  // Pone ventana gráfica 246
                                  texto("Presione <ENTER> cuando termine de escribir la cadena
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                  pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                                  texto("Se ha seleccionado la opcion de ...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                  switch(i)
                                        {          // Pone ventana gráfica 313
                                          case 342: pon(tope,313,COPY_PUT);
                                                    texto("CONSULTA DE PROFESORES
TITULARES",313,BLUE,SMALL_FONT,3,2,2,1);
                                                    break;
                                          case 343: pon(tope,313,COPY_PUT);
                                                    texto("CONSULTA DE AYUDANTES DE
PROFESOR",313,BLUE,SMALL_FONT,3,2,2,1);
                                                    break;
                                          case 344: pon(tope,313,COPY_PUT);
                                                    texto("CONSULTA DE PROFESORES SIN
NOMBRAMIENTO",313,BLUE,SMALL_FONT,3,2,2,1);
                                                    break;
                                          case 345: pon(tope,313,COPY_PUT);
                                                    texto("CONSULTA DE TODOS LOS
PROFESORES",313,BLUE,SMALL_FONT,3,2,2,1);
                                                    break;
                                        }
                                  texto("Escriba el nombre completo del profesor que desea consultar o escriba las
primeras letras",314,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                                  texto(" o solo un * para consultar todos los profesores existentes en la lista.
",315,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                                  texto("PROFESOR . . . .",316,BLUE,SMALL_FONT,3,2,2,1);
                                  lee_cadena(cadena,32,317,BLUE,SMALL_FONT,2,2,2,1);
                                        // Lee una cadena desde el teclado
                                  texto(cadena,317,BLUE,SMALL_FONT,2,2,2,1);
                                  for (k=0; cadena[k]; k++)
                                  cadena[k] = toupper(cadena[k]);
                                        // Convierte cadena en mayúsculas
                                  strcpy(profesor,cadena);
                                  for (k = 314; k <= 317; k++)
                                        {
                                         setfillstyle(SOLID_FILL,GREEN);
                                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                        }
                                        quita(tope);                    // Quita ventana gráfica 313
                                        quita(tope);                    // Quita ventana gráfica 165
                                  b = 0;
                                  if (!strcmp(cadena,"*")) then
                                    b = 2;
                                  else
                                        {
```

```
        strcpy(cad,"\0");
        for (k = 0; profesor[k] && b != 1; k++)
          {
           if (cadena[k] == cad[0]) then
                b = 1;
          }
        if (b == 1) then
          {
           for (a=0; a < (k-1); a++)
                cad[a] = cadena[a];
           cad[a] = '\0';
           strcpy(profesor,cad);
          }
      }
                              // Busca el dato según el tipo de cadena
tipo = l;
switch(tipo)
      {
        case 342: if (b == 0) then
                              busca_profe_tipo_cuenta(r,profesor,1,&cuenta);
                  if (b == 1) then
                  busca_profe_cadena_tipo_cuenta(r,profesor,1,&cuenta);
                  if (b == 2) then
                              busca_profe_todos_tipo_cuenta(r,1,&cuenta);
                  if (b == 3) then
                              busca_profe_materia_tipo_cuenta(r,1,&cuenta);
                  break;
        case 343: if (b == 0) then
                              busca_profe_tipo_cuenta(r,profesor,2,&cuenta);
                  if (b == 1) then
                  busca_profe_cadena_tipo_cuenta(r,profesor,2,&cuenta);
                  if (b == 2) then
                              busca_profe_todos_tipo_cuenta(r,2,&cuenta);
                  if (b == 3) then
                              busca_profe_materia_tipo_cuenta(r,2,&cuenta);
                  break;
        case 344: if (b == 0) then
                              busca_profe_tipo_cuenta(r,profesor,3,&cuenta);
                  if (b == 1) then
                              busca_profe_cadena_tipo_cuenta(r,profesor,3,&cuenta);
                  if (b == 2) then
                              busca_profe_todos_tipo_cuenta(r,3,&cuenta);
                  if (b == 3) then
                              busca_profe_materia_tipo_cuenta(r,3,&cuenta);
                  break;
        case 345: if (b == 0) then
                              busca_profe_cuenta(r,profesor,&cuenta);
                  if (b == 1) then
                              busca_profe_cadena_cuenta(r,profesor,&cuenta);
                  if (b == 2) then
                              busca_profe_todos_cuenta(r,&cuenta);
                  if (b == 3) then
                              busca_profe_todos_materia_cuenta(r,&cuenta);
                  break;
      }
                        // Asigna espacio en memoria para arreglo
 arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
      if (!arreglo)
        {
         printf("Error.");
         exit(1);
        }
a = 0;                  // Despliega en pantalla los datos de profesor
while (a != -2) do
      {
```

```
                        a = profesor_todos(r,profesor,arreglo,b,tipo,cuenta);
                        if (a >= 0) then
                          consulta_datos_profesor_todos(r,arreglo,a,b,semestre);
                       }
                    farfree(arreglo);
                  }
                  quita(tope);                    // Quita ventana gráfica 246
                  quita(tope);                    // Quita ventana gráfica 150
                  i = 239;
                  pon(tope,i,NOT_PUT);            // Quita ventana gráfica del cursor
                  break;

            case 240:                             // Opción MATERIA SELECCIONADA
                if (dato != 0) then
                    {
                    while ((*r).num != dato)
                          r = (*r).materias;
                    if ((*r).grupos == NULL) then
                      llama_error(46);
                    else
                     {
                      pon(tope,150,COPY_PUT);          // Pone ventana gráfica 150
                      cuadro(150,BLUE);
                      pon(tope,240,COPY_PUT);          // Pone ventana gráfica 240
                      texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
                      pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
                      texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                      pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                      texto("Seleccione el tipo de profesor que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                      pon(tope,342,COPY_PUT);          // Pone ventana gráfica 342
                      pon(tope,343,COPY_PUT);          // Pone ventana gráfica 343
                      pon(tope,344,COPY_PUT);          // Pone ventana gráfica 344
                      pon(tope,345,COPY_PUT);          // Pone ventana gráfica 345
                      texto("TITULAR",342,BLUE,SMALL_FONT,2,2,2,1);
                      texto("AYUDANTE",343,BLUE,SMALL_FONT,2,2,2,1);
                      texto("SIN NOMBRAMIENTO",344,BLUE,SMALL_FONT,2,2,2,1);
                      texto("TODOS",345,BLUE,SMALL_FONT,2,2,2,1);
                      pon(tope,342,NOT_PUT);       // Pone ventana gráfica del cursor
                      i = opciones1(342,342,345,43);   // Mueve la pila de ventanas
                      quita(tope);                     // Quita ventana gráfica del cursor
                      quita(tope);     // Quita ventana gráfica 345
                      quita(tope);     // Quita ventana gráfica 344
                      quita(tope);     // Quita ventana gráfica 343
                      quita(tope);     // Quita ventana gráfica 342
                      quita(tope);     // Quita ventana gráfica 165
                      if (i != -1) then
                       {
                            quita(tope);          // Quita ventana gráfica 246
                            pon(tope,246,COPY_PUT);    // Pone ventana gráfica 246
                            texto("Presione <ENTER> cuando termine de escribir la cadena
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                                  // Busca el dato según el tipo de cadena
                            tipo = i;
                            b = 3;
                            switch(tipo)
                             {
                                case 342: busca_profe_materia_tipo_cuenta(r,1,&cuenta);
                                          break;
                                case 343: busca_profe_materia_tipo_cuenta(r,2,&cuenta);
                                          break;
                                case 344: busca_profe_materia_tipo_cuenta(r,3,&cuenta);
                                          break;
                                case 345: busca_profe_todos_materia_cuenta(r,&cuenta);
```

```
                                        break;
                                    }
                                        // Asigna espacio en memoria para arreglo
                                    arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                                    if (!arreglo)
                                      {
                                        printf("Error.");
                                        exit(1);
                                      }
                                    // Despliega la información de Profesor de la lista GRUPO
                                    a = 0;
                                    while (a != -2) do
                                      {
                                        a = profesor_todos(r,profesor,arreglo,b,tipo,cuenta);
                                        if (a >= 0) then
                                          consulta_datos_profesor_todos(r,arreglo,a,b,semestre);
                                      }
                                    farfree(arreglo);
                                  }
                                  quita(tope);      // Quita ventana gráfica 248
                                  quita(tope);      // Quita ventana gráfica 240
                                  quita(tope);      // Quita ventana gráfica 150
                                }
                                while (r != NULL)
                                        r = (*r).materias;
                              }
                              else
                                llama_error(43);
                              i = 240;
                              pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                              break;
                          }
      }
    }
  quita(tope);                                  // Quita ventana gráfica del cursor
  quita(tope);                                  // Quita ventana gráfica 100
}

//    Llama las funciones para desplegar la información de Salones en pantalla.
void consulta_salones(struct MATERIA *p, int *d, char semestre[10])
{
  struct MATERIA *r;
  struct ARREGLO *arreglo;
  int i,j,c,dato,k,b,a,tipo,cuenta;
  long int numsalon;
  char cadena[20];

  cuenta = 0;
  dato = *d;
  pon(tope,100,COPY_PUT);                       // Pone ventana gráfica 100
  texto("CONSULTA DE SALONES",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
  if ((*p).num == 0) then
    {
      llama_error(46);
      quita(tope);                              // Quita ventana gráfica 100
      return;
    }
  ventana_menu_consulta();                      // Pone un submenú de opciones
  i = 237;
  j = 1;
  pon(tope,i,NOT_PUT);                          // Pone ventana gráfica del cursor
  while (1 == 1)
    {
      c = bioskey(0);
      switch(c)
```

```
{
        case F1 : llama_ayuda(44);              // Llama al módulo AYUDA
                  break;
        case DERECHA:  quita(tope);             // Permite el desplazamiento del cursor sobre
                       i++;                     // el submenú de opciones
                       if (i == 241) then
                         i = 237;
                       pon(tope,i,NOT_PUT);
                       break;
        case IZQUIERDA: quita(tope);
                       i--;
                       if (i == 236) then
                          i = 240;
                       pon(tope,i,NOT_PUT);
                       break;
        case ENTER: quita(tope);                // Quita ventana gráfica del cursor
                    r = p;
                    switch(i)
                     {
                          case 237:                         // Opción SALIR
                                    j = 0;
                                    i = 237;
                                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                                    break;

                          case 238:                         // Opción IMPRIMIR
                                    if (cuenta < 1) then
                                      llama_error(52);
                                    else
                                     imprimir_salon_salida(r,semestre,b,tipo,numsalon,cuenta);
                                    i = 238;
                                    pon(tope,i,NOT_PUT);
                                    break;

                          case 239:                         // Opción *.*
                                    pon(tope,150,COPY_PUT);  // Pone ventana gráfica 150
                                    cuadro(150,BLUE);
                                    pon(tope,246,COPY_PUT);  // Pone ventana gráfica 246
                                    texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,165,COPY_PUT);  // Pone ventana gráfica 165
                                    texto("Seleccione el tipo de salon que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,332,COPY_PUT);  // Pone ventana gráfica 332
                                    pon(tope,333,COPY_PUT);  // Pone ventana gráfica 333
                                    pon(tope,334,COPY_PUT);  // Pone ventana gráfica 334
                                    pon(tope,335,COPY_PUT);  // Pone ventana gráfica 335
                                    pon(tope,336,COPY_PUT);  // Pone ventana gráfica 336
                                    texto("DIBUJO",332,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("LABORATORIO",333,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("CHICO",334,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("GRANDE",335,BLUE,SMALL_FONT,2,2,2,1);
                                    texto("TODOS",336,BLUE,SMALL_FONT,2,2,2,1);
                                    pon(tope,332,NOT_PUT);   // Pone ventana gráfica del cursor
                                    i = opciones1(332,332,336,45); // Mueve la pila de ventanas
                                    quita(tope);                        // Quita ventana gráfica del cursor
                                    quita(tope);                        // Quita ventana gráfica 336
                                    quita(tope);                        // Quita ventana gráfica 335
                                    quita(tope);                        // Quita ventana gráfica 334
                                    quita(tope);                        // Quita ventana gráfica 333
                                    quita(tope);                        // Quita ventana gráfica 332
                                    quita(tope);                        // Quita ventana gráfica 165
                                    if (i != -1) then
                                     {
                                       quita(tope);                 // Quita ventana gráfica 246
```

```
                                        pon(tope,248,COPY_PUT);          // Pone ventana gráfica 248
                                        texto("Presione <ENTER> cuando termine de escribir la cadena
....",248,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                        pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                                        texto("Se ha seleccionado la opcion de ...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                        switch(i)
                                              {                          // Pone ventana gráfica 313
                                                case 332: pon(tope,313,COPY_PUT);
                                                          texto("CONSULTA DE SALONES DE
DIBUJO",313,BLUE,SMALL_FONT,3,2,2,1);
                                                          break;
                                                case 333: pon(tope,313,COPY_PUT);
                                                          texto("CONSULTA DE SALONES DE
LABORATORIO",313,BLUE,SMALL_FONT,3,2,2,1);
                                                          break;
                                                case 334: pon(tope,313,COPY_PUT);
                                                          texto("CONSULTA DE SALONES CHICOS (MAX. 30
ALUMNOS)",313,BLUE,SMALL_FONT,3,2,2,1);
                                                          break;
                                                case 335: pon(tope,313,COPY_PUT);
                                                          texto("CONSULTA DE SALONES GRANDES (MAX. 60
ALUMNOS)",313,BLUE,SMALL_FONT,3,2,2,1);
                                                          break;
                                                case 336: pon(tope,313,COPY_PUT);
                                                          texto("CONSULTA DE SALONES DE CUALQUIER
TIPO",313,BLUE,SMALL_FONT,3,2,2,1);
                                                          break;
                                              }
                                        texto("Escriba el numero de salon que desea consultar o escriba los primeros digitos
seguidos",314,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                                        texto("de un  *  (ej. _*, __*, ___*), o solo un * para consultar todos los salones
existentes.",315,LIGHTBLUE,SMALL_FONT,2,2,2,1);
                                        texto("SALON . . . .",316,BLUE,SMALL_FONT,3,2,2,1);
                                        lee_cadena(cadena,5,317,BLUE,SMALL_FONT,3,2,2,1);
                                              // Lee una cadena del teclado
                                        texto(cadena,317,BLUE,SMALL_FONT,3,2,2,1);
                                        numsalon = atoi(cadena);// Convierte a cadena en un long int
                                        for (k = 314; k <= 317; k++)
                                              {
                                               setfillstyle(SOLID_FILL,GREEN);
                                               bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                              }
                                              quita(tope);          // Quita ventana gráfica 313
                                              quita(tope);          // Quita ventana gráfica 165
                                        b = 0;
                                        if (!strcmp(cadena,"")) then
                                          b = 2;
                                        else
                                         {
                                          for (k=0; cadena[k]; k++)
                                              {
                                               if (!isdigit(cadena[k]))
                                                  b = 1;
                                              }
                                         }
                                         tipo = i;          // Busca el dato según el tipo de cadena
                                         switch(tipo)
                                              {
                                                  case 332: if (b == 0) then
                                                                  busca_salon_tipo_cuenta(r,numsalon,1,&cuenta);
                                                            if (b == 1) then
                                                                busca_salon_cadena_tipo_cuenta(r,numsalon,1,&cuenta);
                                                            if (b == 2) then
                                                                busca_salon_todo_tipo_cuenta(r,1,&cuenta);
                                                            if (b == 3) then
```

```
                                busca_salon_materia_tipo_cuenta(r,1,&cuenta);
                            break;
                    case 333: if (b == 0) then
                                busca_salon_tipo_cuenta(r,numsalon,2,&cuenta);
                            if (b == 1) then
                                busca_salon_cadena_tipo_cuenta(r,numsalon,2,&cuenta);
                            if (b == 2) then
                                busca_salon_todo_tipo_cuenta(r,2,&cuenta);
                            if (b == 3) then
                                busca_salon_materia_tipo_cuenta(r,2,&cuenta);
                            break;
                    case 334: if (b == 0) then
                                busca_salon_tipo_cuenta(r,numsalon,3,&cuenta);
                            if (b == 1) then
                                busca_salon_cadena_tipo_cuenta(r,numsalon,3,&cuenta);
                            if (b == 2) then
                                busca_salon_todo_tipo_cuenta(r,3,&cuenta);
                            if (b == 3) then
                                busca_salon_materia_tipo_cuenta(r,3,&cuenta);
                            break;
                    case 335: if (b == 0) then
                                busca_salon_tipo_cuenta(r,numsalon,4,&cuenta);
                            if (b == 1) then
                                busca_salon_cadena_tipo_cuenta(r,numsalon,4,&cuenta);
                            if (b == 2) then
                                busca_salon_todo_tipo_cuenta(r,4,&cuenta);
                            if (b == 3) then
                                busca_salon_materia_tipo_cuenta(r,4,&cuenta);
                            break;
                    case 336: if (b == 0) then
                                busca_salon_cuenta(r,numsalon,&cuenta);
                            if (b == 1) then
                                busca_salon_cadena_todo_cuenta(r,numsalon,&cuenta);
                            if (b == 2) then
                                busca_salon_todo_cuenta(r,&cuenta);
                            if (b == 3) then
                                busca_salon_materia_todo_cuenta(r,&cuenta);
                            break;
                    }
                                // Asigna espacio en memoria para arreglo
                arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                 if (!arreglo)
                       {
                        printf("Error.");
                        exit(1);
                       }
                            // Despliega la información de Salón en pantalla
                a = 0;
                while (a != -2) do
                      {
                       a = salones_todos(r,numsalon,arreglo,b,tipo,cuenta);
                       if (a >= 0) then
                         consulta_datos_salones_todos(r,arreglo,a,b,semestre);
                      }
                farfree(arreglo);
              }
              quita(tope);      // Quita ventana gráfica 245
              quita(tope);      // Quita ventana gráfica 150
              l = 239;
              pon(tope,l,NOT_PUT);  // Quita ventana gráfica del cursor
              break;

case 240:                       // Opción MATERIA SELECCIONADA
    if (dato != 0) then
        {
```

```
                        while ((*r).num != dato)
                                r = (*r).materias;
                        if ((*r).grupos == NULL) then
                          llama_error(46);
                        else
                        {
                          pon(tope,150,COPY_PUT);          // Pone ventana gráfica 150
                          cuadro(150,BLUE);
                          pon(tope,240,COPY_PUT);          // Pone ventana gráfica 240
                          texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
                          pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
                          texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                          pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                          texto("Seleccione el tipo de salon que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                          pon(tope,332,COPY_PUT);          // Pone ventana gráfica 332
                          pon(tope,333,COPY_PUT);          // Pone ventana gráfica 333
                          pon(tope,334,COPY_PUT);          // Pone ventana gráfica 334
                          pon(tope,335,COPY_PUT);          // Pone ventana gráfica 335
                          pon(tope,336,COPY_PUT);          // Pone ventana gráfica 336
                          texto("DIBUJO",332,BLUE,SMALL_FONT,2,2,2,1);
                          texto("LABORATORIO",333,BLUE,SMALL_FONT,2,2,2,1);
                          texto("CHICO",334,BLUE,SMALL_FONT,2,2,2,1);
                          texto("GRANDE",335,BLUE,SMALL_FONT,2,2,2,1);
                          texto("TODOS",336,BLUE,SMALL_FONT,2,2,2,1);
                          pon(tope,332,NOT_PUT);           // Pone ventana gráfica del cursor
                          i = opciones1(332,332,336,46);  // Mueve la pila de ventanas
                          quita(tope);                     // Quita ventana gráfica del cursor
                          quita(tope);     // Quita ventana gráfica 336
                          quita(tope);     // Quita ventana gráfica 335
                          quita(tope);     // Quita ventana gráfica 334
                          quita(tope);     // Quita ventana gráfica 333
                          quita(tope);     // Quita ventana gráfica 332
                          quita(tope);     // Quita ventana gráfica 165
                          if (i != -1) then
                          {
                                quita(tope);                     // Quita ventana gráfica 246
                                pon(tope,246,COPY_PUT);    // Pone ventana gráfica 246
                                texto("Presione <ENTER> cuando termine de escribir la cadena
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                                                 // Busca el dato según el tipo de la cadena
                                tipo = i;
                                b = 3;
                                switch(tipo)
                                 {
                                        case 332: busca_salon_materia_tipo_cuenta(r,1,&cuenta);
                                                  break;
                                        case 333: busca_salon_materia_tipo_cuenta(r,2,&cuenta);
                                                  break;
                                        case 334: busca_salon_materia_tipo_cuenta(r,3,&cuenta);
                                                  break;
                                        case 335: busca_salon_materia_tipo_cuenta(r,4,&cuenta);
                                                  break;
                                        case 336: busca_salon_materia_todo_cuenta(r,&cuenta);
                                                  break;
                                 }
                                                 // Asigna espacio en memoria para arreglo
                                arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                                if (!arreglo)
                                 {
                                   printf("Error.");
                                   exit(1);
                                 }
                        // Despliega en pantalla los datos de Salones de la lista GRUPO
```

```
                                        a = 0;
                                        while (a != -2) do
                                         {
                                          a = salones_todos(r,numsalon,arreglo,b,tipo,cuenta);
                                          if (a >= 0) then
                                            consulta_datos_salones_todos(r,arreglo,a,b,semestre);
                                         }
                                        farfree(arreglo);
                                    }
                                    quita(tope);      // Quita ventana gráfica 246
                                    quita(tope);      // Quita ventana gráfica 240
                                    quita(tope);      // Quita ventana gráfica 150
                                  }
                                  while (r != NULL)
                                        r = (*r).materias;
                                }
                                else
                                 llama_error(43);

                                i = 240;
                                pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                                break;
                        }
    }
   }
  quita(tope);          // Quita ventana gráfica del cursor
  quita(tope);          // Quita ventana gráfica 100
}

/*    Llama a las funciones que permiten desplegar los datos de Exámen de la lista enlazada
  EXAMEN. */
void consulta_examenes(struct MATERIA *p, int *d, char semestre[10])
{
  struct MATERIA *r;
  struct ARREGLO *arreglo;
  int i,j,c,dato,k,numgrupo,b,a,tipo,cuenta;
  char cadena[10];

  tipo = 0;
  cuenta = 0;
  dato = *d;
  pon(tope,100,COPY_PUT);                     // Pone ventana gráfica 100
  texto("CONSULTA DE EXAMENES",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
  if ((*p).num == 0) then
   {
    llama_error(47);
    quita(tope);                              // Quita ventana gráfica 100
    return;
   }
  ventana_menu_consulta();                    // Pone un submenú de opciones
  i = 237;
  j = 1;
  pon(tope,i,NOT_PUT);                        // Pone ventana gráfica del cursor
  while (j == 1)
   {
    c = bioskey(0);
    switch(c)
     {
          case F1: llama_ayuda(47);           // Llama al módulo AYUDAR
                   break;
          case DERECHA:  quita(tope);// Permite desplazar el cursor sobre el submenú de opciones
                         i++;
                         if (i == 241) then
                            i = 237;
                         pon(tope,i,NOT_PUT);
```

```
                        break;
        case IZQUIERDA: quita(tope);
                        i--;
                        if (i == 286) then
                           i = 240;
                        pon(tope,i,NOT_PUT);
                        break;
        case ENTER: quita(tope);                    // Quita ventana gráfica del cursor
                    r = p;
                    switch(i)
                     {
                          case 237:                             // Opción SALIR
                                    j = 0;
                                    i = 237;
                                    pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                                    break;

                          case 238:                             // Opción IMPRIMIR
                                    if (tipo == 251) then
                                     {
                                      if (cuenta < 1) then
                                        llama_error(52);
                                      else
                                       imprimir(r,arreglo,4,semestre);
                                     }
                                    if (tipo == 252) then
                                     {
                                      if (cuenta < 1) then
                                        llama_error(52);
                                      else
                                       imprimir(r,arreglo,5,semestre);
                                     }
                                    i = 238;
                                    pon(tope,i,NOT_PUT);            // Pone ventana gráfica del cursor
                                    break;

                          case 239:                             // Opción "."
                                    pon(tope,150,COPY_PUT);     // Pone ventana gráfica 150
                                    cuadro(150,BLUE);
                                    pon(tope,246,COPY_PUT);     // Pone ventana gráfica 246
                                    texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,165,COPY_PUT);     // Pone ventana gráfica 165
                                    texto("Seleccione la clase de examen que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,251,COPY_PUT);     // Pone ventana gráfica 251
                                    pon(tope,252,COPY_PUT);     // Pone ventana gráfica 252
                                    texto("EXTRAORDINARIO",251,BLUE,SMALL_FONT,3,2,2,1);
                                    texto("FINAL",252,BLUE,SMALL_FONT,3,2,2,1);
                                    pon(tope,251,NOT_PUT);      // Pone ventana gráfica del cursor
                                    i = opciones1(251,251,252,48); // Mueve la pila de ventanas
                                    tipo = i;
                                    quita(tope);      // Quita ventana gráfica del cursor
                                    quita(tope);      // Quita ventana gráfica 252
                                    quita(tope);      // Quita ventana gráfica 251
                                    quita(tope);      // Quita ventana gráfica 165
                                    if (i != -1) then
                                     {
                                      c = 0;
                                      switch(i)
                                            {
                                              case 251: b = 2;
                                                        break;
                                              case 252: pon(tope,165,COPY_PUT); // Pone ventana gráfica 165
                                                        texto("Se ha seleccionado la opcion
```

```
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                        pon(tope,313,COPY_PUT);    // Pone ventana gráfica 313
                                        texto("CONSULTA DE EXAMEN

FINAL",313,BLUE,SMALL_FONT,3,2,2,1);
                                        texto("Seleccione la forma de busqueda

...",315,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                        pon(tope,316,COPY_PUT);    // Pone ventana gráfica 316
                                        pon(tope,317,COPY_PUT);    // Pone ventana gráfica 317
                                        texto("POR NUMERO DE

GRUPO",316,BLUE,SMALL_FONT,3,2,2,1);
                                        texto("TODOS",317,BLUE,SMALL_FONT,3,2,2,1);
                                        pon(tope,316,NOT_PUT); // Pone ventana gráfica 316
                                        c = opciones1(316,316,317,50);
                                        quita(tope);      // Quita ventana gráfica del cursor
                                        quita(tope);      // Quita ventana gráfica 317
                                        quita(tope);      // Quita ventana gráfica 316
                                        if (c == 316) then
                                         {   // Pone ventana gráfica 246
                                              pon(tope,246,COPY_PUT);
                                              texto("Presione <ENTER> cuando termine de escribir la
cadena ....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                              texto("GRUPO . . . . .",316,BLUE,SMALL_FONT,3,2,2,1);
                                              lee_cadena(cadena,4,317,BLUE,SMALL_FONT,3,2,2,1);
                                              texto(cadena,317,BLUE,SMALL_FONT,3,2,2,1);
                                               // Convierte una cadena en entero
                                             numgrupo = atoi(cadena);
                                              b = 0;
                                              quita(tope);          // Quita ventana gráfica 246
                                         }
                                        if (c == 317) then
                                          b = 2;
                                        for (k = 315; k <= 317; k++)
                                         {
                                              setfillstyle(SOLID_FILL,GREEN);
                                              bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                         }
                                        quita(tope);      // Quita ventana gráfica 313
                                        quita(tope);      // Quita ventana gráfica 165
                                        break;
                                   }
                                   tipo = i; // Busca el dato según el tipo de cadena
                                   switch(tipo)
                                    {
                                        case 251: if (b == 2) then
                                                    busca_examen_todo_cuenta(r,1,&cuenta);
                                                  if (b == 3) then
                                                   busca_examen_materia_tipo_cuenta(r,1,&cuenta);
                                                  break;
                                        case 252: if (b == 0) then
                                                   busca_examen_final_cuenta(r, numgrupo,&cuenta);
                                                  if (b == 2) then
                                                   busca_examen_todo_cuenta(r,2,&cuenta);
                                                  if (b == 3) then
                                                   busca_examen_materia_tipo_cuenta(r, 2,&cuenta);
                                                  break;
                                    }
                                          // Asigna espacio en memoria para arreglo
                              arreglo = (struct ARREGLO *) farcalloc(cuenta+1,sizeof(struct ARREGLO));
                               if (!arreglo)
                                    {
                                     printf("Error.");
                                     exit(1);
                                    }
                                          // Despliega los datos de examen de la lista EXAMEN
                              if (c != -1) then
```

```
                                        {
                                            a = 0;
                                            while (a != -2) do
                                             {
                                              a = examen_todos(r,numgrupo,arreglo,b,tipo,cuenta);
                                              if (a >= 0) then
                                               {
                                                    switch(i)
                                                     {
                                                       case 251: consulta_datos_examen_extraord(r,
arreglo,a,b,semestre);
                                                                 break;
                                                       case 252: consulta_datos_examen_final(r,
arreglo,a,b,semestre);
                                                                 break;
                                                     }
                                               }
                                             }
                                           farfree(arreglo);          // Libera el espacio asignado a arreglo
                                        }
                                        quita(tope);        // Quita ventana gráfica 246
                                        quita(tope);        // Quita ventana gráfica 150
                                        i = 239;
                                        pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                                        break;
                                case 240:           // Opción MATERIA SELECCIONADA
                                      if (dato != 0) then
                                      {
                                       while ((*r).num != dato)
                                             r = (*r).materias;
                                       if ((*r).examen == NULL) then
                                        llama_error(51);
                                       else
                                        {
                                         pon(tope,150,COPY_PUT);          // Pone ventana gráfica 150
                                         cuadro(150,BLUE);
                                         pon(tope,240,COPY_PUT);          // Pone ventana gráfica 240
                                         texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
                                         pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
                                         texto("Presione <ENTER> para seleccionar o <ESC> para salir
....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                         pon(tope,165,COPY_PUT);          // Pone ventana gráfica 165
                                         texto("Seleccione la clase de examen que desea consultar
...",165,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                         pon(tope,251,COPY_PUT);          // Pone ventana gráfica 251
                                         pon(tope,252,COPY_PUT);          // Pone ventana gráfica 252
                                         texto("EXTRAORDINARIO",251,BLUE,SMALL_FONT,3,2,2,1);
                                         texto("FINAL",252,BLUE,SMALL_FONT,3,2,2,1);
                                         pon(tope,251,NOT_PUT); // Pone ventana gráfica del cursor
                                         i = opciones1(251,251,252,49);
                                         quita(tope);      // Quita ventana gráfica del cursor
                                         quita(tope);      // Quita ventana gráfica 252
                                         quita(tope);      // Quita ventana gráfica 251
                                         quita(tope);      // Quita ventana gráfica 165
                                         if (i != -1) then
                                          {
                                              quita(tope);          // Quita ventana gráfica 246
                                              pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
                                              texto("Presione <ENTER> cuando termine de escribir la
cadena....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                                        // Busca el dato según el tipo de cadena
                                              tipo = i;
                                              b = 3;
                                              switch(tipo)
```

```
                                        {
                                                case 251: busca_examen_materia_tipo_cuenta(r, 1,&cuenta);
                                                        break;
                                                case 252: busca_examen_materia_tipo_cuenta(r, 2,&cuenta);
                                                        break;
                                        }
                                // Despliega en pantalla los datos de exámen de la lista de EXAMEN
                                        a = 0;
                                        while (a != -2) do
                                        {
                                          a = examen_todos(r,numgrupo,arreglo,b,tipo,cuenta);
                                          if (a >= 0) then
                                          {
                                                switch(l)
                                                {
                                                  case 251: consulta_datos_examen_extraord(r,
arreglo,a,b,semestre);
                                                        break;
                                                  case 252: consulta_datos_examen_final(r, arreglo,a,b,semestre);
                                                        break;
                                                }
                                          }
                                        }
                                  }
                                  quita(tope);      // Quita ventana gráfica 248
                                  quita(tope);      // Quita ventana gráfica 240
                                  quita(tope);      // Quita ventana gráfica 150
                                 }
                                 while (r != NULL)
                                        r = (*r).materias;
                                }
                                else
                                 llama_error(43);
                                i = 240;
                                pon(tope,l,NOT_PUT);        // Pone ventana gráfica del cursor
                                break;
                        }
    }
   }
  quita(tope);                                      // Quita ventana gráfica del cursor
  quita(tope);                                      // Quita ventana gráfica 100
}

//    Llama a las funciones según el valor de l.
int consulta(int l, struct MATERIA *lista, int *d, char semestre[10])
{
 int dato;

 dato = *d;
 cuadro(61,GREEN);
 switch(l)
   {
   case 38: dato = consulta_materias(lista,semestre);       // Permite desplegar en pantalla los datos de Materias
            break;
   case 39: consulta_grupos(lista,&dato,semestre);          // Permite desplegar en pantalla los datos de Grupos
            break;
   case 40: consulta_profesores(lista,&dato,semestre);      // Permite desplegar en pantalla los datos de Profesores
            break;
   case 41: consulta_salones(lista,&dato,semestre);         // Permite desplegar en pantalla los datos de Salones
            break;
   case 42: consulta_examenes(lista,&dato,semestre);        // Permite desplegar en pantalla los datos de Exámen
   }
 cuadro(61,LIGHTBLUE);
 return(dato);
}
```

B.3.4. CUADROS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\grafico.h"
#include "d:\cuadros.h"
#include "d:\ayuda.h"

//    Construye una ventana i rellena con color
void cuadro(int i, int color)
{
 setfillstyle(SOLID_FILL,color);
 bar(vent[i].x1,vent[i].y1,vent[i].x2,vent[i].y2);
 setcolor(WHITE);
 rectangle(vent[i].x1,vent[i].y1,vent[i].x2,vent[i].y2);
}

//    Centra un texto dentro de una ventana n, el font esta definido por tipo.
void cuadrito(int n,int color,int tipo,int x1,int x2,int y1,int y2,char *p)
{
 int        x, y, i, j;

 setfillstyle(SOLID_FILL,color);
 bar(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
 setcolor(WHITE);
 rectangle(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
 settextstyle(tipo,HORIZ_DIR,USER_CHAR_SIZE);
 setusercharsize(x1,x2,y1,y2);
 y = textheight(p);
 x = textwidth(p);
 i = vent[n].x1  + div(vent[n].x2 - vent[n].x1 - x,2).quot;
 j = vent[n].y1  + div(vent[n].y2 - vent[n].y1 - y,2).quot;
 outtextxy(i,j,p);
}

/*      Selecciona la opción en ventana.  Regresa el valor de la  ventana i ó regresa -1 si no se
  quiso seleccionar. */
int opciones1(int i, int min, int max, int num)
{
 int        c;

// Construye los cuadros de las ventanas
 c = DERECHA;
 while (c != ENTER) do
   {
    c = bioskey(0);
    switch(c)
          {                     // Desplaza el cursor de derecha a izquierda y viceversa
           case DERECHA: quita(tope);
                         i++;
                         if (i == (max+1)) then
                               i = min;
                         pon(tope,i,NOT_PUT);
                         break;
```

```
            case IZQUIERDA:  quita(tope);
                             i--;
                             if (i == (min-1)) then
                                 i = max;
                             pon(tope,i,NOT_PUT);
                             break;
            case ESC:    return(-1);
            case F1:     llama_ayuda(num);
                             break;
            }
    }
 return(i);
}


//      Selecciona la opción en ventana. Regresa el valor de la ventana i.
int opciones2(int i, int min, int max)
{
 int        c;

 // Construye los cuadros de las ventanas
 c = DERECHA;
 while (c != ENTER) do
    {
    c = bioskey(0);
    switch(c)
            {                       // Desplaza el cursor en forma vertical
            case ABAJO: quita(tope);
                             i++;
                             if (i == (max+1)) then
                                 i = min;
                             pon(tope,i,NOT_PUT);
                             break;
            case ARRIBA: quita(tope);
                             i--;
                             if (i == (min-1)) then
                                 i = max;
                             pon(tope,i,NOT_PUT);
                             break;
            }
    }
 return(i);
}

//      Escribe en la ventana de entrada uno de los caracteres siguientes: '!', '|', ']', 'i', 'i'
void letra_especial(int x, int y, int largo, char *p)
{
 int        x0, ancho;

 ancho = textwidth(p);
 x0 = div(largo - ancho,2).quot;
 outtextxy(x+x0,y,p);
}

//      Escribe un cadena en una ventana n. El tipo de font esta definido por tipo.
void texto(char *p, int n, int color, int tipo,int x1, int x2, int y1, int y2)
{
 int        x, y, i, j;

 setfillstyle(SOLID_FILL,color);
 bar(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
 setcolor(WHITE);
 rectangle(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
 settextstyle(tipo,HORIZ_DIR,USER_CHAR_SIZE);
 setusercharsize(x1,x2,y1,y2);
```

```
y = (unsigned int)(1.2*textheight(p));
x = textwidth(p);
i = vent[n].x1 + div(vent[n].x2 - vent[n].x1 - x,2).quot;
j = vent[n].y1 + div(vent[n].y2 - vent[n].y1 - y,2).quot;
outtextxy(i,j,p);
}

/*      Lee una cadena de caracteres desde el teclado. La cadena es puesta en p, cuya máxima longitud es max y
  sólo puede tener caracteres de los que existen en el arreglo letras. La lectura se hecha en la ventana número  v.
  Color es el color empleado en el cuadro. La variables tipo corresponde al font empleado, mientras que las varia-
  bles x1, x2, y1 y y2 son compatibles con la función setusercharsize. */
void lee_cadena(char *p, int max, int v, int color, int tipo,int x1, int x2, int y1, int y2)
{
int     alto, largo, x, y, n, bit, c, w[50], i, j, y0;
char    nombre[2], *r, s;
                // Hace el cuadro donde se va a trabajar
cuadro(v,color);
settextstyle(tipo,HORIZ_DIR,USER_CHAR_SIZE);
setusercharsize(x1,x2,y1,y2);
                // Calcula el tamaño del cursor; sus dimensiones: alto y ancho
nombre[0] = 'W';
nombre[1] = '\0';
largo = textwidth(nombre);
alto = textheight(nombre);
                // Posición inicial del cursor
x = vent[v].x1 + div(vent[v].x2 - vent[v].x1 + 1 - max*largo,2).quot;
y = vent[v].y1 + div(vent[v].y2 - vent[v].y1 + 1 - alto,2).quot;
                // Construye las posiciones de cada carácter en la cadena (pixeles)
for (i = 0; i < 50; i++)
   w[i] = x + i*largo;
largo = (unsigned int) (0.85*textwidth(nombre));
y0 = (unsigned int) (0.8*textheight(nombre));
                // Establece el modo de llenado del cursor y la posición inicial del cursor
setfillstyle(SOLID_FILL,WHITE);
bar(x,y+y0,x+largo,y+alto);
                // Lee la cadena de caracteres en modo gráfico
p[0] = '\0';
nombre[1] = '\0';
i = 0;
bit = 1;
while (bit) do
  {
  c = bioskey(0);
  switch(c)
    {
    case ENTER:  bit = 0;
                 break;
    case DERECHA:   if (i < max) then
                          {
                                // Si no hay carácter a la derecha no se mueve
                          n = strlen(p);
                          if (i >= n) then
                            break;
                          // Quita el cursor de la posicion i
                          setfillstyle(SOLID_FILL,color);
                          bar(w[i],y+y0,w[i]+largo,y+alto);
                          // Pone el carácter que había en la posición i
                          setcolor(WHITE);
                          nombre[0] = p[i];
                          letra_especial(w[i],y,largo,nombre);
                          // Quita el carácter en la posición i+1
                          setcolor(color);
                          nombre[0] = p[i+1];
                          letra_especial(w[i+1],y,largo,nombre);
                          // Pone el cursor en la posición i+1
```

```
                        setfillstyle(SOLID_FILL,WHITE);
                        bar(w[i+1],y+y0,w[i+1]+largo,y+alto);
                        // Pone el carácter en la posición i+1
                        setcolor(WHITE);
                        nombre[0] = p[i+1];
                        letra_especial(w[i+1],y,largo,nombre);
                        // Incrementa el apuntador
                        i++;
                        }
                      break;

case IZQUIERDA:       if (i > 0) then
                        {
                        // Quita el cursor de la posicion i
                        setfillstyle(SOLID_FILL,color);
                        bar(w[i],y+y0,w[i]+largo,y+alto);
                        // Quita el carácter de la posición i-1
                        setcolor(color);
                        nombre[0] = p[i-1];
                        letra_especial(w[i-1],y,largo,nombre);
                        // Pone el cursor en la posición i-1
                        setfillstyle(SOLID_FILL,WHITE);
                        bar(w[i-1],y+y0,w[i-1]+largo,y+alto);
                        // Pone el carácter en la posición i-1
                        setcolor(WHITE);
                        nombre[0] = p[i-1];
                        letra_especial(w[i-1],y,largo,nombre);
                        // Si hay carácter en la posición i lo pone
                        n = strlen(p) - 1;
                        if (i <= n) then
                          {
                          setcolor(WHITE);
                          nombre[0] = p[i];
                          letra_especial(w[i],y,largo,nombre);
                          }
                        // Decrementa el apuntador
                        i--;
                        }
                      break;

case HOME:      if (i < 0) then
                        break;

                      // Quita el cursor de la posición i
                      setfillstyle(SOLID_FILL,color);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      // Pone el carácter de la posición i
                      setcolor(WHITE);
                      nombre[0] = p[i];
                      letra_especial(w[i],y,largo,nombre);
                      // Pone el cursor en la posición i = 0
                      i = 0;
                      setfillstyle(SOLID_FILL,WHITE);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      break;

case END:       n = strlen(p);
                      if (i >= n) then
                        break;
                      // Quita el cursor de la posición i
                      setfillstyle(SOLID_FILL,color);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      // Pone la letra de la posición i
                      setcolor(WHITE);
                      nombre[0] = p[i];
```

```
                    letra_especial(w[i],y,largo,nombre);
                    // Pone el cursor en fin de cadena
                    i = n;
                    setfillstyle(SOLID_FILL,WHITE);
                    bar(w[i],y+y0,w[i]+largo,y+alto);
                    break;

case DELETE:        if ((i < max) && (i >= 0)) then
                      {
                      // Si no hay caracteres a la derecha de i sale
                      n = strlen(p);
                      if (i >= n) then
                        break;
                      // Quita el carácter i en la posición cursor
                      setfillstyle(SOLID_FILL,color);
                      bar(w[i],y,w[i]+largo,vert[v].y2-1);
                      // Quita el carácter i y corre los demás
                      j = i;
                      while (p[j] != '\0')
                        {
                        p[j] = p[j+1];
                        j++;
                        }
                      p[j] = '\0';
                      // Corre caracteres a la derecha de i en pantalla
                      j = i;
                      while (p[j] != '\0')
                        {
                        // Quita el carácter j
                        setcolor(color);
                        nombre[0] = p[j];
                        letra_especial(w[j+1],y,largo,nombre);
                        // Pone el nuevo carácter
                        setcolor(WHITE);
                        nombre[0] = p[j+1];
                        letra_especial(w[j+1],y,largo,nombre);
                        j++;
                        }
                      // Pone el cursor en i y el carácter
                      setfillstyle(SOLID_FILL,WHITE);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      setcolor(WHITE);
                      nombre[0] = p[i];
                      letra_especial(w[i],y,largo,nombre);
                      }
                    break;

case BACKSPACE:     if (i > 0) then
                      {
                      // Borra el carácter i-1 y corre los demás
                      setcolor(color);
                      nombre[0] = p[i-1];
                      letra_especial(w[i-1],y,largo,nombre);
                      j = i-1;
                      while (p[j] != '\0')
                        {
                        p[j] = p[j+1];
                        j++;
                        }
                      // Quita el cursor de su posición inicial
                      setcolor(color);
                      setfillstyle(SOLID_FILL,color);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      // Quita el carácter en la posición del cursor
                      setcolor(color);
```

```
                        nombre[0] = p[i-1];
                        letra_especial(w[i],y,largo,nombre);
                        // Corre todos los carácter a partir de i
                        j = i;
                        while (p[j] != '\0')
                          {
                          nombre[0] = p[j];
                          setcolor(color);
                          letra_especial(w[j+1],y,largo,nombre);
                          setcolor(WHITE);
                          letra_especial(w[j],y,largo,nombre);
                          j++;
                          }
                        // Decrementa el apuntador i
                        i--;
                        // Pone el cursor una posición atrás
                        setfillstyle(SOLID_FILL,WHITE);
                        bar(w[i],y+y0,w[i]+largo,y+alto);
                        // Pone el carácter una posición atrás
                        setcolor(WHITE);
                        nombre[0] = p[i];
                        letra_especial(w[i],y,largo,nombre);
                        }
                      break;
                      // Escribe cualquier carácter en la cadena p si es aceptado
    default: if (isprint(bajo(c)) && (i < max) && (i >= 0)) then
                      {
                      // Quita el cursor anterior
                      setcolor(color);
                      setfillstyle(SOLID_FILL,color);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      // Pone el nuevo carácter
                      setcolor(color);
                      nombre[0] = p[i];
                      letra_especial(w[i],y,largo,nombre);
                      if (i == strlen(p)) then
                        {
                        p[i] = bajo(c);
                        p[i+1] = '\0';
                        }
                      else
                        p[i] = bajo(c);
                      nombre[0] = bajo(c);
                      setcolor(WHITE);
                      letra_especial(w[i],y,largo,nombre);
                      // Incrementa el apuntador
                      i++;
                      // Pone el cursor en la posición siguiente
                      setfillstyle(SOLID_FILL,WHITE);
                      bar(w[i],y+y0,w[i]+largo,y+alto);
                      }
                    break;
  }
 }
}
```

B.3.5. ERRORES.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\grupos.h"
#include "d:\ayuda.h"
#define  then


/*    Verifica que la cadena del dato grupo tenga cuatro dígitos, de lo contrario aumenta ceros a
  la izquierda. Devuelve la cadena correcta.*/
void grupo_correcto(char cadena[100])
{
 char numero[100];
 int k;

 for (k=0; cadena[k]; k++);
 switch(k)
   {
   case 1: strcpy(numero,"000");
           strcat(numero,cadena);
           break;
   case 2: strcpy(numero,"00");
           strcat(numero,cadena);
           break;
   case 3: strcpy(numero,"0");
           strcat(numero,cadena);
           break;
   case 4: strcpy(numero,cadena);
           break;
   }
  strcpy(cadena,numero);
}


//    Cuadros del menú de Error
void ventana_error(char cadena[20][100], int inicio, int fin)
{
int x1, y1, x2, y2, x, y, i, j, m, k, n, mul;

x1 = 1;
x2 = 1;
y1 = 3;
y2 = 2;
n = 384;
setfillstyle(SOLID_FILL,LIGHTBLUE);
bar(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
setcolor(WHITE);
rectangle(vent[n].x1,vent[n].y1,vent[n].x2,vent[n].y2);
```

```
 settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
 setusercharsize(x1,x2,y1,y2);
 m = div(vent[n].y2 - vent[n].y1,4).quot;
 for (k=inicio, mul=0;  k <= fin;  k++, mul++)
  {
   y = (unsigned int)(1.2*textheight(cadena[k]));
   x = textwidth(cadena[k]);
   i = vent[n].x1 + div(vent[n].x2 - vent[n].x1 - x,2).quot;
   j = (vent[n].y1  + div(m - y,2).quot) + m*mul;
   outtextxy(i,j,cadena[k]);
  }
}


/*    Construye la ventana del mensaje de error. El mensaje seleccionado será el valor que tome
  num.*/
void llama_error(int num)
{
 char cadena[20][100], cad[100], sig[100];
 int c,quot,rem,ultimo,j,grupo,inicio,fin,k,a;
 struct POS
     {
         int inicio;
         int fin;
     }pos[15];
                                                        // Mensajes de error
 switch(num)
  {
   case 4: strcpy(cadena[0],"   Para escribir la información de un archivo en disco, tiene");
           strcpy(cadena[1],"que leer un archivo que ya exista ó crear una nueva  lista  de");
           strcpy(cadena[2],"materias.                                                    ");
           strcpy(cadena[3],"");
           break;

   case 5: strcpy(cadena[0],"                                                             ");
           strcpy(cadena[1],"   Para agregar una nueva materia a la lista, tiene que leer");
           strcpy(cadena[2],"un archivo ó crear una nueva lista de materias.              ");
           strcpy(cadena[3],"");
           break;

   case 6: strcpy(cadena[0],"                                                             ");
           strcpy(cadena[1],"   Para eliminar una materia de la lista, tiene que leer");
           strcpy(cadena[2],"un archivo ó crear una nueva lista de materias.          ");
           strcpy(cadena[3],"");
           break;

   case 7: strcpy(cadena[0],"                                                             ");
           strcpy(cadena[1],"   Para cambiar datos de una materia de la lista, tiene que");
           strcpy(cadena[2],"leer un archivo ó crear una nueva lista de materias.       ");
           strcpy(cadena[3],"");
           break;

   case 10: strcpy(cadena[0],"                                                            ");
           strcpy(cadena[1],"   El dato que escribió no se encuentra en la lista.");
           strcpy(cadena[2],"");
           break;

   case 15:strcpy(cadena[0],"   El directorio que escribió es incorrecto. Vuelva a escribir");
           strcpy(cadena[1],"el directorio seleccionando de entre los siguientes drives:   ");
           strcpy(cadena[2],"A:\\, B:\\, C:\\, D:\\, E:\\ y F:\\.                          ");
           strcpy(cadena[3],"");
           break;

   case 18:strcpy(cadena[0],"                                                             ");
           strcpy(cadena[1],"   No se encuentra ningún salón para la cadena que escribió.");
```

```
        strcpy(cadena[2],"");
        break;

case 17:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  No se encuentra ningún profesor para la cadena que escribió.");
        strcpy(cadena[2],"");
        break;

case 18:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  El dato que escribió contiene carácteres alfabéticos. Sólo");
        strcpy(cadena[2],"puede escribir carácteres numéricos.                ");
        strcpy(cadena[3],"");
        break;

case 19:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  El nombre de materia que escribió se encuentra incluido");
        strcpy(cadena[2],"en la lista de materias. Escriba un nuevo nombre.   ");
        strcpy(cadena[3],"");
        break;

case 20:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  El número de clave que escribió pertenece a una materia");
        strcpy(cadena[2],"incluida en la lista. Escriba un nuevo número de clave.  ");
        strcpy(cadena[3],"");
        break;

case 21:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Para crear una lista de grupos es necesario leer un archivo");
        strcpy(cadena[2],"o crear una nueva lista de materias.                ");
        strcpy(cadena[3],"");
        break;

case 22:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Para agregar un nuevo grupo es necesario leer un archivo");
        strcpy(cadena[2],"o crear una nueva lista de materias.                ");
        strcpy(cadena[3],"");
        break;

case 23:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Para eliminar un grupo es necesario leer un archivo");
        strcpy(cadena[2],"o crear una nueva lista de materias.                ");
        strcpy(cadena[3],"");
        break;

case 24:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Para cambiar los datos de grupos es necesario leer un archivo");
        strcpy(cadena[2],"o crear una nueva lista de materias.                ");
        strcpy(cadena[3],"");
        break;

case 25:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Antes de agregar un grupo, tiene que crear una nueva lista");
        strcpy(cadena[2],"de grupos.                                          ");
        strcpy(cadena[3],"");
        break;

case 26:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Antes de eliminar un grupo, tiene que crear una nueva lista");
        strcpy(cadena[2],"de grupos.                                          ");
        strcpy(cadena[3],"");
        break;

case 27:strcpy(cadena[0],"                                                    ");
        strcpy(cadena[1],"  Antes de cambiar un grupo, tiene que crear una nueva lista");
        strcpy(cadena[2],"de grupos.                                          ");
```

```
            strcpy(cadena[3],"");
            break;

case 28:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   El número de grupo que escribió ya existe para la materia");
            strcpy(cadena[2],"seleccionada. Escriba un nuevo número de grupo.          ");
            strcpy(cadena[3],"");
            break;

case 29:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   No se encuentra ningún grupo para la cadena que escribió.");
            strcpy(cadena[2],"");
            break;

case 30:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Para crear una lista de exámenes es necesario leer un archivo");
            strcpy(cadena[2]," ya existente o crear una nueva lista de materias.          ");
            strcpy(cadena[3],"");
            break;

case 31:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Para agregar un nuevo exámen es necesario leer un archivo");
            strcpy(cadena[2],"ya existente o crear una nueva lista de materias.          ");
            strcpy(cadena[3],"");
            break;

case 32:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Para eliminar un exámen es necesario leer un archivo ya");
            strcpy(cadena[2],"existente o crear una nueva lista de materias.          ");
            strcpy(cadena[3],"");
            break;

case 33:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Para cambiar los datos de exámen es necesario leer un archivo");
            strcpy(cadena[2],"ya existente o crear una nueva lista de materias.          ");
            strcpy(cadena[3],"");
            break;

case 34:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de agregar un exámen, tiene que crear una nueva lista");
            strcpy(cadena[2],"de exámenes.                                          ");
            strcpy(cadena[3],"");
            break;

case 35:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de eliminar un exámen, tiene que crear una nueva lista");
            strcpy(cadena[2],"de exámenes.                                          ");
            strcpy(cadena[3],"");
            break;

case 36:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de cambiar un exámen, tiene que crear una nueva lista");
            strcpy(cadena[2],"de exámenes.                                          ");
            strcpy(cadena[3],"");
            break;

case 37:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   El número de grupo que escribió ya existe en la lista de exámenes");
            strcpy(cadena[2],"de la materia seleccionada.  Escriba un nuevo número de grupo.      ");
            strcpy(cadena[3],"");
            break;

case 38:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   El día que escribió es incorrecto. Vuelva a escribir el dato");
            strcpy(cadena[2],"seleccionando de entre el día 1 al 31.                ");
```

```
            strcpy(cadena[3],"");
            break;

case 39:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   El mes que escribió es incorrecto. Vuelva a escribir el dato");
            strcpy(cadena[2],"seleccionando de entre el mes 1 al 12.                    ");
            strcpy(cadena[3],"");
            break;

case 40:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   El año que escribió es incorrecto. Vuelva a escribir el dato");
            strcpy(cadena[2],"seleccionando apartir del año de 1900.                    ");
            strcpy(cadena[3],"");
            break;

case 41:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   La hora que escribió es incorrecta. Vuelva a escribirla con");
            strcpy(cadena[2],"el siguiente formato:   hh:mm                    ");
            strcpy(cadena[3],"");
            break;

case 42:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   La materia que consultó no tiene exámenes del tipo seleccionado.");
            strcpy(cadena[2],"");
            break;

case 43:strcpy(cadena[0],"   Para consultar los datos de una MATERIA SELECCIONADA, es");
            strcpy(cadena[1],"necesario entrar al módulo de consultar materias,  seleccionar");
            strcpy(cadena[2],"alguna, salir y entrar al módulo del dato de que desea obtener");
            strcpy(cadena[3],"la información. Si no seleccionó ninguna materia en el  módulo");
            strcpy(cadena[4],"de consulta de materias, no podrá consultar en este módulo.  ");
            strcpy(cadena[5],"");
            break;

case 44:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de consultar los datos de grupos es necesario leer un");
            strcpy(cadena[2],"archivo ya existente o crear uno nuevo.                    ");
            strcpy(cadena[3],"");
            break;

case 45:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de consultar los datos de profesores es necesario leer un");
            strcpy(cadena[2],"archivo ya existente o crear uno nuevo.                    ");
            strcpy(cadena[3],"");
            break;

case 46:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de consultar los datos de salones es necesario leer un");
            strcpy(cadena[2],"archivo ya existente o crear uno nuevo.                    ");
            strcpy(cadena[3],"");
            break;

case 47:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de consultar los datos de exámenes es necesario leer un");
            strcpy(cadena[2],"archivo ya existente o crear uno nuevo.                    ");
            strcpy(cadena[3],"");
            break;

case 48:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   La materia que seleccionó no tiene grupos. ");
            strcpy(cadena[2],"");
            break;

case 49:strcpy(cadena[0],"                                                    ");
            strcpy(cadena[1],"   Antes de consultar los datos de materias es necesario leer un");
```

```
            strcpy(cadena[2],"archivo ya existente o crear uno nuevo.                    ");
            strcpy(cadena[3],"");
            break;

  case 50:strcpy(cadena[0],"                                                            ");
            strcpy(cadena[1],"  No se encuentra ningún examen del tipo que seleccionó.");
            strcpy(cadena[2],"");
            break;

  case 51:strcpy(cadena[0],"                                                            ");
            strcpy(cadena[1],"  La materia que seleccionó no tiene exámenes.           ");
            strcpy(cadena[2],"");
            break;

  case 52:strcpy(cadena[0],"                                                            ");
            strcpy(cadena[1],"  Para imprimir es necesario consultar. La última consulta que");
            strcpy(cadena[2],"realize es la que podrá enviar a impresión.             ");
            strcpy(cadena[3],"");
            break;

  case 53:strcpy(cadena[0],"                                                            ");
            strcpy(cadena[1],"  No se puede accesar ó no se encuentra el directorio AYUDA");
            strcpy(cadena[2],"");
            break;

  case 54:strcpy(cadena[0],"  No es posible realizar la impresión, esto se debe a errores");
            strcpy(cadena[1],"de Entrada/Salida ó por falta de papel. Verifique su impresora,");
            strcpy(cadena[2],"la cuál tiene que estar conectada al puerto LPT1.       ");
            strcpy(cadena[3],"");
            break;
}
ultimo = 0;
for (k=0; stromp(cadena[k],""); k++)
  ultimo = ultimo + 1;
quot = ultimo/4;
rem = ultimo%4;
if (rem != 0) then
  grupo = quot + 1;
else
  grupo = quot;
k = 0;
for (j=1; j <= grupo; j++)
  {
  a = 0;
  while ((stromp(cadena[k],"")) && (a < 4)) do
    {
        fin = k;
        if (a == 0) then
          inicio = fin;
        a = a + 1;
        k = k + 1;
    }
  pos[].inicio = inicio;
  pos[].fin = fin;
  }
j = 1;
inicio = pos[].inicio;
fin = pos[].fin;
pon(tope,382,COPY_PUT);                          // Pon ventana gráfica 382
cuadro_texto(" ",382,MAGENTA,SMALL_FONT,1,1,1,1); // Construye el cuadro mayor
setfillstyle(SOLID_FILL,MAGENTA);                // Establece el tipo y color del relleno
rectangle(102,219,538,341);                      // Crea dos rectangulos con diferentes coordenadas.
rectangle(104,221,536,339);
cuadro_texto(" ERROR ",382,MAGENTA,COMPLEX_FONT,1,2,1,2);
ventana_error(cadena,inicio,fin);                // Pone el texto dentro de la ventana de error
```

```
  cuadro_texto("Presione <ESC> para salir . . .",365,DARKGRAY,TRIPLEX_SCR_FONT,1,2,1,2);
  if (grupo > 1) then
    {
     strcpy(cad," PAG. 1 ");
     cuadro_texto(cad,366,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
     cuadro_texto("PgUp/PgDn",367,GREEN,SIMPLEX_FONT,2,5,1,2);
     c = END;
     while (c != ESC) do
          {
           c = bioskey(0);
           switch(c)
            {
                  case PGDN: j++;                          // Cambia a la página siguiente
                             if (j == (grupo + 1)) then
                                j = 1;
                             inicio = pos[j].inicio;
                             fin = pos[j].fin;
                             ventana_error(cadena,inicio,fin);
                             itoa(j,sig,10);
                             strcpy(cad,"PAG. ");
                             strcat(cad,sig);
                             cuadro_texto(cad,366,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
                             break;

                  case PGUP: j--;                          // Cambia a la página anterior
                             if (j == 0) then
                                j = grupo;
                             inicio = pos[j].inicio;
                             fin = pos[j].fin;
                             ventana_error(cadena,inicio,fin);
                             itoa(j,sig,10);
                             strcpy(cad,"PAG. ");
                             strcat(cad,sig);
                             cuadro_texto(cad,366,MAGENTA,SANS_SERIF_FONT,2,5,2,5);
                             break;
            }
          }
    }
  else
    {
     c = END;
     while (c != ESC) do
          c = bioskey(0);
    }
  quita(tope);                                             // Quita ventana gráfica 382
}


// Verifica que cadena sea un número entero. Regresa 1 si hubo error de lo contrario regresa 0.
int error_numero_entero(char cadena[100])
 {
  int valor,k;

  valor = 0;
  for (k=0; cadena[k]; k++)
   {
    if (!isdigit(cadena[k]) == 0) then
       valor = 1;
   }
  if (valor == 1) then
     llama_error(16);
  return(valor);
}
```

```
//  Verifica que cadena sea un entero dentro del rango de 1 a 31. Regresa 1 si hubo error de lo contrario regresa 0.
int error_dia(char cadena[100])
{
 int n, k, valor;

 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
         valor = 1;
  }
 k = atoi(cadena);
 if ((k < 1) || (k > 31)) then
  {
   valor = 1;
   n = 38;
  }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}

//  Verifica que cadena sea un entero dentro del rango de 1 a 12. Regresa 1 si hubo error de lo contrario regresa 0.
int error_mes(char cadena[100])
{
 int n, k, valor;

 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
         valor = 1;
  }
 k = atoi(cadena);
 if ((k < 1) || (k > 12)) then
  {
   valor = 1;
   n = 39;
  }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}

//  Verifica que cadena sea un entero mayor a 1899.  Regresa 1 si hubo error de lo contrario regresa 0.
int error_ano(char cadena[100])
{
 int n, k, valor;

 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
         valor = 1;
  }
 k = atoi(cadena);
 if (k < 1900) then
  {
   valor = 1;
   n = 40;
  }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}
```

```
/*  Verifica que cadena1 tenga un formato correcto para el horario. Regresa la cadena correcta
    en cadena2. Si el formato es erroneo, envia un mensaje de error.*/
int hora_correcta(char cadena1[100], char cadena2[100], int num)
{
int hora1,minuto1,hora2,minuto2,a,k,c,l,m;
char cadena[100],min1[100],min2[100],numero[100];

l = 0;
m = 0;                                          // Verifica la hora1 correcta
 hora1=atoi(cadena1);
 if ((hora1 <= 0) || (hora1 > 24)) then
  {
   llama_error(41);
   return(1);
  }
 min1[0] = '\0';                                // Verifica los minutos1 correctos
 a = 0;
 for (k=0; cadena1[k]; k++)
   {
    if (!isdigit(cadena1[k]))
     {
          a = k + 1;
          m = k;
     }
   }
 if (cadena1[m] != ':') then
   {
    llama_error(41);
    return(1);
   }
 if (a > 0)
   {
    for (c=0; cadena1[a]; a++,c++)
          min1[c] = cadena1[a];
   }
 min1[c] = '\0';
 minuto1=atoi(min1);
 if ((minuto1 < 0) || (minuto1 > 60)) then
  {
   llama_error(41);
   return(1);
  }
 itoa(hora1,numero,10);
 if (hora1 < 10) then
  {
   strcpy(cadena,"0");
   strcat(cadena,numero);
  }
 else
    strcpy(cadena,numero);
 strcat(cadena,":");
 itoa(minuto1,numero,10);
 if (minuto1 < 10) then
   {
    strcat(cadena,"0");
    strcat(cadena,numero);
   }
 else
   strcat(cadena,numero);
 strcpy(cadena1,cadena);
 if (num == 2) then                                     // Verifica la hora2 correcta
  {
    m = 0;
     hora2=atoi(cadena2);
    if ((hora2 <= 0) || (hora2 > 24)) then
```

```
    {
        llama_error(41);
        return(1);
    }
    min2[0] = '\0';                                    // Verifica los minutos2 correctos
    a = 0;
    for (k=0; cadena2[k]; k++)
        {
         if (!isdigit(cadena2[k]))
          {
           a = k + 1;
           m = k;
          }
        }
    if (cadena2[m] != ':') then
        {
         llama_error(41);
         return(1);
        }
    if (a > 0)
        {
         for (c=0; cadena2[a]; a++,c++)
           min2[c] = cadena2[a];
        }
    min2[c] = '\0';
    minuto2=atoi(min2);
    if ((minuto2 < 0) || (minuto2 > 60)) then
    {
        llama_error(41);
        return(1);
    }
    if (hora1 > hora2) then
    {
         llama_error(41);
         return(1);
    }
    if ((hora1 >=hora2) && (minuto1 >= minuto2)) then
        {
         llama_error(41);
         return(1);
        }
    itoa(hora2,numero,10);
    if (hora2 < 10) then
        {
         strcpy(cadena,"0");
         strcat(cadena,numero);
        }
    else
         strcpy(cadena,numero);
    strcat(cadena,":");
    itoa(minuto2,numero,10);
    if (minuto2 < 10) then
        {
         strcat(cadena,"0");
         strcat(cadena,numero);
        }
    else
         strcat(cadena,numero);
    strcpy(cadena2,cadena);
  }
 return(i);
}
```

B.3.6. EXAMEN.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\examen.h"
#include "d:\e_todos.h"
#include "d:\errores.h"
#include "d:\consulta.h"
#include "d:\ayuda.h"
#define  then

/*    Verifica que cadena no se encuentre en la lista enlazada Exámen. Regresa 1 si existe en la lista, regresa 0 si no
  se encontró. */
int error_grupoexa_igual(struct EXAMEN *p, char cadena[100])
{
 struct EXAMEN *r;
 int valor, numero,k,n;
                                                        // Verifica que cadena sea entero
 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
    {
     valor = 1;
     n = 18;
    }
  }
 numero = atoi(cadena);
 r = p;
 while (r != NULL) do                                   // Compara cada elemento de la lista con numero
 {
  if (((*r).numgrupo == numero) && ((*r).tipoexa == 2)) then
   {
    valor = 1;
    n = 37;
   }
  r = (*r).sig;
 }
 if (valor == 1) then
  llama_error(n);
 return(valor);
}

/*    Verifica que dato no se encuentre en la lista enlazada Exámen. Regresa 1 si existe en la lista, regresa 0 si no se
  encontró. */
int error_grupoexa_cambiar(struct EXAMEN *p, char cadena[100], int dato)
{
 struct EXAMEN *r;
 int valor, numero,n,k;
```

```
 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
    {
     valor = 1;
     n = 18;
    }
  }
 numero = atoi(cadena);
 r = p;
 while ((*p).num != dato) do
    p = (*p).sig;
 (*p).numgrupo = 0;
 while (r != NULL) do
 {
  if (((*r).numgrupo == numero) && ((*r).tipoesa == 2)) then
   {
    valor = 1;
    n = 37;
   }
  r = (*r).sig;
 }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}


//     Lee desde el teclado la información para cada campo de la lista enlazada EXAMEN.
void lee_nodo_examenes(struct EXAMEN *g, struct EXAMEN *e)
{
 char nombreprofe[100], rfc[100], cadena[100], hora1[100], profe[100], rfc2[100];
 int tipoesa, numgrupo, i, c, regresa, j, k, tipoprofe, tipo, tipoaslon, dia1, dia2,mes1, mes2, ano1, ano2;
 long int numsalon;
                                        // Se inicializan los campos.
 dia1 = 0;
 dia2 = 0;
 mes1 = 0;
 mes2 = 0;
 ano1 = 0;
 ano2 = 0;
 tipo = 0;
 tipoaslon = 0;
 tipoprofe = 0;
 numgrupo = 0;
 numsalon = 0;
 strcpy(cadena,"-");
 strcpy(nombreprofe,"-");
 strcpy(profe,"-");
 strcpy(rfc,"-");
 strcpy(rfc2,"-");
 strcpy(hora1,"00:00");
 pon(tope,150,COPY_PUT);          // Pone ventana gráfica 150
 pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
 pon(tope,210,COPY_PUT);          // Pone ventana gráfica 210
 cuadro(150,BLUE);
 texto("Presione <ENTER> para seleccionar o <ESC> para salir ....",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
 texto("Seleccione el tipo de exámen ...",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                                        // Lectura para la opción TIPO DE EXAMEN
 pon(tope,249,COPY_PUT);          // Pone ventana gráfica 249
 pon(tope,250,COPY_PUT);          // Pone ventana gráfica 250
 texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
 texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
 pon(tope,249,NOT_PUT);           // Pone ventana gráfica del cursor
```

```
i = opciones1(249,249,250,27);
quita(tope);                                        // Quita ventana gráfica del cursor
quita(tope);                                        // Quita ventana gráfica 250
quita(tope);                                        // Quita ventana gráfica 249
quita(tope);                                        // Quita ventana gráfica 210
quita(tope);                                        // Quita ventana gráfica 248
quita(tope);                                        // Quita ventana gráfica 150
if (i != -1) then
 {
  switch(i)
  {
       case 249: texto("EXAMEN EXTRAORDINARIO",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                 tipoexa = 1;
                 break;
       case 250: texto("EXAMEN FINAL",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                 tipoexa = 2;
                 break;
  }
  ventana_cambia_datos_materia();              // Pone menú inferior de opciones
  texto("Seleccione los datos de exámen que desea introducir . . .",165,BLUE,SMALL_FONT,3,2,2,1);
  texto("PROFESOR",166,LIGHTBLUE,SMALL_FONT,2,1,2,1);
  texto("SALON",167,LIGHTBLUE,SMALL_FONT,2,1,2,1);
  texto("GRUPO",168,LIGHTBLUE,SMALL_FONT,2,1,2,1);
  texto("HORARIO",169,LIGHTBLUE,SMALL_FONT,2,1,2,1);
  texto("FECHA",170,LIGHTBLUE,SMALL_FONT,2,1,2,1);
  i=166;
  j=1;
  pon(tope,i,NOT_PUT);                         // Pone ventana gráfica del cursor
  while (j == 1)
   {
       c = bioskey(0);
       switch(c)
       {
         case F1:  llama_ayuda(28);            // Llama módulo AYUDA
                   break;
                // Permite el desplazamiento del cursor sobre el menú de opciones
         case DERECHA:  quita(tope);
                        i++;
                        if (i == 171) then
                              i = 166;
                        pon(tope,i,NOT_PUT);
                        break;
         case IZQUIERDA: quita(tope);
                        i--;
                        if (i == 165) then
                              i = 170;
                        pon(tope,i,NOT_PUT);
                        break;
         case TAB:      if ((i >= 166) && (i <= 170)) then      // Cambia el menú inferior
                         {
                          regresa = i;
                          i = 152;
                          i = mover_ventana_menu(i,i,152,regresa);
                          if (i == 152) then
                                i = 0;
                         }
                        break;
         case ENTER:   quita(tope);
                       switch(i)
                       {
                         case 166: // Lee el nombre del profesor
                                   if (tipoexa == 1) then
                                   {
                                    pon(tope,179,COPY_PUT);          // Pone ventana gráfica 179
                                    texto("Datos del profesor titular ...",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
```

```
                                    }
                                    pon(tope,223,COPY_PUT);   // Pone ventana gráfica 223
                                    pon(tope,224,COPY_PUT);     // Pone ventana gráfica 224
                                    texto("Nombre del Profesor . . .",223,RED,SMALL_FONT,1,1,2,1);
                                    lee_cadena(nombreprofe,32,224,RED,SMALL_FONT,1,1,2,1);
                                    texto(nombreprofe,224,RED,SMALL_FONT,1,1,2,1);
                                    // Lee el nombramiento del profesor
                                    pon(tope,225,COPY_PUT);   // Pone ventana gráfica 225
                                    pon(tope,226,COPY_PUT);   // Pone ventana gráfica 226
                                    texto("TITULAR",225,RED,SMALL_FONT,1,1,2,1);
                                    texto("AYUTE",226,RED,SMALL_FONT,1,1,2,1);
                                    pon(tope,225,NOT_PUT);  // Pone ventana gráfica del cursor
                                    i = opciones1(225,225,226,18);
                                    quita(tope); // Quita ventana gráfica del cursor
                                    quita(tope); // Quita ventana gráfica 226
                                    quita(tope); // Quita ventana gráfica 225
                                    pon(tope,227,COPY_PUT);   // Pone ventana gráfica 227
                                    if (i == 225) then
                                      {
                                        texto("PROF. TITULAR",227,RED,SMALL_FONT,1,1,2,1);
                                        tipoprofe = 1;
                                      }
                                    if (i == 226) then
                                      {
                                        texto("AYUDANTE",227,RED,SMALL_FONT,1,1,2,1);
                                        tipoprofe = 2;
                                      }
                                    if (i == -1) then
                                      {
                                        texto("SIN NOMBRAMIENTO",227,RED, SMALL_FONT,1,1,2,1);
                                        tipoprofe = 3;
                                      }
                                    // Lee el R.F.C. del profesor
                                    pon(tope,173,COPY_PUT);   // Pone ventana gráfica 173
                                    pon(tope,174,COPY_PUT);   // Pone ventana gráfica 174
                                    texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                                    lee_cadena(rfc,14,174,RED,SMALL_FONT,1,1,2,1);
                                    texto(rfc,174,RED,SMALL_FONT,1,1,2,1);
                                    pon(tope,179,COPY_PUT);   // Pone ventana gráfica 179
                                    texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                    bioskey(0);
                                    quita(tope);                    // Quita ventana gráfica del cursor
                                    quita(tope);                    // Quita ventana gráfica 174
                                    quita(tope);                    // Quita ventana gráfica 173
                                    quita(tope);                    // Quita ventana gráfica 227
                                    quita(tope);                    // Quita ventana gráfica 224
                                    quita(tope);                    // Quita ventana gráfica 223
                                    if (tipoexa == 1) then
                                       quita(tope);
                                    // Datos del profesor suplente
                                    if (tipoexa == 1) then
                                     {
                                       pon(tope,179,COPY_PUT);              // Pone ventana gráfica 179
                                       texto("Datos del profesor suplente
...",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                                       pon(tope,223,COPY_PUT);              // Pone ventana gráfica 223
                                       pon(tope,224,COPY_PUT);              // Pone ventana gráfica 224
                                       texto("Nombre del Profesor . . .",223,RED,SMALL_FONT,1,1,2,1);
                                       lee_cadena(profe,32,224,RED,SMALL_FONT,1,1,2,1);
                                       texto(profe,224,RED,SMALL_FONT,1,1,2,1);
                                       // Lee el nombramiento del profesor
                                       pon(tope,225,COPY_PUT);              // Pone ventana gráfica 225
                                       pon(tope,226,COPY_PUT);              // Pone ventana gráfica 226
                                       texto("TITULAR",225,RED,SMALL_FONT,1,1,2,1);
```

```
                texto("AYUTE",226,RED,SMALL_FONT,1,1,2,1);
                        // Pone ventana gráfica del cursor
                pon(tope,225,NOT_PUT);
                i = opciones1(225,226,228,18);
                quita(tope);        // Quita ventana gráfica del cursor
                quita(tope);        // Quita ventana gráfica 226
                quita(tope);        // Quita ventana gráfica 225
                pon(tope,227,COPY_PUT);             // Pone ventana gráfica 227
                if (i == 225) then
                 {
                      texto("PROF. TITULAR",227,RED, SMALL_FONT,1,1,2,1);
                      tipo = 1;
                 }
                if (i == 226) then
                 {
                      texto("AYUDANTE",227,RED,SMALL_FONT,1,1,2,1);
                      tipo = 2;
                 }
                if (i == -1) then
                 {
                      texto("SIN NOMBRAMIENTO",227,RED, SMALL_FONT,1,1,2,1);
                      tipo = 3;
                 }
                // Lee el  R.F.C. del profesor
                pon(tope,173,COPY_PUT);             // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);             // Pone ventana gráfica 174
                texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                lee_cadena(rfc2,14,174,RED,SMALL_FONT,1,1,2,1);
                        // Lee una cadena desde el teclado
                texto(rfc2,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);             // Pone ventana gráfica 179
                texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                blockey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 227
                quita(tope);        // Quita ventana gráfica 224
                quita(tope);        // Quita ventana gráfica 223
                quita(tope);        // Quita ventana gráfica 179
               }
              i = 166;
              pon(tope,i,NOT_PUT);// Quita ventana gráfica del cursor
              break;

        case 167: // Lee salón
                pon(tope,171,COPY_PUT);   // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);   // Pone ventana gráfica 172
                texto("Número de salón . . .",171,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) do
                {
                 // Lee una cadena del teclado
                 lee_cadena(cadena,5,172,RED,SMALL_FONT,1,1,2,1);
                 k = error_numero_entero(cadena);
                }
                texto(cadena,172,RED,SMALL_FONT,1,1,2,1);
                numsalon = atol(cadena);
                // Lee la clasificación del salón
                pon(tope,173,COPY_PUT);   // Pone ventana gráfica 173
                texto("Tipo de salón",173,RED,SMALL_FONT,1,1,2,1);
                pon(tope,175,COPY_PUT);   // Pone ventana gráfica 175
                pon(tope,176,COPY_PUT);   // Pone ventana gráfica 176
                pon(tope,177,COPY_PUT);   // Pone ventana gráfica 177
```

```
                    pon(tope,178,COPY_PUT);   // Pone ventana gráfica 178
                    texto("D",175,RED,SMALL_FONT,1,1,2,1);
                    texto("L",176,RED,SMALL_FONT,1,1,2,1);
                    texto("C",177,RED,SMALL_FONT,1,1,2,1);
                    texto("G",178,RED,SMALL_FONT,1,1,2,1);
                    pon(tope,175,NOT_PUT);// Pone ventana gráfica del cursor
                    l = opciones1(175,175,178,24);
                    quita(tope);              // Quita ventana gráfica del cursor
                    quita(tope);              // Quita ventana gráfica 178
                    quita(tope);              // Quita ventana gráfica 177
                    quita(tope);              // Quita ventana gráfica 176
                    quita(tope);              // Quita ventana gráfica 175
                    pon(tope,174,COPY_PUT);   // Pone ventana gráfica 174
                    if (l == 175) then
                     {
                      texto("Salón de dibujo",174,RED,SMALL_FONT,1,1,2,1);
                      tiposalon = 1;
                     };
                    if (l == 176) then
                     {
                      texto("Salón de laboratorio",174,RED,SMALL_FONT,1,1,2,1);
                      tiposalon = 2;
                     };
                    if (l == 177) then
                     {
                      texto("Salón chico (máx 30 alumnos)",174,RED, SMALL_FONT,1,1,2,1);
                      tiposalon = 3;
                     };
                    if (l == 178) then
                     {
                      texto("Salón grande (máx 60 alumnos)",174,RED, SMALL_FONT,1,1,2,1);
                      tiposalon = 4;
                     };
                    if (l == -1) then
                     {
                      texto("Salón sin clasificar",174,RED,SMALL_FONT,1,1,2,1);
                      tiposalon = 5;
                     };
                    pon(tope,179,COPY_PUT);   // Pone ventana gráfica 179
                    texto("Presione una tecla para continuar...",179,LIGHTRED,
SMALL_FONT,1,1,2,1);

                    bioskey(0);
                    quita(tope);              // Quita ventana gráfica 179
                    quita(tope);              // Quita ventana gráfica 174
                    quita(tope);              // Quita ventana gráfica// 173
                    quita(tope);              // Quita ventana gráfica 172
                    quita(tope);              // Quita ventana gráfica 171
                    l = 167;
                    pon(tope,l,NOT_PUT);// Pone ventana gráfica del cursor
                    break;

          case 168: // Lee  Grupo
                    pon(tope,171,COPY_PUT);   // Pone ventana gráfica 171
                    pon(tope,172,COPY_PUT);   // Pone ventana gráfica 172
                    texto("Número de grupo . . .",171,RED,SMALL_FONT,1,1,2,1);
                    k = 1;
                    while (k == 1) do
                    {
                          // Lee una cadena desde el teclado
                      lee_cadena(cadena,4,172,RED,SMALL_FONT,1,1,2,1);
                      k = error_gruposa_igual(e,cadena);
                    }
                    texto(cadena,172,RED,SMALL_FONT,1,1,2,1);
                    numgrupo = atoi(cadena);
                    pon(tope,179,COPY_PUT);   // Pone ventana gráfica 179
```

```
                                texto("Presione una tecla para continuar...",179,LIGHTRED,
SMALL_FONT,1,1,2,1);
                                bioskey(0);
                                quita(tope);          // Quita ventana gráfica 179
                                quita(tope);          // Quita ventana gráfica 172
                                quita(tope);          // Quita ventana gráfica 171
                                i = 168;
                                pon(tope,i,NOT_PUT); // Quita ventana gráfica del cursor
                                break;

                        case 169: // Lee Horario
                                pon(tope,171,COPY_PUT);          // Pone ventana gráfica 171
                                pon(tope,172,COPY_PUT);          // Pone ventana gráfica 172
                                texto("Hora  (hh:mm) . . . .",171,RED,SMALL_FONT,1,1,2,1);
                                k = 1;
                                while (k == 1) do
                                {       // Lee una cadena desde el teclado
                                  lee_cadena(hora1,5,172,RED,SMALL_FONT,1,1,2,1);
                                  k = hora_correcta(hora1,hora1,1);
                                }
                                texto(hora1,172,RED,SMALL_FONT,1,1,2,1);
                                pon(tope,179,COPY_PUT);          // Pone ventana gráfica 179
                                texto("Presione una tecla para continuar...",179,LIGHTRED,
SMALL_FONT,1,1,2,1);
                                        bioskey(0);
                                quita(tope);          // Quita ventana gráfica 179
                                quita(tope);          // Quita ventana gráfica 172
                                quita(tope);          // Quita ventana gráfica 171
                                i = 169;
                                pon(tope,i,NOT_PUT); // Pone ventana gráfica del cursor
                                break;

                        case 170: // Lee Fecha
                                texto("Seleccione...",180,BLUE,SMALL_FONT,1,1,2,1);
                                texto("DIA",182,RED,SMALL_FONT,1,1,2,1);
                                texto("MES",183,RED,SMALL_FONT,1,1,2,1);
                                texto("A&O",184,RED,SMALL_FONT,1,1,2,1);
                                if (tipoesa == 2) then
                                  texto("1a. vuelta",185,BLUE,SMALL_FONT,1,1,2,1);
                                pon(tope,179,COPY_PUT);    // Pone ventana gráfica 179
                                texto("Presione <ESC> cuando termine de seleccionar",
179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                                pon(tope,182,NOT_PUT); // Pone ventana gráfica del cursor
                                i = 182;
                                while (i != -1) do
                                 {
                                  i = opciones1(i,182,184,29);
                                  switch(i)
                                   { // Quita ventana gráfica del cursor
                                        case 182: quita(tope);          // Pone ventana gráfica 173
                                                  pon(tope,173,COPY_PUT);
                                                        // Pone ventana gráfica 174
                                                  pon(tope,174,COPY_PUT);
                                                  texto("Dia  (dd) . . . .",179,RED,SMALL_FONT,1,1,2,1);
                                                  k = 1;
                                                  while (k == 1) do
                                                  {
                                                        // Lee una cadena del teclado
                                                    lee_cadena(cadena,2,174,RED, SMALL_FONT,1,1,2,1);
                                                    k = error_dia(cadena);
                                                  }
                                                  texto(cadena,174,RED,SMALL_FONT,1,1,2,1);
                                                  dia1 = atoi(cadena);
                                                        // Pone ventana gráfica 179
                                                  pon(tope,179,COPY_PUT);
```

```
                                        texto("Presione una tecla para continuar...",179,
LIGHTRED,SMALL_FONT,1,1,2,1);

                                        bioskey(0);
                                        quita(tope);        // Quita ventana gráfica 179
                                        quita(tope);        // Quita ventana gráfica 174
                                        quita(tope);        // Quita ventana gráfica 173
                                        if (tipoesa == 2) then
                                         { // Pone ventana gráfica 185
                                           pon(tope,185,COPY_PUT);
                                           texto("2a. vuelta",185,BLUE,SMALL_FONT,1,1,2,1);
                                                // Pone ventana gráfica 173
                                               pon(tope,173,COPY_PUT);
                                                // Pone ventana gráfica 174
                                               pon(tope,174,COPY_PUT);
                                           texto("Dia  (dd) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                                                k = 1;
                                           while (k == 1) do
                                                {
                                                // Lee una cadena desde el teclado
                                                lee_cadena(cadena,2,174,RED,
SMALL_FONT,1,1,2,1);

                                                k = error_dia(cadena);
                                                }
                                           texto(cadena,174,RED, SMALL_FONT,1,1,2,1);
                                           dia2 = atoi(cadena);
                                                // Pone ventana gráfica 179
                                           pon(tope,179,COPY_PUT);
                                           texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                           bioskey(0);
                                           quita(tope); // Quita ventana gráfica 179
                                           quita(tope); // Quita ventana gráfica 174
                                           quita(tope); // Quita ventana gráfica 173
                                           quita(tope); // Quita ventana gráfica 185
                                         }
                                        i = 182; // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;

                                        // Quita ventana gráfica del cursor
                                case 183: quita(tope); // Pone ventana gráfica 173
                                        pon(tope,173,COPY_PUT);
                                                // Pone ventana gráfica 174
                                        pon(tope,174,COPY_PUT);
                                        texto("Mes  (mm) . . . .",173,RED, SMALL_FONT,1,1,2,1);
                                        k = 1;
                                        while (k == 1) do
                                         {
                                                // Lee una cadena del teclado
                                          lee_cadena(cadena,2,174,RED,SMALL_FONT,1,1,2,1);
                                          k = error_mes(cadena);
                                         }
                                        texto(cadena,174,RED,SMALL_FONT,1,1,2,1);
                                        mes1 = atoi(cadena);
                                                // Pone ventana gráfica 179
                                        pon(tope,179,COPY_PUT);
                                        texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                        bioskey(0);
                                        quita(tope);        // Quita ventana gráfica 179
                                        quita(tope);        // Quita ventana gráfica 174
                                        quita(tope);        // Quita ventana gráfica 173
                                        if (tipoesa == 2) then
                                         {    // Pone ventana gráfica 185
                                           pon(tope,185,COPY_PUT);
```

```
                                        texto("2a. vuelta",165,BLUE, SMALL_FONT,1,1,2,1);
                                            // Pone ventana gráfica 179
                                        pon(tope,179,COPY_PUT);
                                            // Pone ventana gráfica 174
                                        pon(tope,174,COPY_PUT);
                                        texto("Mes  (mm) . . .
.",173,RED,SMALL_FONT,1,1,2,1);

                                        k = 1;
                                        while (k == 1) do
                                            { // Lee una cadena desde el teclado
                                            lee_cadena(cadena,2,174,RED,
SMALL_FONT,1,1,2,1);

                                             k = error_mes(cadena);
                                            }
                                        texto(cadena,174,RED,SMALL_FONT,1,1,2,1);
                                        mes2 = atoi(cadena);
                                            // Lee una cadena desde el teclado
                                        pon(tope,179,COPY_PUT);
                                        texto("Presione una tecla para continuar...",
179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                        bioskey(0);
                                        quita(tope); // Quita ventana gráfica 179
                                        quita(tope); // Quita ventana gráfica 174
                                        quita(tope); // Quita ventana gráfica 173
                                        quita(tope); // Quita ventana gráfica 185
                                      }
                                    l = 183; // Pone ventana gráfica del cursor
                                    pon(tope,l,NOT_PUT);
                                    break;
                                            // Quita ventana gráfica del cursor
                            case 184: quita(tope);          // Pone ventana gráfica 173
                                    pon(tope,173,COPY_PUT);
                                            // Pone ventana gráfica 174
                                    pon(tope,174,COPY_PUT);
                                    texto("A&o  (aaaa) . . . .",173,RED, SMALL_FONT,1,1,2,1);
                                    k = 1;
                                    while (k == 1) do
                                     {
                                            // Lee una cadena del teclado
                                      lee_cadena(cadena,4,174,RED, SMALL_FONT,1,1,2,1);
                                       k = error_ano(cadena);
                                     }
                                    texto(cadena,174,RED,SMALL_FONT,1,1,2,1);
                                    ano1 = atoi(cadena);
                                            // Pone ventana gráfica 179
                                    pon(tope,179,COPY_PUT);
                                    texto("Presione una tecla para continuar...",179,
LIGHTRED,SMALL_FONT,1,1,2,1);

                                    bioskey(0);
                                    quita(tope);        // Quita ventana gráfica 179
                                    quita(tope);        // Quita ventana gráfica 174
                                    quita(tope);        // Quita ventana gráfica 173
                                    if (tipoasa == 2) then
                                     {      // Pone ventana gráfica 185
                                       pon(tope,185,COPY_PUT);
                                       texto("2a. vuelta",165,BLUE, SMALL_FONT,1,1,2,1);
                                            // Pone ventana gráfica 179
                                       pon(tope,173,COPY_PUT);
                                            // Pone ventana gráfica 174
                                       pon(tope,174,COPY_PUT);
                                       texto("A&o  (aaaa) . . . .",179,RED,
SMALL_FONT,1,1,2,1);

                                       k = 1;
                                       while (k == 1) do
                                            {
```

```
                                                    // Lee una cadena del teclado
                                                    lee_cadena(cadena,4,174,RED,
SMALL_FONT,1,1,2,1);
                                                     k = error_ano(cadena);
                                                    }
                                                  texto(cadena,174,RED, SMALL_FONT,1,1,2,1);
                                                  ano2 = atoi(cadena);
                                                      // Pone ventana gráfica 179
                                                          pon(tope,179,COPY_PUT);
                                                  texto("Presione una tecla para continuar...",
179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                  bioskey(0);
                                                  quita(tope); // Quita ventana gráfica 179
                                                  quita(tope); // Quita ventana gráfica 174
                                                  quita(tope); // Quita ventana gráfica 173
                                                  quita(tope); // Quita ventana gráfica 185
                                                 }
                                                i = 184;  // Pone ventana gráfica 184
                                                pon(tope,i,NOT_PUT);
                                                break;
                                    }
                                   }
                                  quita(tope);          // Quita ventana gráfica del cursor
                                  quita(tope);          // Quita ventana gráfica 179
                                  pon(tope,179,COPY_PUT);    // Pone ventana gráfica 179
                                  texto("Presione una tecla para continuar...",
179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                  bioskey(0);
                                  quita(tope);          // Quita ventana gráfica 179
                                  for (k=180; k<=185; k++)
                                   {
                                    setfillstyle(SOLID_FILL,GREEN);
                                    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                   }
                                  i = 170;
                                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                                  break;
                          }
        }
    }
  quita(tope);        // Quita ventana gráfica del cursor
  k = 210;
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  for (k = 165; k <= 170; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
  if (tipoasa == 2) then
  {
   k = 165;
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
  quita(tope);                              // Quita ventana gráfica 150
                                            // Asignar campos al nodo *g
  (*g).tipoasa = tipoasa;
  (*g).numgrupo = numgrupo;
  (*g).tipoprofe = tipoprofe;
  (*g).tipo = tipo;
  (*g).tipoasion = tipoasion;
  for (k=0; nombreprofe[k]; k++)
    nombreprofe[k] = toupper(nombreprofe[k]);
  strcpy((*g).nombreprofe,nombreprofe);
```

```
    for (k=0; profe[k]; k++)
      profe[k] = toupper(profe[k]);
    strcpy((*g).profe,profe);
    for (k=0; rfc[k]; k++)
      rfc[k] = toupper(rfc[k]);
    strcpy((*g).rfc,rfc);
    for (k=0; rfc2[k]; k++)
      rfc2[k] = toupper(rfc2[k]);
    strcpy((*g).rfc2,rfc2);
    (*g).numsalon = numsalon;
    strcpy((*g).hora1,hora1);
    (*g).dia1 = dia1;
    (*g).dia2 = dia2;
    (*g).mes1 = mes1;
    (*g).mes2 = mes2;
    (*g).ano1 = ano1;
    (*g).ano2 = ano2;
   }
}


//    Opción para agregar más nodos a la lista enlazada EXAMEN.
int salir_lee_examen(void)
{
 int c,j;

 pon(tope,253,COPY_PUT);                          // Pone ventana gráfica 253
 texto("AGREGAR UNO MAS",253,RED,SANS_SERIF_FONT,1,2,1,2);
 pon(tope,254,COPY_PUT);                          // Pone ventana gráfica 254
 texto("SALIR",254,RED,SANS_SERIF_FONT,1,2,1,2);
 c=0;
 j = -1;
 pon(tope,253,NOT_PUT);                           // Pone ventana gráfica del cursor
 while (c == 0)
 {
  while (j == -1) do
   j=opciones1(253,253,254,30);
   switch (j)
     {
          case  253 : c = -1;
              break;
          case  254 : c = -3;
              break;
     }
 }
 quita(tope);                                     // Quita ventana gráfica del cursor
 quita(tope);                                     // Quita ventana gráfica 253
 quita(tope);                                     // Quita ventana gráfica 254
 return(c);
}


//  Despliega en pantalla los datos de la lista enlazada EXAMEN. Regresa el valor de la última ventana colocada.
int ventanas_datos_examen(struct EXAMEN *a, int inicio, int fin, int *max, int z)
{
 struct EXAMEN *r;
 int i,c,k;
 char dia[20],mes[20],ano[20],numero[20];

 r=a;
 i = 257;
 while ((*r).num != inicio) do
        r=(*r).sig;
j = *max;
for (k=inicio; k <= fin; k++)
```

```
{
    if ((*r).tipoesa == z) then
        {
          cuadro(i,CYAN);
          if (z == 1) then
           {
            pon(tope,i,COPY_PUT);
            itoa((*r).dia1,dia,10);
            for (c=0; dia[c]; c++);
            if (c == 1) then
                  {
                   strcpy(numero,"0");
                   strcat(numero,dia);
                  }
            else
                  strcpy(numero,dia);
            strcat(numero,"/");
            itoa((*r).mes1,mes,10);
            for (c=0; mes[c]; c++);
            if (c == 1) then
                  strcat(numero,"0");
            strcat(numero,mes);
            strcat(numero,"/");
            itoa((*r).ano1,ano,10);
            strcat(numero,ano);
           }
          else
           {
            pon(tope,i,COPY_PUT);
            itoa((*r).numgrupo,numero,10);
           }
          texto (numero,i,RED,SMALL_FONT,3,2,2,1);
          i=i+1;
          pon(tope,i,COPY_PUT);
          texto ((*r).nombreprofe,i,RED,SMALL_FONT,3,2,2,1);
          i=i+1;
          *max=j;
          j=j+1;
        }
        r = (*r).sig;
}
return(i);
}


//   Pone un submenú de opciones
void ventana_examen(int tipo)
{
if (tipo == 1) then
 texto("FECHA",368,LIGHTBLUE,SMALL_FONT,2,1,2,1);
if (tipo == 2) then
 texto("GRUPO",368,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("PROFESOR TITULAR",369,LIGHTBLUE,SMALL_FONT,2,1,2,1);
pon(tope,150,COPY_PUT);
cuadro(150,BLUE);
texto("SALIR",231,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("<F1> AYUDA",232,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("<TAB> CAMBIA MENU",233,LIGHTRED,SMALL_FONT,4,3,2,1);
texto("PAG. 1",234,LIGHTRED,SMALL_FONT,2,1,2,1);
}


//   Llama a las funciones correspondientes para desplegar en pantalla los datos de la lista enlazada EXAMEN.
int mover_pantallas_examen(int i, int min, int salir, struct EXAMEN *e, int tipo, int cuenta)
{
```

```
struct EXAMEN *r, *s;
struct POS{
            int inicio;
            int fin;
           };
 struct POS *pos;
int c,regresa,inicio,fin,grupo,ultimo,j,quot,rem,numo,k,dato,max,a;
                                                    // Se reserva espacio en memoria
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
   {
    printf("Error.");
    exit(1);
   }
r = e;
s = e;
dato = 0;
ultimo = 0;
while (r != NULL) do
 {
  if ((*r).tipoexa == tipo) then
        ultimo = ultimo + 1;
  r=(*r).sig;
 }
quot=(ultimo/4);
rem=(ultimo%4);
if (rem != 0) then
  grupo=quot + 1;
else
  grupo=quot;
if (grupo == 0) then
 {
  llama_error(42);
  return(-2);
 }
fin=0;
for (j=1; j <= grupo; j++)
{
  a = 0;
  while (( s != NULL) && (a < 4)) do
  {
   if ((*s).tipoexa == tipo) then
         {
          fin=(*s).num;
          if (a == 0) then
            inicio = (*s).num;
          a = a + 1;
         }
    s=(*s).sig;
  }
  pos[j].inicio=inicio;
  pos[j].fin=fin;
}
ventana_examen(tipo);                          // Pone un menú de opciones
max=121;
j=1;
inicio=pos[j].inicio;
fin=pos[j].fin;
numo=ventanas_datos_examen(e,inicio,fin,&max,tipo);  // Despliega en pantalla los datos de exámen
c=END;
regresa=0;
pon(tope,i,NOT_PUT);
while (c != ENTER) do
{
  c=bioskey(0);
```

```
switch(c)
{
      case F1: llama_ayuda(35);          // Llama al módulo AYUDA
               break;
      case ABAJO:  quita(tope);          // Permite desplazar el cursor sobre el menú de opciones
                   i++;
                   if (i == (max + 1)) then
                         i = min;
                   pon(tope,i,NOT_PUT);
                   break;
      case ARRIBA: quita(tope);
                   i--;
                   if (i == (min - 1)) then
                         i = max;
                   pon(tope,i,NOT_PUT);
                   break;
      case TAB:  if ((i >= min) && (i <= max)) then  // Cambia al menú inferior de opciones
                   {
                    regresa=i;
                    i=salir;
                    i=mover_ventana_menu(i,i,salir,regresa);
                    if (i == 231) then
                          {
                           c = ENTER;
                           dato = -2;
                          }
                   }
                   break;
      case PGDN:  quita(tope);                 // Cambia a la siguiente página
                  numc--;
                  for (k=257; k <= numc; k++)
                        quita(tope);
                  for (k=121; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                  j++;
                  if (j == (grupo+1)) then
                    j=1;
                  inicio=pos[j].inicio;
                  fin=pos[j].fin;
                  max=121;        // Despliega en pantalla la página apuntada por j
                  numc=ventanas_datos_examen(e,inicio,fin,&max,tipo);
                  if (i > max) then
                   {
                        i = max;
                        pon(tope,i,NOT_PUT);
                   }
                  else
                   pon(tope,i,NOT_PUT);
                  break;
      case PGUP:  quita(tope);                 // Cambia a la página anterior
                  numc--;
                  for(k=257; k <= numc; k++)
                        quita(tope);
                  for (k=121; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                  j--;
                  if (j == 0) then
                    j=grupo;
                  inicio=pos[j].inicio;
```

```
                        fin=pos[i].fin;
                        max=121;
                        numc=ventanas_datos_examen(e,inicio,fin,&max,tipo);
                        if (i > max) then
                          {
                              i = max;
                              pon(tope,i,NOT_PUT);
                          }
                        else
                          pon(tope,i,NOT_PUT);
                        break;
    }
  }
  while ( s != NULL) do
          s = (*s).sig;
  s = s;
  if (dato != -2) then                          // Obtiene la posición del nodo seleccionado
    {
     dato = i - 120;
     dato=dato + 4*(j-1);
     a = 0;
     ultimo = 0;
     while ((s != NULL) && (dato != ultimo)) do
          {
           if ((*s).tipoesa == tipo) then
             {
              ultimo = ultimo + 1;
              if (ultimo == dato) then
                    a = (*s).num;
             }
            s = (*s).sig;
          }
    }
    else
     a = dato;
  quita(tope);                                  // Quita ventana gráfica del cursor
  numc--;
  for (k=257; k <= numc; k++)
    quita(tope);
  for (k=121; k <= max; k++)
     {
          setfillstyle(SOLID_FILL,GREEN);
          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  quita(tope);                                  // Quita ventana gráfica 150
  for (k=230; k <= 234; k++)
     {
          setfillstyle(SOLID_FILL,GREEN);
          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  for (k=368; k <= 369; k++)
     {
          setfillstyle(SOLID_FILL,GREEN);
          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  farfree(pos);                                 // Libera el espacio reservado
  return(dato);                                 // Regresa la posición del nodo seleccionado
}


//   Crea una nueva lista enlazada EXAMEN.
void crear_examen(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct EXAMEN m;
```

```
int dato, k, l, ultimo;

r=p;
pon(tope,100,COPY_PUT);                          // Pone ventana gráfica 100
texto("CREANDO UNA NUEVA LISTA DE EXAMENES",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
if ((*p).num == 0) then
{
 llama_error(30);
 quita(tope);                                    // Quita ventana gráfica 100
 return;
}
  l = 0;
  dato = 0;
  while (l != -1) do
   {
    r = p;
    dato = escoje_consulta_materia(r,&l);        // Selecciona la forma de consultar MATERIA
    if (l != -1) then
          {
          if (l != 351) then
           {
            while ((*r).num != dato)
                    r = (*r).materias;
            k = 1;
            if ((*r).examen != NULL) then
                  k = confirmar(1);
            if (k == 0) then
             {
              quita(tope);                       // Quita ventana gráfica 100
              return;
             }              // Lee los nuevos campos para la lista enlazada EXAMEN
            lee_nodo_examenes(&m,(*r).examen);
            (*r).examen = crea_lista_examen();   // Crea, inicializando la nueva lista enlazada EXAMEN
            pon_examen((*r).examen,m);           // Pone el nuevo nodo en la lista enlazada EXAMEN
            while (r != NULL)
              r = (*r).materias;
           }
          else
           {
            s = p;
            ultimo = 0;
            while (s != NULL) do
             {
                  ultimo = ultimo + 1;
                  s = (*s).materias;
             }
            while (dato != -2) do
            {
             r = p;
                  dato=mover_pantallas_consulta(120,120,231,p,ultimo);
                  if (dato > 0) then
                    {
                     while ((*r).num != dato)
                             r = (*r).materias;
                     k = 1;
                     if ((*r).examen != NULL) then
                       k = confirmar(1);
                     if (k == 0) then
                      {
                       quita(tope); /* 100 */
                       return;
                      }
                      // Lee los datos para el nuevo nodo de la lista enlazada EXAMEN
                     lee_nodo_examenes(&m,(*r).examen);
                     // Crea, inicializando una nueva lista enlazada EXAMEN
```

```
                    (*r).examen = crea_lista_examen();
                    pon_examen((*r).examen,m);   // Agrega un nuevo nodo a la lista enlazada EXAMEN
                    while (r != NULL)
                      r = (*r).materias;
                   }
                }
              }
            }
          }
        }
 quita(tope);                    // Quita ventana gráfica 100
}


// Añade nuevos nodos a la lista enlazada EXAMEN.
void agregar_examen(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct EXAMEN m;
 int dato,c,ultimo,i;

 pon(tope,100,COPY_PUT);                          // Pone ventana gráfica 100
 texto("AGREGANDO UN NUEVO EXAMEN A LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(31);
  quita(tope);                                                    // Quita ventana gráfica 100
  return;
 }
 i = 0;
 dato = 0;
 while (i != -1) do
  {
   r = p;
   dato = escoje_consulta_materia(r,&i);          // Selecciona la forma de consultar MATERIA
   if (i != -1) then
    {
        if (i != 351) then
         {
          while ((*r).num != dato)
                r = (*r).materias;
          if ((*r).examen == NULL) then
          {
           llama_error(34);
          }
          else
           {
                while (dato != -3) do
                {          // Lee los datos para un nuevo nodo en la lista enlazada EXAMEN
                  lee_nodo_examenes(&m,(*r).examen);
                        // Pone el nuevo nodo al final de la lista enlazad EXAMEN
pon_examen((*r).examen,m);
                        // Entra al ciclo para seguir añadiendo nuevos nodos a la lista
                      dato = salir_lee_examen();
                }
                while (r != NULL)
                      r = (*r).materias;
           }
         else
          {
           s = p;
           ultimo = 0;
           while (s != NULL) do
            {
                ultimo = ultimo + 1;
                s = (*s).materias;
```

```
            }
            while (dato != -2) do              // Muestra toda la lista de materias para seleccionar alguna
            {
             r = p;
              dato=mover_pantallas_consulta(120,120,231,p,ultimo);
              if (dato > 0) then
                    {
                     while ((*r).num != dato)
                              r = (*r).materias;
                     if ((*r).examen == NULL) then
                     {
                      llama_error(34);
                     }
                     else
                      {
                        while (dato != -3) do
                             {          // Lee los datos de un nuevo nodo de la lista enlazada MATERIA
                              lee_nodo_examenes(&m,(*r).examen);
                                        // Pone al final de la lista enlazada el nuevo nodo
                              pon_examen((*r).examen,m);
                                        // Entra al ciclo para seguir añadiendo nodos a la lista.
                              dato = salir_lee_examen();
                             }
                      }
                      while (r != NULL)
                              r = (*r).materias;
                    }
            }
           }
      }
   }
 quita(tope);                               // Quita ventana gráfica 100
}


//    Elimina nodos de la lista enlazada EXAMEN.
void eliminar_examen(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 int dato,c,i,k, ultimo, m;

 pon(tope,100,COPY_PUT);                            // Pone ventana gráfica 100
 texto("ELIMINANDO UN EXAMEN DE LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(32);
  quita(tope);                              // Quita ventana gráfica 100
  return;
 }
 i = 0;
 dato = 0;
 while (i != -1) do
  {
    r = p;
    dato = escoje_consulta_materia(r,&i);          // Selecciona la forma de consultar MATERIA
    if (i != -1) then
     {
      if (i != 351) then
            {
             while ((*r).num != dato) do
                      r = (*r).materias;
             if ((*r).examen == NULL) then
               llama_error(35);
             else
```

```
{
        m = 0;
        while (m != -1) do
        {
         pon(tope,150,COPY_PUT);     // Pone ventana gráfica 150
         cuadro(150,BLUE);
         pon(tope,210,COPY_PUT);     // Pone ventana gráfica 210
         texto("Presione <ENTER> para seleccionar o <ESC> para salir
...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
         texto("Seleccione el exámen que desea eliminar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
         pon(tope,249,COPY_PUT);     // Pone ventana gráfica 249
         pon(tope,250,COPY_PUT);     // Pone ventana gráfica 250
         texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
         texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
         pon(tope,249,NOT_PUT);      // Pone ventana gráfica del cursor
         m = opciones1(249,249,250,31);
         quita(tope);          // Quita ventana gráfica del cursor
         quita(tope);          // Quita ventana gráfica 250
         quita(tope);          // Quita ventana gráfica 249
         quita(tope);          // Quita ventana gráfica 210
         quita(tope);          // Quita ventana gráfica 150
         if (m != -1) then
          {
           switch(m)
                {                       // Pone ventana gráfica 100
                 case 249: pon(tope,100,COPY_PUT);
                           texto("ELIMINANDO UN EXAMEN
EXTRAORDINARIO",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                           m = 1;
                           break;
                 case 250: pon(tope,100,COPY_PUT);
                           texto("ELIMINANDO UN EXAMEN FINAL",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                           m = 2;
                           break;
                }
           k = 0;
           e = (*r).examen;
           while (e != NULL) do
                {
                if ((*e).tipoexa == m) then
                  k = k + 1;
                e = (*e).sig;
                }
           if (k == 0) then
                llama_error(42);
           else
                {       // Despliega los datos de EXAMEN y elimina el nodo seleccionado
                  c = 0;
                  while (c != -2) do
                   {
                    c = mover_pantallas_examen(121,121,231,(*r).examen,m,k);
                    if (c > 0) then
                        {
                         k = confirmar(2);
                         if (k == 1) then
                          {
                           c = quita_examen((*r).examen,c);
                           if (c == 0) then
                            {
                                (*r).examen = NULL;
                                c = -2;
                                m = -1;
                            }
                          }
                        }
```

```
                    }
                  }
                quita(tope);        // Quita ventana gráfica 100
              }
            }
          }
        while (r != NULL) do
          r = (*r).materias;
      }
    else
      {
        s = p;
        ultimo = 0;
        while (s != NULL) do
          {
            ultimo = ultimo + 1;
            s = (*s).materias;
          }
        while (dato != -2) do
          {
            r = p;
            dato = mover_pantallas_consulta(120,120,231,p,ultimo);
            if (dato > 0) then
                {
                  while ((*r).num != dato) do
                    r = (*r).materias;
                  if ((*r).examen == NULL) then
                    llama_error(35);
                  else
                    {
                      m = 0;
                      while (m != -1) do
                          {
                            pon(tope,150,COPY_PUT);      // Pone ventana gráfica 150
                            cuadro(150,BLUE);
                            pon(tope,210,COPY_PUT);      // Pone ventana gráfica 210
                            texto("Presione <ENTER> para seleccionar o <ESC> para salir
...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                            texto("Seleccione el examen que desea eliminar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                            pon(tope,249,COPY_PUT);      // Pone ventana gráfica 249
                            pon(tope,250,COPY_PUT);      // Pone ventana gráfica 250
                            texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                            texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                            pon(tope,249,NOT_PUT);       // Pone ventana gráfica del cursor
                            m = opciones1(249,249,250,32)
                            quita(tope);                 // Quita ventana gráfica del cursor
                            quita(tope);                 // Quita ventana gráfica 250
                            quita(tope);                 // Quita ventana gráfica 249
                            quita(tope);                 // Quita ventana gráfica 210
                            quita(tope);                 // Quita ventana gráfica 150
                            if (m != -1) then
                              {
                                switch(m)
                                    {                    // Pone ventana gráfica 100
                                      case 249: pon(tope,100,COPY_PUT);
                                                texto("ELIMINANDO UN EXAMEN
EXTRAORDINARIO",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                                m = 1;
                                                break;
                                      case 250: pon(tope,100,COPY_PUT);
                                                texto("ELIMINANDO UN EXAMEN
FINAL",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                                m = 2;
                                                break;
                                    }
```

```
                                k = 0;
                                e = (*r).examen;
                                while (e != NULL) do
                                {
                                    if ((*e).tipoexa == m) then
                                      k = k + 1;
                                    e = (*e).sig;
                                }
                                if (k == 0) then
                                    llama_error(42);
                                else
                                    {// Despliega los datos de EXAMEN y elimina el nodo seleccionado
                                      c = 0;
                                      while (c != -2) do
                                      {
                                       c = mover_pantallas_examen(121,121,231,(*r).examen,m,k);
                                       if (c > 0) then
                                       {
                                            k = confirmar(2);
                                            if (k == 1) then
                                            {
                                              c = quita_examen((*r).examen,c);
                                              if (c == 0) then
                                                {
                                                (*r).examen = NULL;
                                                c = -2;
                                                m = -1;
                                                }
                                            }
                                       }
                                      }
                                    }
                                quita(tope);     // Quita ventana gráfica 100
                              }
                            }
                          }
                          while (r != NULL) do
                            r = (*r).materias;
                        }
                    }
                }
            }
        }
 quita(tope);     // Quita ventana gráfica 100
}

//     Modifica los datos de algún nodo de la lista enlazada EXAMEN.
void cambiar_examen(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 int dato,c,i,m,k,ultimo;

 pon(tope,100,COPY_PUT);                                    // Pone ventana gráfica 100
 texto("CAMBIANDO DATOS DE UN EXAMEN DE LA LISTA",100,BLUE, SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(33);
  quita(tope);                                              // Quita ventana gráfica 100
  return;
 }
 i = 0;
 dato = 0;
 while (i != -1) do                                         // Selecciona la forma de consultar MATERIA
  {
```

```
r = p;
dato = escoje_consulta_materia(r,&i);
if (i != -1) then
 {
  if (i != 351) then
        {
         while ((*r).num != dato) do
                r = (*r).materias;
         if ((*r).examen == NULL) then
           llama_error(36);
         else
          {
                m = 0;
                while (m != -1) do
                {
                 pon(tope,150,COPY_PUT);     // Pone ventana gráfica 150
                 cuadro(150,BLUE);
                 pon(tope,210,COPY_PUT);     // Pone ventana gráfica 210
                 texto("Presione <ENTER> para seleccionar o <ESC> para salir
...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                 texto("Seleccione el examen que desea cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                 pon(tope,249,COPY_PUT);     // Pone ventana gráfica 249
                 pon(tope,250,COPY_PUT);     // Pone ventana gráfica 250
                 texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                 texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                 pon(tope,249,NOT_PUT);      // Pone ventana gráfica del cursor
                 m = opciones1(249,249,250,32);
                 quita(tope);                // Quita ventana gráfica del cursor
                 quita(tope);                // Quita ventana gráfica 250
                 quita(tope);                // Quita ventana gráfica 249
                 quita(tope);                // Quita ventana gráfica 210
                 quita(tope);                // Quita ventana gráfica 150
                 if (m != -1) then
                  {
                   switch(m)
                        {                    // Pone ventana gráfica 100
                         case 249: pon(tope,100,COPY_PUT);
                                   texto("CAMBIANDO DATOS DE EXAMEN
EXTRAORDINARIO",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                   m = 1;
                                   break;
                         case 250: pon(tope,100,COPY_PUT);
                                   texto("CAMBIANDO DATOS DE EXAMEN
FINAL",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                   m = 2;
                                   break;
                        }
                   k = 0;
                   e = (*r).examen;
                   while (e != NULL) do
                        {
                        if ((*e).tipoexa == m) then
                          k = k + 1;
                        e = (*e).sig;
                        }
                   if (k == 0) then
                        llama_error(42);
                   else
                        {  // Despliega los datos de EXAMEN y efectúa la modificació
                          c = 0;
                          while (c != -2) do
                          {
                            c = mover_pantallas_examen(121,121,231,(*r).examen,m,k);
                            if (c > 0) then
                                {
```

```
                                        if (m == 1) then
                                          cambio_extraord((*r).examen,c);
                                        if (m == 2) then
                                          cambio_final((*r).examen,c);
                                       }
                                  }
                                }
                              quita(tope);        // Quita ventana gráfica 100
                             }
                           }
                        }
                       while (r != NULL) do
                        r = (*r).materias;
                      }
              else
                      {
                       s = p;
                       ultimo = 0;
                       while (s != NULL) do
                        {
                         ultimo = ultimo + 1;
                         s = (*s).materias;
                        }
                       while (dato != -2) do
                        {
                         r = p;
                         dato = mover_pantallas_consulta(120,120,231,p,ultimo);
                         if (dato > 0) then
                               {
                                while ((*r).num != dato) do
                                   r = (*r).materias;
                                if ((*r).examen == NULL) then
                                  llama_error(36);
                                else
                                 {
                                  m = 0;
                                  while (m != -1) do
                                       {
                                        pon(tope,150,COPY_PUT);      // Pone ventana gráfica 150
                                        cuadro(150,BLUE);
                                        pon(tope,210,COPY_PUT);      // Pone ventana gráfica 210
                                        texto("Presione <ENTER> para seleccionar o <ESC> para salir
...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                                        texto("Seleccione el exámen que desea cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
                                        pon(tope,249,COPY_PUT);      // Pone ventana gráfica 249
                                        pon(tope,250,COPY_PUT);      // Pone ventana gráfica 250
                                        texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                                        texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
                                        pon(tope,249,NOT_PUT);       // Pone ventana gráfica del cursor
                                        m = opciones1(249,249,250,32);
                                        quita(tope);       // Quita ventana gráfica del cursor
                                        quita(tope);       // Quita ventana gráfica 250
                                        quita(tope);       // Quita ventana gráfica 249
                                        quita(tope);       // Quita ventana gráfica 210
                                        quita(tope);       // Quita ventana gráfica 150
                                        if (m != -1) then
                                         {
                                          switch(m)
                                                {          // Pone ventana gráfica 100
                                                 case 249: pon(tope,100,COPY_PUT);
                                                           texto("CAMBIANDO DATOS DE EXAMEN
EXTRAORDINARIO",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                                           m = 1;
                                                           break;
                                                 case 250: pon(tope,100,COPY_PUT);
```

```
                                                    texto("CAMBIANDO DATOS DE EXAMEN
FINAL",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                                                    m = 2;
                                                    break;
                                            }
                                        k = 0;
                                        e = (*r).examen;
                                        while (e != NULL) do
                                        {
                                            if ((*e).tipoexa == m) then
                                              k = k + 1;
                                            e = (*e).sig;
                                        }
                                        if (k == 0) then
                                            llama_error(42);
                                        else
                                            { // Despliega los datos de EXAMEN y realiza la modificación
                                              c = 0;
                                              while (c != -2) do
                                              {
                                               c = mover_pantallas_examen(121,121,231,(*r).examen,m,k);
                                               if (c > 0) then
                                               {
                                                    if (m == 1) then
                                                      cambio_extraord((*r).examen,c);
                                                    if (m == 2) then
                                                      cambio_final((*r).examen,c);
                                               }
                                              }
                                            }
                                        quita(tope);      // Quita ventana gráfica 100
                                      }
                                  }
                              }
                            while (r != NULL) do
                              r = (*r).materias;
                          }
                  }
              }
          }
      }
    quita(tope);      // Quita ventana gráfica 100
}


//    Llama a las funciones para realizar las operaciones sobre la lista enlazada EXAMEN.
void examen(int i, struct MATERIA *lista)
{
 cuadro(61,GREEN);
 switch(i)
   {
   case 49: crear_examen(lista);          // Crea una nueva lista enlazada EXAMEN
            break;
   case 50: agregar_examen(lista);        // Añade nuevos nodos a la lista enlazada EXAMEN
            break;
   case 51: eliminar_examen(lista);       // Elimina nodos de la lista enlazada EXAMEN
            break;
   case 52: cambiar_examen(lista);        // Modifica los datos de los nodos de la lista enlazada EXAMEN
            break;
   }
 cuadro(61,LIGHTBLUE);
}
```

B.3.7. E_TODOS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\examen.h"
#include "d:\e_todos.h"
#include "d:\g_todos.h"
#include "d:\consulta.h"
#include "d:\errores.h"
#include "d:\ayuda.h"
#define  then

//     Construye un menú de opciones para la consulta de datos de la lista enlazada EXAMEN.
void cuadros_consulta_examen(int tipo, int b)
{
  switch(tipo)
    {
          case 251: texto("FECHAS",257,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
          case 252: texto("GRUPOS",257,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
    }

  if (b == 3) then
    texto("PROFESOR TITULAR",258,LIGHTBLUE,SMALL_FONT,3,2,2,1);
  else
    texto("MATERIA CORRESPONDIENTE",258,LIGHTBLUE,SMALL_FONT,3,2,2,1);
  texto("PAG. 1",240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                                        // Construye el menú inferior
  cuadro(150,BLUE);
  texto("Seleccione un grupo para obtener más información",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("<TAB> cambia menú",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("PgUp/PgDn",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
}

//  Realiza la operación de modificación de los datos de un nodo de la lista enlazada EXAMEN.
void cambio_extraord(struct EXAMEN *e, int dato)
{
 struct EXAMEN *r;
 char nombreprofe[100], rfc[100], hora1[100], profe[100], rfc2[100], semestre[100], numero[100];
 int i, j, c, regresa, numgrupo, k, tipo, dia1, dia2, mes1, mes2, ano1, ano2, tipoprofe, tiposalon;
 long int numsalon;

 r=e;
 while ((*r).num != dato)                       // Se recorre la lista hasta que apunte a dato
         r = (*r).sig;
 ventana_cambia_datos_materia();                // Construye un menú inferior
                                                // Construye el menú de opciones
 texto("Seleccione el módulo que desee cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
 texto("PROFESOR TITULAR",211,LIGHTBLUE,SMALL_FONT,1,1,2,1);
 texto("RFC (PROF. TIT.)",212,LIGHTBLUE,SMALL_FONT,1,1,2,1);
```

```
texto("PROFESOR SUPLENTE",213,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("RFC (PROF.SUPL.)",214,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("SALON",215,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("TIPO DE SALON",216,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("GRUPO",217,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("HORA",218,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("DIA",219,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("MES",220,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("A&O",221,LIGHTBLUE,SMALL_FONT,1,1,2,1);
i = 211;
j = 1;
pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
while(j == 1)
 {
  c = biosKey(0);
  switch(c)
   {
        case F1: llama_ayuda(33);       // Llama al módulo AYUDA
                 break;
        case DERECHA:  quita(tope);     // Desplaza el cursor sobre el menú de opciones
                       i++;
                       if (i == 222) then
                          i = 211;
                       pon(tope,i,NOT_PUT);
                       break;
        case IZQUIERDA: quita(tope);
                       i--;
                       if (i == 210) then
                          i = 221;
                       pon(tope,i,NOT_PUT);
                       break;
        case TAB:      if ((i >= 211) && (i <= 221)) then     // Cambia el cursor al menú inferior
                        {
                         regresa=i;
                         i=152;
                         i=mover_ventana_menu(i,i,153,regresa);
                         if (i == 152) then
                           i=regresa;
                         if (i == 153) then
                               j = 0;
                        }
                       break;
        case ENTER: quita(tope);
                    switch(i)
                     {
                          case 211: // Lee el nombre del profesor titular
                                    pon(tope,171,COPY_PUT);  // Pone ventana gráfica 171
                                    pon(tope,172,COPY_PUT);  // Pone ventana gráfica
                                    texto("Nombre del profesor titular actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                                    texto((*r),nombreprofe,172,BLUE,SMALL_FONT,1,1,2,1);
                                    pon(tope,173,COPY_PUT);  // Pone ventana gráfica 173
                                    pon(tope,174,COPY_PUT);  // Pone ventana gráfica 174
                                    texto("Nuevo nombre del profesor . . .",173,BLUE,SMALL_FONT,1,1,2,1);
                                          // Lee una cadena del teclado
                                    lee_cadena(nombreprofe,32,174,RED,SMALL_FONT,1,1,2,1);
                                    texto(nombreprofe,174,RED,SMALL_FONT,1,1,2,1);
                                    pon(tope,179,COPY_PUT);  // Pone ventana gráfica 179
                                    texto("Presione una tecla para continuar",179, LIGHTRED,SMALL_FONT,1,1,2,1);
                                    biosKey(0);
                                    quita(tope);          // Quita ventana gráfica 179
                                    quita(tope);          // Quita ventana gráfica 174
                                    quita(tope);          // Quita ventana gráfica 173
                                    quita(tope);          // Quita ventana gráfica 172
                                    quita(tope);          // Quita ventana gráfica 171
                                    for (k=0; nombreprofe[k]; k++)
```

```
                // Cambia nombreprofe a mayúsculas
            nombreprofe[k] = toupper(nombreprofe[k]);
                // Asigna el nuevo dato al nodo r
            strcpy((*r).nombreprofe,nombreprofe);
            // Despliega el nombramiento del profesor actual
            pon(tope,171,COPY_PUT);                 // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);                 // Pone ventana gráfica 172
            texto("Nombramiento actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            c = (*r).tipoprofe; // Selecciona el tipo de nombramiento actual
            switch(c)
             {
                case 0: texto("No se asigno profesor",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 1: texto("Profesor Titular",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 2: texto("Ayudante de profesor",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 3: texto("Sin nombramiento",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
             }
                        // Lee el nuevo nombramiento del profesor
            pon(tope,173,COPY_PUT);                 // Pone ventana gráfica 173
            texto("Nuevo nombramiento . . .",173,RED,SMALL_FONT,1,1,2,1);
            pon(tope,193,COPY_PUT);                 // Pone ventana gráfica 193
            pon(tope,194,COPY_PUT);                 // Pone ventana gráfica 194
            texto("TITULAR",193,RED,SMALL_FONT,1,1,2,1);
            texto("AYUDANTE",194,RED,SMALL_FONT,1,1,2,1);
            pon(tope,193,NOT_PUT);      // Pone ventana gráfica del cursor
                // Selección del nuevo nombramiento del profesor
            i = opciones1(193,193,194,18);
            quita(tope);            // Quita ventana gráfica del cursor
            quita(tope);            // Quita ventana gráfica 194
            quita(tope);            // Quita ventana gráfica 193
            pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
            if (i == 193) then    // Asigna el nuevo nombramiento al nodo r
                    {
                     texto("PROFESOR TITULAR",174,RED, SMALL_FONT,1,1,2,1);
                     tipoprofe = 1;
                    }
            if (i == 194) then
                    {
                     texto("AYUDANTE DE PROFESOR",174,RED,SMALL_FONT,1,1,2,1);
                     tipoprofe = 2;
                    }
            if (i == -1) then
                    {
                     texto("SIN NOMBRAMIENTO",174,RED,SMALL_FONT,1,1,2,1);
                     tipoprofe = 3;
                    }
            pon(tope,179,COPY_PUT);                 // Pone ventana gráfica 179
            texto("Presione una tecla para continuar", 179,LIGHTRED,SMALL_FONT,1,1,2,1);
            biosekey(0);
            quita(tope);            // Quita ventana gráfica 179
            quita(tope);            // Quita ventana gráfica 174
            quita(tope);            // Quita ventana gráfica 173
            quita(tope);            // Quita ventana gráfica 172
            quita(tope);            // Quita ventana gráfica 171
            (*r).tipoprofe = tipoprofe;
            i=211;
            pon(tope,i,NOT_PUT);            // Pone ventana gráfica del cursor
            break;

        case 212: // Despliega en pantalla el R.F.C. actual del profesor titular
            pon(tope,171,COPY_PUT);                 // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);                 // Pone ventana gráfica 172
```

```
                texto("RFC del profesor titular actual. . .",171, BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).rfc,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee el nuevo R.F.C. del profesor
                pon(tope,173,COPY_PUT);                 // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
                texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                 // Lee una cadena del teclado
                lee_cadena(rfc,12,174,RED,SMALL_FONT,1,1,2,1);
                texto(rfc,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);                 // Pone ventana gráfica 179
                texto("Presione una tecla para continuar", 179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                for (k=0; rfc[k]; k++)      // Cambia la cadena rfc a mayúsculas
                rfc[k] = toupper(rfc[k]);
                strcpy((*r).rfc,rfc);           // Asigna rfc al nodo r
                i=212;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

case 213: // Despliega Nombre del profesor suplente
                pon(tope,171,COPY_PUT);                 // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);                 // Pone ventana gráfica 172
                texto("Nombre del profesor suplente actual. . .",171, BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).profe,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee el nuevo nombre del profesor
                pon(tope,173,COPY_PUT);                 // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
                texto("Nuevo nombre del profesor . . .",173, BLUE,SMALL_FONT,1,1,2,1);
                // Lee una cadena del teclado
                lee_cadena(nombreprofe,32,174,RED,SMALL_FONT,1,1,2,1);
                texto(profe,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);                 // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179, LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                for (k=0; profe[k]; k++)
                profe[k] = toupper(profe[k]);
                strcpy((*r).profe,profe);       // Asigna profe al nodo r

                // Despliega en pantalla el nombramiento actual del profesor
                pon(tope,171,COPY_PUT);                 // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);                 // Pone ventana gráfica 172
                texto("Nombramiento actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                c = (*r).tipo;      // Selección del nombramiento actual
                switch(c)
                 {
                        case 0: texto("No se asigno profesor",172, BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 1: texto("Profesor Titular",172, BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 2: texto("Ayudante de profesor",172, BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 3: texto("Sin nombramiento",172, BLUE,SMALL_FONT,1,1,2,1);
                                break;
                 }
                // Lee el nuevo nombramiento del profesor
```

```
                pon(tope,173,COPY_PUT);                 // Pone ventana gráfica 173
                texto("Nuevo nombramiento . . .",173,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,COPY_PUT);                 // Pone ventana gráfica 193
                pon(tope,194,COPY_PUT);                 // Pone ventana gráfica 194
                texto("TITULAR",193,RED,SMALL_FONT,1,1,2,1);
                texto("AYUDANTE",194,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,NOT_PUT);      // Pone ventana gráfica del cursor
                    // Selección del nuevo nombramiento del profesor
                i = opciones1(193,193,194,15);
                quita(tope);            // Quita ventana gráfica del cursor
                quita(tope);            // Quita ventana gráfica 194
                quita(tope);            // Quita ventana gráfica 193
                pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
                if (i == 193) then          // Asigna el nuevo nombramiento al nodo r
                        {
                         texto("PROFESOR TITULAR",174, RED,SMALL_FONT,1,1,2,1);
                         tipo = 1;
                        }
                if (i == 194) then
                        {
                         texto("AYUDANTE DE PROFESOR",174, RED,SMALL_FONT,1,1,2,1);
                         tipo = 2;
                        }
                if (i == -1) then
                        {
                         texto("SIN NOMBRAMIENTO",174, RED,SMALL_FONT,1,1,2,1);
                         tipo = 3;
                        }
                pon(tope,179,COPY_PUT);                 // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                (*r).tipo = tipo;
                i=213;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 214: // Despliega en pantalla el actual R.F.C. del profesor suplente
                pon(tope,171,COPY_PUT);                 // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);                 // Pone ventana gráfica 172
                texto("RFC del profesor suplente actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).rfc2,172,BLUE,SMALL_FONT,1,1,2,1);
                pon(tope,173,COPY_PUT);                 // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
                texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                    // Lee una cadena del teclado
                lee_cadena(rfc2,12,174,RED,SMALL_FONT,1,1,2,1);
                texto(rfc2,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);                 // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                for (k=0; rfc2[k]; k++)     // Cambia la cadena rfc2 a mayúsculas
                rfc2[k] = toupper(rfc2[k]);
                strcpy((*r).rfc2,rfc2);         // Asigna el rfc2 al nodo r
                i=214;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
```

```
            break;

case 215: // Despliega en pantalla el actual número de salón */
            pon(tope,171,COPY_PUT);              // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);              // Pone ventana gráfica 172
            texto("Número de salón actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            // Convierta el entero largo en una cadena
             itoa((*r).numsalon,numero,10);
            texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
            // Lee el nuevo número de salón
            pon(tope,173,COPY_PUT);              // Pone ventana gráfica 173
            pon(tope,174,COPY_PUT);              // Pone ventana gráfica 174
            texto("Nuevo número de salón . . .",173,RED,SMALL_FONT,1,1,2,1);
             // Lee una cadena del teclado
            lee_cadena(numero,5,174,RED,SMALL_FONT,1,1,2,1);
            texto(numero,174,RED,SMALL_FONT,1,1,2,1);
            // Convierta la cadena numero en un entero largo
            numsalon = atol(numero);
            pon(tope,179,COPY_PUT);              // Pone ventana gráfica 179
            texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
            bioskey(0);
            quita(tope);          // Quita ventana gráfica 179
            quita(tope);          // Quita ventana gráfica 174
            quita(tope);          // Quita ventana gráfica 173
            quita(tope);          // Quita ventana gráfica 172
            quita(tope);          // Quita ventana gráfica 171
            (*r).numsalon = numsalon; // Asigna numsalon al nod r
            i=215;
            pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
            break;

case 216: // Despliega en pantalla el actual tipo de salón
            pon(tope,171,COPY_PUT);              // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);              // Pone ventana gráfica 172
            texto("Tipo de salón actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            c = (*r).tiposalon;
            switch(c)
             {
               case 0: texto ("No se asignó salón",172,BLUE,SMALL_FONT,1,1,2,1);
                          break;
               case 1: texto ("Salón de dibujo",172,BLUE,SMALL_FONT,1,1,2,1);
                          break;
               case 2: texto ("Salón de laboratorio",172,BLUE, SMALL_FONT,1,1,2,1);
                          break;
               case 3: texto ("Salón chico (máx. 30 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
                          break;
               case 4: texto ("Salón grande (máx. 60 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
                          break;
                  case 5: texto ("Salon sin clasificar",172,BLUE, SMALL_FONT,1,1,2,1);
                          break;
             }
            // Lee el nuevo tipo de salón
            pon(tope,173,COPY_PUT);              // Pone ventana gráfica 173
            texto("Nuevo tipo de salón",173,RED,SMALL_FONT,1,1,2,1);
            pon(tope,175,COPY_PUT);              // Pone ventana gráfica 175
            pon(tope,176,COPY_PUT);              // Pone ventana gráfica 176
            pon(tope,177,COPY_PUT);              // Pone ventana gráfica 177
            pon(tope,178,COPY_PUT);              // Pone ventana gráfica 178
            texto("D",175,RED,SMALL_FONT,1,1,2,1);
            texto("L",176,RED,SMALL_FONT,1,1,2,1);
            texto("C",177,RED,SMALL_FONT,1,1,2,1);
            texto("G",178,RED,SMALL_FONT,1,1,2,1);
            pon(tope,175,NOT_PUT);       // Pone ventana gráfica del cursor
            i = opciones1(175,175,178,24);  // Selecciona el nuevo tipo de salón
            quita(tope);          // Quita ventana gráfica del cursor
```

```
                quita(tope);            // Quita ventana gráfica 178
                quita(tope);            // Quita ventana gráfica 177
                quita(tope);            // Quita ventana gráfica 176
                quita(tope);            // Quita ventana gráfica 175
                pon(tope,174,COPY_PUT);                 // Pone ventana gráfica 174
                if (i == 175) then                      // Selección del tipo de salón
                  {
                   texto("Salón de dibujo",174,RED,SMALL_FONT,1,1,2,1);
                   tiposalon = 1;
                  };
                if (i == 176) then
                  {
                   texto("Salón de laboratorio",174,RED,SMALL_FONT,1,1,2,1);
                   tiposalon = 2;
                  };
                 if (i == 177) then
                  {
                   texto("Salón chico (máx. 30 alumnos)",174,RED, SMALL_FONT,1,1,2,1);
                   tiposalon = 3;
                  };
                 if (i == 178) then
                  {
                   texto("Salón grande (máx. 60 alumnos)",174,RED,SMALL_FONT,1,1,2,1);
                   tiposalon = 4;
                  };
                 if (i == -1) then
                  {
                   texto("Salón sin clasificar",174,RED,SMALL_FONT,1,1,2,1);
                   tiposalon = 5;
                  };
                pon(tope,179,COPY_PUT);         // Pone ventana gráfica 179
                texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                (*r).tiposalon = tiposalon;             // Asigna tiposalon al nodo r
                i = 216;
                pon(tope,i,NOT_PUT);            // Pone ventana gráfica del cursor
                break;

        case 217: // Despliega en pantalla el grupo acutal
                pon(tope,171,COPY_PUT);         // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);         // Pone ventana gráfica 172
                texto("Número de grupo actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                itoa((*r).numgrupo,numero,10);
                texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee el nuevo número de grupo
                pon(tope,173,COPY_PUT);         // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);         // Pone ventana gráfica 174
                texto("Nuevo número de grupo . . .",173,RED,SMALL_FONT,1,1,2,1);
                // Lee una cadena del teclado
                lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                numgrupo=atoi(numero);
                pon(tope,179,COPY_PUT);         // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
```

```
                (*r).numgrupo = numgrupo;                    // Asigna numgrupo al nodo r
                i=217;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

case 218: // Despliega en pantalla la hora actual
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Hora actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).hora1,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee la nue va hora
                pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                texto("Nueva hora (hh:mm) . . .",173,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) do                    // Verifica la hora correcta
                 {
                  lee_cadena(hora1,5,174,RED,SMALL_FONT,1,1,2,1);
                  k = hora_correcta(hora1,hora1,1);
                 }
                texto(hora1,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 172
                quita(tope);        // Quita ventana gráfica 171
                strcpy((*r).hora1,hora1);                    // Asigna hora1 al nodo r
                i=218;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

case 219: // Despliega en pantalla el día actual
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Dia actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                itoa((*r).dia1,numero,10);
                texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee del teclado el nuevo día
                pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                texto("Nuevo dia (dd). . .",173,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) do            // Verifica que el dato sea correcto
                 {
                  lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
                  k = error_dia(numero);
                 }
                texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                dia1 = atoi(numero);
                pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED, SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 172
                quita(tope);        // Quita ventana gráfica 171
                (*r).dia1 = dia1;                    // Asigna dia1 al nodo r
                i=219;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;
```

```
case 220: // Despliega en pantalla el mes actual
        pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
        pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
        texto("Mes actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
        itoa((*r).mes1,numero,10);
        texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
        // Lee del teclado del nuevo dato de mes
        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
        texto("Nuevo mes (mm). . .",173,RED,SMALL_FONT,1,1,2,1);
        k = 1;
        while (k == 1) do              // Verifica que el dato sea correcto
        {
         lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
         k = error_mes(numero);
        }
        texto(numero,174,RED,SMALL_FONT,1,1,2,1);
        mes1 = atoi(numero);
        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
        texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
        bioskey(0);
        quita(tope);       // Quita ventana gráfica  179
        quita(tope);       // Quita ventana gráfica  174
        quita(tope);       // Quita ventana gráfica  173
        quita(tope);       // Quita ventana gráfica  172
        quita(tope);       // Quita ventana gráfica  171
        (*r).mes1 = mes1;              // Asigna mes1 al nodo r
        i=220;
        pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
        break;

case 221: // Despliega en pantalla el año actual
        pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
        pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
        texto("Año actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
        itoa((*r).ano1,numero,10);
        texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
        // Lee del teclado el nuevo año
        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
        texto("Nuevo año (aaaa). . .",173,RED,SMALL_FONT,1,1,2,1);
        k = 1;
        while (k == 1) do              // Verifica que el dato sea correcto
        {
         lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
         k = error_ano(numero);
        }
        texto(numero,174,RED,SMALL_FONT,1,1,2,1);
        ano1 = atoi(numero);
        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
        texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
        bioskey(0);
        quita(tope);       // Quita ventana gráfica  179
        quita(tope);       // Quita ventana gráfica  174
        quita(tope);       // Quita ventana gráfica  173
        quita(tope);       // Quita ventana gráfica  172
        quita(tope);       // Quita ventana gráfica  171
        (*r).ano1 = ano1;              // Asigna ano1 al nodo r
        i=221;
        pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
        break;
    }
  }
}
```

```
quita(tope);        // Quita ventana gráfica del cursor
 for (k=210; k <= 221; k++)
    {
         setfillstyle(SOLID_FILL,GREEN);
         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
 for (k=152; k <= 154; k++)
    {
         setfillstyle(SOLID_FILL,GREEN);
         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
quita(tope);        // Quita ventana gráfica  150
}

//    Realiza la operación de modificación en los datos del tipo de exámen final, de la lista enlazada EXAMEN.
void cambio_final(struct EXAMEN *e, int dato)
{
struct EXAMEN *r;
char nombreprofe[100], rfc[100], hora1[100], profe[100], rfc2[100], numero[100];
int i, j, c, regresa, numgrupo, k, tipo, dia1, dia2, mes1, mes2, ano1, ano2, tpoprofe, tiposalon;
long int numsalon;

r=e;
while ((*r).num != dato)
        r = (*r).sig;
ventana_cambia_datos_materia();        // Construye el menú inferior
texto("Seleccione el modulo que desee cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("PROFESOR",211,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("RFC",212,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("NOMBRAMIENTO",213,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("SALON",214,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("TIPO DE SALON",215,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("GRUPO",216,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("HORARIO",217,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("FECHA 1a.vuelta",218,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("FECHA 2a.vuelta",219,LIGHTBLUE,SMALL_FONT,1,1,2,1);
i = 211;
j = 1;
pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
while(j == 1)
 {
  c = bioskey(0);
  switch(c)
   {
        case F1: llama_ayuda(34);   // Llama al módulo AYUDA
                 break;
        case DERECHA:  quita(tope);                    // Quita ventana gráfica del cursor
                       i++;                            // Desplaza el cursor sobre el menú de opciones
                       if (i == 220) then
                          i = 211;
                       pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                       break;
        case IZQUIERDA: quita(tope);                   // Quita ventana gráfica del cursor
                       i--;
                       if (i == 210) then
                          i = 219;
                       pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                       break;
        case TAB:      if ((i >= 211) && (i <= 219)) then          // Cambia al menú inferior
                        {
                         regresa=i;
                         i=152;
                         :=mover_ventana_menu(i,i,152,regresa);
                         if (i == 152) then
                                 j = 0;
```

```
                }
                break;
case ENTER: quita(tope);        // Quita ventana gráfica del cursor
            switch(i)
             {
                case 211: // Despliega en pantalla el actual nombre del profesor
                          pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
                          pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
                          texto("Nombre del profesor actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                          texto((*r).nombreprofe,172,BLUE,SMALL_FONT,1,1,2,1);
                          // Lee del teclado el nuevo nombre del profesor
                          pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
                          pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
                          texto("Nuevo nombre del profesor . . .",173,BLUE,SMALL_FONT,1,1,2,1);
                                // Lee una cadena del teclado
                          lee_cadena(nombreprofe,32,174,RED,SMALL_FONT,1,1,2,1);
                          texto(nombreprofe,174,RED,SMALL_FONT,1,1,2,1);
                          pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                          texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                          bioskey(0);
                          quita(tope);      // Quita ventana gráfica  179
                          quita(tope);      // Quita ventana gráfica  174
                          quita(tope);      // Quita ventana gráfica  173
                          quita(tope);      // Quita ventana gráfica  172
                          quita(tope);      // Quita ventana gráfica  171
                          for (k=0; nombreprofe[k]; k++)      // Cambia a mayúsculas la cadena nombreprofe
                          nombreprofe[k] = toupper(nombreprofe[k]);
                          strcpy((*r).nombreprofe,nombreprofe);          // Asigna nombreprofe al nodo r
                          : = 211;
                          pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                          break;

                case 212: // Despliega en pantalla el actual R.F.C. del profesor titular
                          pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
                          pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
                          texto("RFC del profesor actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                          texto((*r).rfc,172,BLUE,SMALL_FONT,1,1,2,1);
                          // Lee el nuevo R.F.C. del profesor
                          pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
                          pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
                          texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                          lee_cadena(rfc,12,174,RED,SMALL_FONT,1,1,2,1);      // Lee una cadena del teclado
                          texto(rfc,174,RED,SMALL_FONT,1,1,2,1);
                          pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                          texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                          bioskey(0);
                          quita(tope);      // Quita ventana gráfica  179
                          quita(tope);      // Quita ventana gráfica  174
                          quita(tope);      // Quita ventana gráfica  173
                          quita(tope);      // Quita ventana gráfica  172
                          quita(tope);      // Quita ventana gráfica  171
                          for (k=0; rfc[k]; k++)
                          rfc[k] = toupper(rfc[k]);
                          strcpy((*r).rfc,rfc);          // Asigna rfc al nodo r
                          i=212;
                          pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                          break;

                case 213: // Despliega en pantalla el nombramiento del profesor
                          pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
                          pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
                          texto("Nombramiento actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                          o = (*r).tpoprofe;                 // Selecciona el nombramiento actual
                          switch(o)
                           {
```

```
                case 0: texto("No se asignó profesor",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 1: texto("Profesor Titular",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 2: texto("Ayudante de profesor",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
                case 3: texto("Sin nombramiento",172,BLUE,SMALL_FONT,1,1,2,1);
                        break;
              }
            // Lee del teclado el nuevo nombramiento del profesor
            pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
            texto("Nuevo nombramiento . . .",173,RED,SMALL_FONT,1,1,2,1);
            pon(tope,193,COPY_PUT);       // Pone ventana gráfica 193
            pon(tope,194,COPY_PUT);       // Pone ventana gráfica 194
            texto("TITULAR",193,RED,SMALL_FONT,1,1,2,1);
            texto("AYUDANTE",194,RED,SMALL_FONT,1,1,2,1);
            pon(tope,193,NOT_PUT);        // Pone ventana gráfica del cursor
            i = opciones1 (193,193,194,16);       // Selecciona del menú de opciones
            quita(tope);      // Quita ventana gráfica  del cursor
            quita(tope);      // Quita ventana gráfica  194
            quita(tope);      // Quita ventana gráfica  193
            pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
            if (i == 193) then                  // Asigna tipoprofe al nodo r
                    {
                     texto("PROFESOR TITULAR",174,RED,SMALL_FONT,1,1,2,1);
                     tipoprofe = 1;
                    }
            if (i == 194) then
                    {
                     texto("AYUDANTE DE PROFESOR",174,RED,SMALL_FONT,1,1,2,1);
                     tipoprofe = 2;
                    }
            if (i == -1) then
                    {
                     texto("SIN NOMBRAMIENTO",174,RED,SMALL_FONT,1,1,2,1);
                     tipoprofe = 3;
                    }
            pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
            texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
            bioskey(0);
            quita(tope);      // Quita ventana gráfica  179
            quita(tope);      // Quita ventana gráfica  174
            quita(tope);      // Quita ventana gráfica  173
            quita(tope);      // Quita ventana gráfica  172
            quita(tope);      // Quita ventana gráfica  171
            (*r).tipoprofe = tipoprofe;
            i=213;
            pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
            break;

    case 214: // Despliega en pantalla el actual número de salón
            pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
            texto("Número de salón actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            itoa((*r).numsalon,numero,10);
            texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
            // Lee del teclado el nuevo número de salón
            pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
            pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
            texto("Nuevo número de salón . . .",173,RED,SMALL_FONT,1,1,2,1);
            k = 1;
            while (k == 1) do                   // Verifica que el dato sea correcto
             {
              lee_cadena(numero,5,174,RED,SMALL_FONT,1,1,2,1);
              k = error_numero_entero(numero);
```

```
        }
        texto(numero,174,RED,SMALL_FONT,1,1,2,1);
        numsalon = atoi(numero);
        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
        texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
        bioskey(0);
        quita(tope);        // Quita ventana gráfica 179
        quita(tope);        // Quita ventana gráfica 174
        quita(tope);        // Quita ventana gráfica 173
        quita(tope);        // Quita ventana gráfica 172
        quita(tope);        // Quita ventana gráfica 171
        (*r).numsalon = numsalon;              // Asigna numsalon al nodo r
        i=214;
        pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
        break;

case 215: // Despliega en pantalla el actual tipo de salón
        pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
        pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
        texto("Tipo de salón actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
        c = (*r).tiposalon;                    // Selecciona el tipo de salón actual
        switch(c)
         {
           case 0: texto ("No se asignó salón",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
           case 1: texto ("Salón de dibujo",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
           case 2: texto ("Salón de laboratorio",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
           case 3: texto ("Salón chico (máx. 30 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
           case 4: texto ("Salón grande (máx. 60 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
           case 5: texto ("Salón sin clasificar",172,BLUE,SMALL_FONT,1,1,2,1);
                   break;
         }
        // Lee del teclado el nuevo tipo de salón
        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
        texto("Nuevo tipo de salón",173,RED,SMALL_FONT,1,1,2,1);
        pon(tope,175,COPY_PUT);        // Pone ventana gráfica 175
        pon(tope,176,COPY_PUT);        // Pone ventana gráfica 176
        pon(tope,177,COPY_PUT);        // Pone ventana gráfica 177
        pon(tope,178,COPY_PUT);        // Pone ventana gráfica 178
        texto("D",175,RED,SMALL_FONT,1,1,2,1);
        texto("L",176,RED,SMALL_FONT,1,1,2,1);
        texto("C",177,RED,SMALL_FONT,1,1,2,1);
        texto("G",178,RED,SMALL_FONT,1,1,2,1);
        pon(tope,175,NOT_PUT);        // Pone ventana gráfica del cursor
        i = opciones1(175,175,178,24);        // Selecciona el tipo de salón
        quita(tope);        // Quita ventana gráfica del cursor
        quita(tope);        // Quita ventana gráfica 178
        quita(tope);        // Quita ventana gráfica 177
        quita(tope);        // Quita ventana gráfica 176
        quita(tope);        // Quita ventana gráfica 175
        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
        if (i == 175) then                     // Asigna tiposalon al nodo r
         {
          texto("Salón de dibujo",174,RED,SMALL_FONT,1,1,2,1);
          tiposalon = 1;
         };
        if (i == 176) then
         {
          texto("Salón de laboratorio",174,RED,SMALL_FONT,1,1,2,1);
          tiposalon = 2;
         };
```

```
                if (i == 177) then
                 {
                  texto("Salón chico (máx. 30 alumnos)",174,RED,SMALL_FONT,1,1,2,1);
                  tiposalon = 3;
                 };
                if (i == 178) then
                 {
                  texto("Salón grande (máx. 60 alumnos)",174,RED,SMALL_FONT,1,1,2,1);
                  tiposalon = 4;
                 };
                if (i == -1) then
                 {
                  texto("Salón sin clasificar",174,RED,SMALL_FONT,1,1,2,1);
                  tiposalon = 5;
                 };
                pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 172
                quita(tope);        // Quita ventana gráfica 171
                (*r).tiposalon = tiposalon;
                i = 215;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 216: // Despliega en pantalla el grupo actual
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Número de grupo actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                itoa((*r).numgrupo,numero,10);
                texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee del teclado el nuevo grupo
                pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                texto("Nuevo número de grupo . . .",173,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) do                // Verifica que el dato sea correcto
                 {
                  lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                  k = error_grupoesta_cambiar(e,numero,dato);
                 }
                texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                numgrupo=atoi(numero);
                pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 172
                quita(tope);        // Quita ventana gráfica 171
                (*r).numgrupo = numgrupo;        // Asigna numgrupo al nodo r
                i=216;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 217: // Despliega en pantalla la hora actual
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Hora actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).hora1,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee del teclado la nueva hora
```

```
                        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                        texto("Nueva hora  (hh:mm). . .",173,RED,SMALL_FONT,1,1,2,1);
                        k = 1;
                        while (k == 1) do                    // Verifica que el dato sea correcto
                        {
                          lee_cadena(hora1,5,174,RED,SMALL_FONT,1,1,2,1);
                          k = hora_correcta(hora1,hora1,1);
                        }
                        texto(hora1,174,RED,SMALL_FONT,1,1,2,1);
                        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                        texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                        bioskey(0);
                        quita(tope);        // Quita ventana gráfica  179
                        quita(tope);        // Quita ventana gráfica  174
                        quita(tope);        // Quita ventana gráfica  173
                        quita(tope);        // Quita ventana gráfica  172
                        quita(tope);        // Quita ventana gráfica  171
                        for (k=0; hora1[k]; k++)
                        hora1[k] = toupper(hora1[k]);
                        strcpy((*r).hora1,hora1);    // Asigna hora1 al nodo r
                        i=217;
                        pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                        break;

                case 216: // Despliega en pantalla el menú de opciones Fecha
                        texto("Seleccione...",180,BLUE,SMALL_FONT,1,1,2,1);
                        texto("DIA",182,RED,SMALL_FONT,1,1,2,1);
                        texto("MES",183,RED,SMALL_FONT,1,1,2,1);
                        texto("AÑO",184,RED,SMALL_FONT,1,1,2,1);
                        texto("1a. vuelta",185,BLUE,SMALL_FONT,1,1,2,1);
                        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                        texto("Presione <ESC> cuando termine de
seleccionar",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                        pon(tope,182,NOT_PUT);        // Pone ventana gráfica del cursor
                        i = 182;
                        while (i != -1) do
                        {
                            i = opciones1(i,182,184,29);
                            switch(i)
                            {
                              case 182: quita(tope);        // Quita ventana gráfica del cursor
                                        pon(tope,173,COPY_PUT);        // Pone ventana gráfica  173
                                        pon(tope,174,COPY_PUT);        // Pone ventana gráfica  174
                                        texto("Dia  (dd) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                                        k = 1;
                                        while (k == 1) do          // Verifica que el dato sea correcto
                                        {
                                            lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
                                            k = error_dia(numero);
                                        }
                                        texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                                        dia1 = atoi(numero);
                                        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                                        texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                        bioskey(0);
                                        quita(tope);        // Quita ventana gráfica  179
                                        quita(tope);        // Quita ventana gráfica  174
                                        quita(tope);        // Quita ventana gráfica  173
                                        (*r).dia1 = dia1;                    // Asigna dia1 al nodo r
                                        i = 182;
                                        pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                                        break;
```

```
                    case 183: quita(tope);      // Quita ventana gráfica del cursor
                              pon(tope,173,COPY_PUT);      // Pone ventana gráfica 173
                              pon(tope,174,COPY_PUT);      // Pone ventana gráfica 174
                              texto("Mes  (mm) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                              k = 1;
                              while (k == 1) do        // Verifica que el dato sea correcto
                               {
                                   lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
                                   k = error_mes(numero);
                               }
                              texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                              mes1 = atoi(numero);
                              pon(tope,179,COPY_PUT);      // Pone ventana gráfica 179
                              texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                              bioskey(0);
                              quita(tope);      // Quita ventana gráfica  179
                              quita(tope);      // Quita ventana gráfica  174
                              quita(tope);      // Quita ventana gráfica  173
                              (*r).mes1 = mes1;        // Asigna mes1 al nodo r
                              i = 183;
                              pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                              break;

                    case 184: quita(tope);      // Quita ventana gráfica del cursor
                              pon(tope,173,COPY_PUT);      // Pone ventana gráfica 173
                              pon(tope,174,COPY_PUT);      // Pone ventana gráfica 174
                              texto("Año  (aaaa) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                              k = 1;
                              while (k == 1) do        // Verifica que el dato sea correcto
                               {
                                   lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                                   k = error_ano(numero);
                               }
                              texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                              ano1 = atoi(numero);
                              pon(tope,179,COPY_PUT);      // Pone ventana gráfica 179
                              texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                              bioskey(0);
                              quita(tope);      // Quita ventana gráfica  179
                              quita(tope);      // Quita ventana gráfica  174
                              quita(tope);      // Quita ventana gráfica  173
                              (*r).ano1 = ano1;        // Asigna ano1 al nodo r
                              i = 184;
                              pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                              break;
                     }
                    quita(tope);      // Quita ventana gráfica del cursor
                    quita(tope);      // Quita ventana gráfica  179
                    pon(tope,179,COPY_PUT);      // Pone ventana gráfica 179
                    texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                    bioskey(0);
                    quita(tope);      // Quita ventana gráfica  179
                    k = 180;
                          setfillstyle(SOLID_FILL,GREEN);
                          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                    for (k=182; k<=185; k++)
                     {
                          setfillstyle(SOLID_FILL,GREEN);
                          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                     }
                    i = 216;
                    pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                    break;
```

```
                    case 219: // Despliega el menú de opciones de Fecha
                              texto("Seleccione...",180,BLUE,SMALL_FONT,1,1,2,1);
                              texto("DIA",182,RED,SMALL_FONT,1,1,2,1);
                              texto("MES",183,RED,SMALL_FONT,1,1,2,1);
                              texto("A&O",184,RED,SMALL_FONT,1,1,2,1);
                              texto("2a. vuelta",185,BLUE,SMALL_FONT,1,1,2,1);
                              pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                              texto("Presione <ESC> cuando termine de
seleccionar",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                              pon(tope,182,NOT_PUT);        // Pone ventana gráfica del cursor
                              i = 182;
                              while (i != -1) do
                               {
                                    i = opciones1(i,182,184,29);  // Selecciona una opción del menú
                                    switch(i)
                                     {
                                       case 182: quita(tope);       // Quita ventana gráfica del cursor
                                                 pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
                                                 pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
                                                 texto("Dia  (dd) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                                                 k = 1;
                                                 while (k == 1) do         // Verifica que el dato sea correcto
                                                  {
                                                       lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
                                                       k = error_dia(numero);
                                                  }
                                                 texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                                                 dia2 = atoi(numero);
                                                 pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                                                 texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                 bioskey(0);
                                                 quita(tope);       // Quita ventana gráfica 179
                                                 quita(tope);       // Quita ventana gráfica 174
                                                 quita(tope);       // Quita ventana gráfica 173
                                                 (*r).dia2 = dia2;          // Asigna dia2 al nodo r
                                                 i = 182;
                                                 pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                                                 break;

                                       case 183: quita(tope);       // Quita ventana gráfica del cursor
                                                 pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
                                                 pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
                                                 texto("Mes  (mm) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                                                 k = 1;
                                                 while (k == 1) do         // Verifica que el dato sea correcto
                                                  {
                                                       lee_cadena(numero,2,174,RED,SMALL_FONT,1,1,2,1);
                                                       k = error_mes(numero);
                                                  }
                                                 texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                                                 mes2 = atoi(numero);
                                                 pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                                                 texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                 bioskey(0);
                                                 quita(tope);       // Quita ventana gráfica 179
                                                 quita(tope);       // Quita ventana gráfica 174
                                                 quita(tope);       // Quita ventana gráfica 173
                                                 (*r).mes2 = mes2;          // Asigna mes2 al nodo r
                                                 i = 183;
                                                 pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                                                 break;

                                      case 184: quita(tope);       // Quita ventana gráfica del cursor
```

```
                                                pon(tope,173,COPY_PUT);      // Pone ventana gráfica 173
                                                pon(tope,174,COPY_PUT);      // Pone ventana gráfica 174
                                                texto("Año  (aaaa) . . . .",173,RED,SMALL_FONT,1,1,2,1);
                                                k = 1;
                                                while (k == 1) do        // Verifica que el dato sea correcto
                                                 {
                                                      lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                                                      k = error_ano(numero);
                                                 }
                                                texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                                                ano2 = atoi(numero);
                                                pon(tope,179,COPY_PUT);      // Pone ventana gráfica 179
                                                texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                bioskey(0);
                                                quita(tope);      // Quita ventana gráfica  179
                                                quita(tope);      // Quita ventana gráfica  174
                                                quita(tope);      // Quita ventana gráfica  173
                                                (*r).ano2 = ano2;         // Asigna ano2 al nodo r
                                                i = 164;
                                                pon(tope,i,NOT_PUT);    // Pone ventana gráfica del cursor
                                                break;
                                          }
                                      }
                                    quita(tope);      // Quita ventana gráfica del cursor
                                    quita(tope);      // Quita ventana gráfica  179
                                    pon(tope,179,COPY_PUT);      // Pone ventana gráfica 179
                                    texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                    bioskey(0);
                                    quita(tope);      // Quita ventana gráfica  179
                                     k = 180;
                                      setfillstyle(SOLID_FILL,GREEN);
                                      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                    for (k=182; k<=185; k++)
                                     {
                                          setfillstyle(SOLID_FILL,GREEN);
                                          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                                     }
                                    i = 219;
                                    pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                                    break;
                          }
     }
   }
  quita(tope);      // Quita ventana gráfica del cursor
   for (k=210; k <= 219; k++)
     {
          setfillstyle(SOLID_FILL,GREEN);
          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
   for (k=152; k <= 154; k++)
     {
          setfillstyle(SOLID_FILL,GREEN);
          bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  quita(tope);      // Quita ventana gráfica  150
}


//  Llama a las funciones que realizan la operación de modificación  de los datos de cada nodo de la lista enlazada EXAMEN.
void cambio_lista_examen(struct EXAMEN *e, int dato)
{
 int i;

 texto("Presione <ENTER> para seleccionar o <ESC> para salir ...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
```

```
 texto("Seleccione el examen que desea cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
 pon(tope,249,COPY_PUT);        // Pone ventana gráfica 249
 pon(tope,250,COPY_PUT);        // Pone ventana gráfica 250
 texto("EXTRAORDINARIO",249,LIGHTBLUE,SMALL_FONT,2,1,2,1);
 texto("FINAL",250,LIGHTBLUE,SMALL_FONT,2,1,2,1);
 pon(tope,249,NOT_PUT);         // Pone ventana gráfica del cursor
 i = opciones1(249,249,250,32);
 quita(tope);        // Quita ventana gráfica cursor
 quita(tope);        // Quita ventana gráfica  250
 quita(tope);        // Quita ventana gráfica  249
 if (i != -1) then      // Selecciona el tipo de examen que se va ha modificar
  {
    switch(i)
     {
          case 249: pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
                    texto("CAMBIANDO DATOS DE EXAMEN EXTRAORDINARIO",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                    cambio_extraord(e,dato);
                    break;
          case 250: pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
                    texto("CAMBIANDO DATOS DE EXAMEN FINAL",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
                    cambio_final(e,dato);
                    break;
     }
    quita(tope);        // Quita ventana gráfica  100
  }
}

//      Busca numero en la lista enlazada EXAMEN
void busca_examen_final(struct MATERIA *p, int numero, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct EXAMEN *e;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   e = (*r).examen;
    while (e != NULL) do
     {
      if  ((*e).tipoexa == 2) then
            {
             if ((*e).numgrupo == numero) then
              {
               arreglo[j].grupo = (*e).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*e).numgrupo;
               j = j + 1;
              }
            }
           e = (*e).sig;
     }
   r = (*r).materias;
  }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
 arreglo[j].numero = -2;
}

//      Busca clase en la lista enlazada EXAMEN
void busca_examen_todo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct EXAMEN *e;
```

```
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  e = (*r).examen;
    while (e != NULL) do
     {
     if  ((*e).tipoexa == clase) then
           {
              arreglo[j].grupo = (*e).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*e).numgrupo;
              j = j + 1;
           }
      e = (*e).sig;
     }
  r = (*r).materias;
 }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
 arreglo[j].numero = -2;
}

//      Busca clase en la lista enlazada EXAMEN
void busca_examen_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct EXAMEN *e;
 int j;

 r = p;
 j = 0;
 e = (*r).examen;
    while (e != NULL) do
     {
     if  ((*e).tipoexa == clase) then
           {
              arreglo[j].grupo = (*e).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*e).numgrupo;
              j = j + 1;
           }
          e = (*e).sig;
     }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

// Cuenta elementos del arreglo ARREGLO
void busca_examen_final_cuenta(struct MATERIA *p, int numero, int *cuenta)
{
 struct MATERIA *r;
 struct EXAMEN *e;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  e = (*r).examen;
    while (e != NULL) do
     {
```

```
      if ((*e).tipoexa == 2) then
            {
             if ((*e).numgrupo == numero) then
                j = j + 1;
            }
           e = (*e).sig;
      }
   r = (*r).materias;
  }
  *cuenta = j;
}

//      Busca clase en la lista enlazada EXAMEN
void busca_examen_todo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct EXAMEN *e;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   e = (*r).examen;
    while (e != NULL) do
     {
      if ((*e).tipoexa == clase) then
                j = j + 1;
      e = (*e).sig;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

//      Busca clase en la lista enlazada EXAMEN
void busca_examen_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct EXAMEN *e;
 int j;

 r = p;
 j = 0;
 e = (*r).examen;
    while (e != NULL) do
     {
      if ((*e).tipoexa == clase) then
                j = j + 1;
           e = (*e).sig;
     }
 *cuenta = j;
}

//      Pone en pantalla los datos de la lista enlazada EXAMEN. Se inicia en  el nodo inicio hasta el nodo fin.
int ventanas_examen_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo,int *max, int z, int b)
{
 struct MATERIA *s, *r;
 struct EXAMEN *e;
 int i,c,k,j;
 char dia[20],mes[20],ano[20],numero[20];

 r=p;
 i = 259;
j = *max;
```

```
for (k=inicio; k <= fin; k++)
 {
  s = busca_materia(r,arreglo[k].materia);
  e = (*e).examen;
  while ((*e).num != arreglo[k].grupo) do
              e = (*e).sig;
  if (e != NULL) then
    {
            cuadro(j,CYAN);
            if (z == 251) then
             {
              pon(tope,i,COPY_PUT);       // Pone ventana gráfica para i
              itoa((*e).dia1,dia,10);
              for (c=0; dia[c]; c++);
              if (c == 1) then
                    {
                     strcpy(numero,"0");
                     strcat(numero,dia);
                    }
              else
                   strcpy(numero,dia);
              strcat(numero,"/");
              itoa((*e).mes1,mes,10);
              for (c=0; mes[c]; c++);
              if (c == 1) then
                    strcat(numero,"0");
              strcat(numero,mes);
              strcat(numero,"/");
              itoa((*e).ano1,ano,10);
              strcat(numero,ano);
             }
            if (z == 252) then
             {
              pon(tope,i,COPY_PUT);       // Pone ventana gráfica para i
              itoa((*e).numgrupo,numero,10);
             }
            texto (numero,i,RED,SMALL_FONT,3,2,2,1);
            i=i+1;
            pon(tope,j,COPY_PUT);       // Pone ventana gráfica para i
            if (b == 3) then
              texto ((*e).nombreprofe,i,RED,SMALL_FONT,3,2,2,1);
            else
              texto ((*s).nombre,i,RED,SMALL_FONT,3,2,2,1);
            i=i+1;
            *max=j;
            j=j+1;
           }
    e = (*e).sig;
 }
 return(i);
}

/*    Llama a las funciones que despliegan en pantalla los nodos de  la lista enlazada EXAMEN. Regresa la posición
  del elemento seleccionado para consultar más información. */
int examen_todos(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int b, int tipo, int cuenta)
{
 struct MATERIA *r;
 struct POS
          {
           int inicio;
           int fin;
          };
 struct POS *pos;
 int k, i,regresa,numc,c,ultimo,grupo,quot,rem,j,a,inicio,fin,dato,max;
 char sig[10],cad[10];
```

```
// Reserva espacio en memoria para pos
pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
if (!pos)
  {
    printf("Error.");
    exit(1);
  }
arreglo[0].grupo = 0;
arreglo[0].materia = 0;
arreglo[0].numero = 0;
r = p;
switch(tipo)
 {
        case 251: if (b == 2) then
                    busca_examen_todo(r,arreglo,1);
                  if (b == 3) then
                    busca_examen_materia_tipo(r,arreglo,1);
                  break;
        case 252: if (b == 0) then
                    busca_examen_final(r,numero,arreglo);
                  if (b == 2) then
                    busca_examen_todo(r,arreglo,2);
                  if (b == 3) then
                    busca_examen_materia_tipo(r,arreglo,2);
                  break;
 }

ultimo = 0;
for (k=0; arreglo[k].grupo != -2; k++)
    ultimo = ultimo + 1;
quot = (ultimo/3);
rem = (ultimo%3);
ultimo = ultimo - 1;
if (ultimo == -1) then
  ultimo = 0;
ordena_arreglo(arreglo,ultimo);
if (rem != 0) then
  grupo = quot + 1;
else
  grupo = quot;
 if (grupo == 0) then
  {
    llama_error(50);
    return(-2);
  }
 k = 0;
 for (j=1; j <= grupo; j++)
   {
     for (a=0; arreglo[k].grupo != -2 && a < 3; k++, a++)
          {
           fin = k;
           if (a == 0) then
             inicio = k;
          }
     pos[j].inicio = inicio;
     pos[j].fin = fin;
   }
cuadros_consulta_examen(tipo,b);
j = 1;
max = 122;
inicio = pos[j].inicio;
fin = pos[j].fin;
numc = ventanas_examen_todos(p,inicio,fin,arreglo,&max,tipo,b);   // Despliega los datos de exámen
i = 122;
c=END;
```

```
regresa=0;
pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1: llama_ayuda(55);       // Llama al módulo AYUDA
                 break;
                                        // Desplaza el cursor sobre las ventanas
        case ABAJO:  quita(tope);       // Quita ventana gráfica del cursor
                     i++;
                     if (i == (max + 1)) then
                           i = 122;
                     pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                     break;
        case ARRIBA: quita(tope);               // Quita ventana gráfica del cursor
                     i--;
                     if (i == (122 - 1)) then
                           i = max;
                     pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                     break;
        case TAB:    if ((i >= 122) && (i <= max)) then   // Cambia al menú inferior
                     {
                      regresa=i;
                      i=237;
                      i=mover_ventana_menu(i,i,237,regresa);
                      if (i == 237) then
                            {
                             c = ENTER;
                             dato = -2;
                            }
                     }
                     break;
                                // Cambia a la siguiente página
        case PGDN:   quita(tope);       // Quita ventana gráfica del cursor
                     numc--;
                     for (k=250; k <= numc; k++)
                           quita(tope);       // Quita ventanas gráficas restantes
                     for (k = 122; k <= max; k++)
                           {
                            setfillstyle(SOLID_FILL,GREEN);
                            bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                           }
                     j++;
                     if (j == (grupo+1)) then
                      j=1;
                     inicio=pos[j].inicio;
                     fin=pos[j].fin;
                     max=122;
                     numc = ventanas_examen_todos(p,inicio,fin,arreglo,&max,tipo,b);
                     strcpy(sig,"PAG. ");
                     itoa(j,cad,10);
                     strcat(sig,cad);
                     texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                     if (i > max) then
                      {
                       i = max;
                       pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                      }
                     else
                      pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                     break;
                                // Cambia a la página anterior
       case PGUP:   quita(tope);        // Quita ventana gráfica  cursor
```

```
                    numc--;
                    for (k=259; k <= numc; k++)
                          quita(tope);        // Quita ventanas gráficas restantes
                    for (k = 122; k <= max; k++)
                          {
                           setfillstyle(SOLID_FILL,GREEN);
                           bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                          }
                    j--;
                    if (j == 0) then
                     j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=122;
                    numc = ventanas_examen_todos(p,inicio,fin,arreglo,&max,tipo,b);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                    if (i > max) then
                     {
                      i = max;
                      pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                     }
                    else
                     pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
   }
  }
  quita(tope);        // Quita ventana gráfica  cursor
  numc--;
  for (k=259; k <= numc; k++)
     quita(tope);        // Quita ventanas gráficas restantes
  for (k = 257; k <= 258; k++)
     {
      setfillstyle(SOLID_FILL,GREEN);
      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  for (k = 122; k <= max; k++)
     {
      setfillstyle(SOLID_FILL,GREEN);
      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
  texto("MATERIA SELECCIONADA",240,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
  if (dato != -2) then
    {
     dato = i - 122;
     a = pos[j].inicio;
     a = a + dato;
    }
  else
   a = dato;
  farfree(pos);                                        // Libera el espacio reservado para pos
  return(a);                                           // Regresa la posición del elemento seleccionado
}


//     Despliega en pantalla los datos de exámen extraordinario del nodo apuntado por la posición a.
void consulta_datos_examen_extraord(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10])
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 int k,c;
 char numero[100],cadena[100],dia[50],mes[50],ano[50];
```

```
r = p;
if (y == 3) then
quita(tope);        // Quita ventana gráfica 240
for (k=200; k <= 240; k++)
 {
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 }
while ((*r).num != arreglo[a].materia) do
        r = (*r).materias;
e = (*r).examen;
while ((*e).num != arreglo[a].grupo) do          // Se recorre la lista hasta la posición a
        e = (*e).sig;
 texto("Presione < ESC > para ir al menú . . . . ",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
 // Despliega los datos de materia
 texto((*r).nombre,370,LIGHTRED,SMALL_FONT,1,1,2,1);
 itoa((*r).clave,numero,10);
 strcpy(cadena,"CLAVE ... ");
 strcat(cadena,numero);
 texto(cadena,371,LIGHTRED,SMALL_FONT,1,1,2,1);

 // Despliega los datos del profesor  titular
 strcpy(cadena,"PROF. TI. ");
 strcat(cadena,(*e).nombreprofe);
 texto(cadena,249,LIGHTRED,SMALL_FONT,1,1,2,1);
 strcpy(cadena,"RFC ... ");
 strcat(cadena,(*e).rfc);
 texto(cadena,250,LIGHTRED,SMALL_FONT,1,1,2,1);

 // Despliega los datos del profesor  suplente
 strcpy(cadena,"PROF. SU. ");
 strcat(cadena,(*e).profe);
 texto(cadena,251,LIGHTRED,SMALL_FONT,1,1,2,1);
 strcpy(cadena,"RFC ... ");
 strcat(cadena,(*e).rfc2);
 texto(cadena,252,LIGHTRED,SMALL_FONT,1,1,2,1);

 // Despliega los datos del salón y periodo
 itoa((*e).numsalon,numero,10);
 if ((*e).tiposalon == 2) then
   strcpy(cadena,"L - ");
 else
   strcpy(cadena,"A - ");
 if ((*e).numsalon == 0) then
   {
    strcat(cadena," ");
   }
 else
     strcat(cadena,numero);
 texto(cadena,253,LIGHTRED,SMALL_FONT,1,1,2,1);
 strcpy(cadena,"SEMESTRE ... ");
 strcat(cadena,semestre);
 texto(cadena,254,LIGHTRED,SMALL_FONT,1,1,2,1);

// Despliega los datos de fecha y Hora
   itoa((*e).dia1,dia,10);
   for (c=0; dia[c]; c++);
   if (c == 1) then
     {
          strcpy(numero,"0");
          strcat(numero,dia);
     }
   else
     strcpy(numero,dia);
   strcat(numero,"/");
```

```
    itoa((*e).mes1,mes,10);
    for (c=0; mes[c]; c++);
    if (c == 1) then
          strcat(numero,"0");
    strcat(numero,mes);
    strcat(numero,"/");
    itoa((*e).ano1,ano,10);
    strcat(numero,ano);
    strcpy(cadena,"FECHA ...  ");
    strcat(cadena,numero);

  texto (cadena,255,LIGHTRED,SMALL_FONT,1,1,2,1);
  strcpy(cadena,"HORA ...  ");
  strcat(cadena,(*e).hora1);
  texto (cadena,256,LIGHTRED,SMALL_FONT,1,1,2,1);
  while (c != ESC) do
     c = biosKey(0);
  for (k = 370; k<= 371; k++)
   {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
  if (y == 3) then
   {
    k = 246;
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
  for (k=249; k <= 256; k++)
   {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
  ventana_menu_consulta();
  if (y == 3) then
   {
    pon(tope,240,COPY_PUT);       // Pone ventana gráfica 240
    texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
   }
}

//    Despliega en pantalla los datos de exámen final del nodo apuntado por la posición a.
void consulta_datos_examen_final(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10])
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 int k,c;
 char numero[100],cadena[100],dia[50],mes[50],ano[50];

 r = p;
 if (y == 3) then
 quita(tope);       // Quita ventana gráfica  240
 for (k=230; k <= 240; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 while ((*r).num != arreglo[a].materia) do                    // Recorre la lista hasta la posición a
          r = (*r).materias;

 e = (*r).examen;
 while ((*e).num != arreglo[a].grupo) do
          e = (*e).sig;
 texto("Presione < ESC > para ir al menú . . . . ",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
```

```
// Despliega los datos de materia
texto((*r).nombre,370,LIGHTRED,SMALL_FONT,1,1,2,1);
itoa((*r).clave,numero,10);
strcpy(cadena,"CLAVE ... ");
strcat(cadena,numero);
texto(cadena,371,LIGHTRED,SMALL_FONT,1,1,2,1);

// Despliega los datos del profesor
strcpy(cadena,"PROF. ");
strcat(cadena,(*e).nombreprofe);
texto(cadena,249,LIGHTRED,SMALL_FONT,1,1,2,1);
strcpy(cadena,"RFC ... ");
strcat(cadena,(*e).rfc);
texto(cadena,250,LIGHTRED,SMALL_FONT,1,1,2,1);

// Despliega los datos de salón
itoa((*e).numsalon,numero,10);
if ((*e).tiposalon == 2) then
  strcpy(cadena,"L - ");
else
  strcpy(cadena,"A - ");

if ((*e).numsalon == 0) then
  {
   strcat(cadena," ");
  }
else
    strcat(cadena,numero);
texto(cadena,251,LIGHTRED,SMALL_FONT,1,1,2,1);

// Despliega los datos del periodo
strcpy(cadena,"SEMESTRE ... ");
strcat(cadena,semestre);
texto(cadena,252,LIGHTRED,SMALL_FONT,1,1,2,1);

// Fecha y hora primera vuelta
   itoa((*e).dia1,dia,10);
   for (c=0; dia[c]; c++);
   if (c == 1) then
     {
          strcpy(numero,"0");
          strcat(numero,dia);
     }
   else
     strcpy(numero,dia);
   strcat(numero,"/");
   itoa((*e).mes1,mes,10);
   for (c=0; mes[c]; c++);
   if (c == 1) then
      strcat(numero,"0");
   strcat(numero,mes);
   strcat(numero,"/");
   itoa((*e).ano1,ano,10);
   strcat(numero,ano);
   strcpy(cadena,"FECHA 1a. vuelta ... ");
   strcat(cadena,numero);
texto (cadena,253,LIGHTRED,SMALL_FONT,1,1,2,1);
strcpy(cadena,"HORA 1a. vuelta ... ");
strcat(cadena,(*e).hora1);
texto (cadena,254,LIGHTRED,SMALL_FONT,1,1,2,1);

// Fecha y Hora segunda vuelta
   itoa((*e).dia2,dia,10);
   for (c=0; dia[c]; c++);
   if (c == 1) then
```

```
    {
        strcpy(numero,"0");
        strcat(numero,dia);
    }
   else
    strcpy(numero,dia);
   strcat(numero,"/");
   itoa((*e).mes2,mes,10);
   for (c=0; mes[c]; c++);
   if (c == 1) then
        strcat(numero,"0");
   strcat(numero,mes);
   strcat(numero,"/");
   itoa((*e).ano2,ano,10);
   strcat(numero,ano);
   strcpy(cadena,"FECHA 2a. vuelta ... ");
   strcat(cadena,numero);
 texto (cadena,255,LIGHTRED,SMALL_FONT,1,1,2,1);
 strcpy(cadena,"HORA 2a. vuelta ... ");
 strcat(cadena,(*e).hora1);
 texto (cadena,256,LIGHTRED,SMALL_FONT,1,1,2,1);
 while (c != ESC) do
    c = bioskey(0);
 for (k = 370; k<= 371; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 if (y == 3) then
  {
   k = 248;
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }

 for (k=249; k <= 256; k++)
 {
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 }
 ventana_menu_consulta();
 if (y == 3) then
 {
  pon(tope,240,COPY_PUT);       // Pone ventana gráfica 240
  texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
 }
}
```

B.3.8. GRUPOS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\grupos.h"
#include "d:\consulta.h"
#include "d:\errores.h"
#include "d:\ayuda.h"

#define  then


/*   Verifica que cadena no se encuentre en la lista enlazada GRUPO. Si no se encuentra regresa 0, de lo contrario
  regresa 1 y un mensaje de error */
int error_grupo_igual(struct GRUPO *p, char cadena[100])
{
 struct GRUPO *r;
 int valor, numero,k,n;

 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
    {
     valor = 1;
     n = 18;
    }
  }
 if (valor == 0) then
  {
   numero = atoi(cadena);
   r = p;
   while (r != NULL) do
   {
    if ((*r).numgrupo == numero) then
     {
          valor = 1;
          n = 26;
     }
    r = (*r).apun;
   }
  }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}

/*   Verifica que cadena no se encuentre en la lista enlazada GRUPO. Si no se encuentra regresa 0, de lo contrario
  regresa 1 y un mensaje de error */
int error_grupo_cambiar(struct GRUPO *p, char cadena[100], int dato)
```

```
{
 struct GRUPO *r;
 int valor, numero,n,k;

 valor = 0;
 for (k=0; cadena[k]; k++)
  {
   if (isdigit(cadena[k]) == 0) then
    {
     valor = 1;
     n = 18;
    }
  }
 numero = atoi(cadena);
 r = p;
 while ((*p).num != dato) do
    p = (*p).apun;
 (*p).numgrupo = 0;
 while (r != NULL) do
 {
  if ((*r).numgrupo == numero) then
    {
     valor = 1;
     n = 28;
    }
  r = (*r).apun;
 }
 if (valor == 1) then
   llama_error(n);
 return(valor);
}


//      Lee los datos de la lista enlazada GRUPO
void lee_nodo_grupos(struct GRUPO *g, struct GRUPO *s)
{
 char cadena[100], nombreprofe[100], rfc[100], hora1[100], hora2[100],telofc[100], telcasa[100], antunam[100], antenep[100];
 int numgrupo, clasegrupo, tipo, numalumnos, i, c, regresa, j, m, k, dias[10], tipoprofe;
 long int numsalon;
                                                        // Se inicializan las variables
 strcpy(cadena," ");
 strcpy(nombreprofe," ");
 strcpy(rfc," ");
 strcpy(hora1," ");
 strcpy(hora2," ");
 strcpy(telofc,"-");
 strcpy(telcasa,"-");
 strcpy(antunam,"-");
 strcpy(antenep,"-");
 numalumnos = 0;
 numsalon = 0;
 dias[0] = 10;
 tipo = 0;
 tipoprofe = 0;
 pon(tope,150,COPY_PUT);        // Pone ventana gráfica 150
 cuadro(150,BLUE);              // Pone menú inferior
 texto("SALIR",152,LIGHTRED,SMALL_FONT,2,1,2,1);
 texto("<F1> AYUDA",153,LIGHTRED,SMALL_FONT,2,1,2,1);
 texto("<TAB> CAMBIA MENU",154,LIGHTRED,SMALL_FONT,3,2,2,1);

// Lee el número de grupo
 pon(tope,160,COPY_PUT);        // Pone ventana gráfica
 pon(tope,161,COPY_PUT);        // Pone ventana gráfica
 texto("Número de grupo . . .",160,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,1);
 k = 1;
```

```
while (k == 1) do              // Verifica que el dato sea correcto
 {
  lee_cadena(cadena,4,161,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,1);
  k = error_grupo_igual(e,cadena);
 }
texto(cadena,161,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,1);
numgrupo=atoi(cadena);

// Selecciona el tipo de grupo
pon(tope,162,COPY_PUT);        // Pone ventana gráfica 162
pon(tope,163,COPY_PUT);        // Pone ventana gráfica 163
texto("Grupo Teoria",162,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,1);
texto("Grupo Lab.",163,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,1);
pon(tope,162,NOT_PUT);         // Pone ventana gráfica del cursor
i = -1;
while (i == -1) do
i = opciones1(162,162,163,16);
quita(tope);        // Quita ventana gráfica del cursor
quita(tope);        // Quita ventana gráfica 163
quita(tope);        // Quita ventana gráfica 162
pon(tope,164,COPY_PUT);        // Pone ventana gráfica 164
   switch(i)
    {
          case 162:  texto("Grupo de Teoria",164,LIGHTBLUE,SANS_SERIF_FONT,2,3,1,1);
                     clasegrupo = 1;
                     k = 0;
                     break;
          case 163:  texto("Grupo de Laboratorio",164,LIGHTBLUE,SANS_SERIF_FONT,2,3,1,1);
                     clasegrupo = 2;
                     k = 0;
                     break;

    }
texto("Seleccione los datos de grupo que desee introducir . . .",165,BLUE,SMALL_FONT,3,2,2,1);
texto("PROFESOR",166,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("SALON",167,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("HORARIO",168,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("DIAS",169,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("ALUMNOS",170,LIGHTBLUE,SMALL_FONT,2,1,2,1);

i=166;
j=1;
pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
while (j == 1)
 {
  c = biosKey(0);
  switch(c)
  {
          case F1:  llama_ayuda(17);                    // Llama al módulo AYUDA
                    break;
                                                        // Desplaza el cursor sobre el menú de opciones
          case DERECHA:  quita(tope);       // Quita ventana gráfica del cursor
                         i++;
                         if (i == 171) then
                               i = 166;
                         pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                         break;

          case IZQUIERDA: quita(tope);      // Quita ventana gráfica del cursor
                          i--;
                          if (i == 165) then
                                i = 170;
                          pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                          break;
```

```
case TAB:    if ((i >= 166) && (i <= 170)) then        // Cambia al menú inferior
                {
                regresa = i;
                i = 152;
                i = mover_ventana_menu(i,i,152,regresa);
                if (i == 152) then
                     j = 0;
                }
             break;

case ENTER:  quita(tope);      // Quita ventana gráfica del cursor
             switch(i)
              {
                    case 166: // Lee el nombre del profesor
                              pon(tope,223,COPY_PUT);       // Pone ventana gráfica 223
                              pon(tope,224,COPY_PUT);       // Pone ventana gráfica 224
                              texto("Nombre del Profesor . . .",223,RED,SMALL_FONT,1,1,2,1);
                              lee_cadena(nombreprofe,32,224,RED,SMALL_FONT,1,1,2,1);
                              texto(nombreprofe,224,RED,SMALL_FONT,1,1,2,1);

                              // Lee el nombramiento del profesor
                              pon(tope,225,COPY_PUT);       // Pone ventana gráfica 225
                              pon(tope,226,COPY_PUT);       // Pone ventana gráfica 226
                              texto("TITULAR",225,RED,SMALL_FONT,1,1,2,1);
                              texto("AYUTE",226,RED,SMALL_FONT,1,1,2,1);
                              pon(tope,225,NOT_PUT);        // Pone ventana gráfica del cursor
                              i = opciones1(225,225,226,16);    // Selecciona opción
                              quita(tope);      // Quita ventana gráfica del cursor
                              quita(tope);      // Quita ventana gráfica 226
                              quita(tope);      // Quita ventana gráfica 225
                              pon(tope,227,COPY_PUT);       // Pone ventana gráfica 227
                              if (i == 225) then
                                   {
                                    texto("PROF. TITULAR",227,RED,SMALL_FONT,1,1,2,1);
                                    tipoprofe = 1;
                                   }
                              if (i == 226) then
                                   {
                                    texto("AYUDANTE",227,RED,SMALL_FONT,1,1,2,1);
                                    tipoprofe = 2;
                                   }
                              if (i == -1) then
                                   {
                                    texto("SIN NOMBRAMIENTO",227,RED,SMALL_FONT,1,1,2,1);
                                    tipoprofe = 3;
                                   }

                              // Lee el R.F.C. del profesor y teléfonos
                              i=188;
                              while (i != -1) do
                                   {
                                    pon(tope,166,COPY_PUT);       // Pone ventana gráfica 166
                                    texto("Seleccione . . .",166,RED,SMALL_FONT,1,1,2,1);
                                    pon(tope,166,COPY_PUT);       // Pone ventana gráfica 166
                                    pon(tope,189,COPY_PUT);       // Pone ventana gráfica 189
                                    pon(tope,190,COPY_PUT);       // Pone ventana gráfica 190
                                    pon(tope,191,COPY_PUT);       // Pone ventana gráfica 191
                                    pon(tope,192,COPY_PUT);       // Pone ventana gráfica 192
                                    texto("RFC.",188,RED,SMALL_FONT,1,1,2,1);
                                    texto("TEL. OFC.",189,RED,SMALL_FONT,1,1,2,1);
                                    texto("TEL. CASA",190,RED,SMALL_FONT,1,1,2,1);
                                    texto("ANT. UNAM",191,RED,SMALL_FONT,1,1,2,1);
                                    texto("ANT. ENEP",192,RED,SMALL_FONT,1,1,2,1);
                                    pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                                    texto("Presione <ESC> para salir
```

```
....",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);

                                        pon(tope,l,NOT_PUT);        // Pone ventana gráfica del cursor
                                        i = opciones1(l,188,192,19);
                                        quita(tope);        // Quita ventana gráfica del cursor
                                        quita(tope);        // Quita ventana gráfica 179
                                        quita(tope);        // Quita ventana gráfica 192
                                        quita(tope);        // Quita ventana gráfica 191
                                        quita(tope);        // Quita ventana gráfica 190
                                        quita(tope);        // Quita ventana gráfica 189
                                        quita(tope);        // Quita ventana gráfica 188
                                        quita(tope);        // Quita ventana gráfica 186
                                        if (i != -1) then
                                         {
                                          switch(i)
                                                {
                                                  case 188: // Lee el R.F.C.
                                                            // Pone ventana gráfica 173
                                                            pon(tope,173,COPY_PUT);
                                                            // Pone ventana gráfica 174
                                                            pon(tope,174,COPY_PUT);
                                                            texto("Registro federal de causante . .

.",173,RED,SMALL_FONT,1,1,2,1);

SMALL_FONT,1,1,2,1);

                                                                        lee_cadena(rfc,14,174,RED,

                                                            texto(rfc,174,RED,SMALL_FONT,1,1,2,1);
                                                            // Pone ventana gráfica 179
                                                            pon(tope,179,COPY_PUT);
                                                            texto("Presione una tecla para continuar

...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                                            bioskey(0);
                                                            quita(tope);        // Quita ventana gráfica 179
                                                            quita(tope);        // Quita ventana gráfica 174
                                                            quita(tope);        // Quita ventana gráfica 173
                                                            break;

                                                  case 189: // Teléfono de oficina
                                                            // Pone ventana gráfica 173
                                                            pon(tope,173,COPY_PUT);
                                                            // Pone ventana gráfica 174
                                                            pon(tope,174,COPY_PUT);
                                                            texto("Teléfono de la oficina . .

.",173,RED,SMALL_FONT,1,1,2,1);

                                                            lee_cadena(telofc,14,174,RED,

SMALL_FONT,1,1,2,1);

                                                            texto(telofc,174,RED,SMALL_FONT,1,1,2,1);
                                                            // Pone ventana gráfica 179
                                                            pon(tope,179,COPY_PUT);
                                                            texto("Presione una tecla para continuar

...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                                            bioskey(0);
                                                            quita(tope);        // Quita ventana gráfica 179
                                                            quita(tope);        // Quita ventana gráfica 174
                                                            quita(tope);        // Quita ventana gráfica 173
                                                            break;

                                                  case 190: // Teléfono de la casa
                                                            // Pone ventana gráfica 173
                                                            pon(tope,173,COPY_PUT);
                                                            // Pone ventana gráfica 174
                                                            pon(tope,174,COPY_PUT);
                                                            texto("Teléfono de la casa . .

.",173,RED,SMALL_FONT,1,1,2,1);

                                                            lee_cadena(telcasa,14,174,RED,

SMALL_FONT,1,1,2,1);

                                                            texto(telcasa,174,RED,SMALL_FONT,1,1,2,1);
```

```
                                                    // Pone ventana gráfica 179
                                                    pon(tope,179,COPY_PUT);
                                                    texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                    bioskey(0);
                                                    quita(tope);      // Quita ventana gráfica 179
                                                    quita(tope);      // Quita ventana gráfica 174
                                                    quita(tope);      // Quita ventana gráfica 173
                                                    break;

                                                case 191: // Antigüedad en la UNAM
                                                    // Pone ventana gráfica 173
                                                    pon(tope,173,COPY_PUT);
                                                    // Pone ventana gráfica 174
                                                    pon(tope,174,COPY_PUT);
                                                    texto("Antigüedad en la UNAM . .
.",173,RED,SMALL_FONT,1,1,2,1);
                                                    lee_cadena(antunam,5,174,RED,
SMALL_FONT,1,1,2,1);
                                                    texto(antunam,174,RED,SMALL_FONT,1,1,2,1);
                                                    // Pone ventana gráfica 179
                                                    pon(tope,179,COPY_PUT);
                                                    texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                    bioskey(0);
                                                    quita(tope);    // Quita ventana gráfica 179
                                                    quita(tope);      // Quita ventana gráfica 174
                                                    quita(tope);      // Quita ventana gráfica 173
                                                    break;

                                                case 192: // Antigüedad en la ENEP
                                                    // Pone ventana gráfica
                                                    pon(tope,173,COPY_PUT);
                                                    // Pone ventana gráfica
                                                    pon(tope,174,COPY_PUT);
                                                    texto("Antigüedad en la ENEP . .
.",173,RED,SMALL_FONT,1,1,2,1);
                                                    lee_cadena(antenep,5,174,RED,
SMALL_FONT,1,1,2,1);
                                                    texto(antenep,174,RED,
SMALL_FONT,1,1,2,1);
                                                    // Pone ventana gráfica 179
                                                    pon(tope,179,COPY_PUT);
                                                    texto("Presione una tecla para continuar
...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                                    bioskey(0);
                                                    quita(tope);      // Quita ventana gráfica 179
                                                    quita(tope);      // Quita ventana gráfica 174
                                                    quita(tope);      // Quita ventana gráfica 173
                                                    break;
                                                }
                                            }
                                        }
                                        quita(tope);      // Quita ventana gráfica 227
                                        quita(tope);      // Quita ventana gráfica 224
                                        quita(tope);      // Quita ventana gráfica 223
                                    i = 166;
                                    pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                                    break;


                            case 167: // Salón
                                    pon(tope,171,COPY_PUT);      // Pone ventana gráfica 171
                                    pon(tope,172,COPY_PUT);      // Pone ventana gráfica 172
                                    texto("Número de salón . . .",171,RED,SMALL_FONT,1,1,2,1);
```

```
k = i;
while (k == 1) do    // Verifica que el dato sea correcto
     {
      lee_cadena(cadena,5,172,RED,SMALL_FONT,1,1,2,1);
      k = error_numero_entero(cadena);
     }
texto(cadena,172,RED,SMALL_FONT,1,1,2,1);
numsalon = atoi(cadena);

// Tipo
pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
texto("Tipo de salón",173,RED,SMALL_FONT,1,1,2,1);
pon(tope,175,COPY_PUT);       // Pone ventana gráfica 175
pon(tope,176,COPY_PUT);       // Pone ventana gráfica 176
pon(tope,177,COPY_PUT);       // Pone ventana gráfica 177
pon(tope,178,COPY_PUT);       // Pone ventana gráfica 178
texto("D",175,RED,SMALL_FONT,1,1,2,1);
texto("L",176,RED,SMALL_FONT,1,1,2,1);
texto("C",177,RED,SMALL_FONT,1,1,2,1);
texto("G",178,RED,SMALL_FONT,1,1,2,1);
pon(tope,175,NOT_PUT);        // Pone ventana gráfica del cursor
i = opciones1(175,175,178,24);
quita(tope);       // Quita ventana gráfica
quita(tope);       // Quita ventana gráfica 178
quita(tope);       // Quita ventana gráfica 177
quita(tope);       // Quita ventana gráfica 176
quita(tope);       // Quita ventana gráfica 175
pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
if (i == 175) then
      {
       texto("Salón de dibujo",174,RED,SMALL_FONT,1,1,2,1);
       tipo = 1;
      };
if (i == 176) then
      {
       texto("Salón de laboratorio",174,RED,SMALL_FONT,1,1,2,1);
       tipo = 2;
      };
     if (i == 177) then
      {
       texto("Salón chico (máx. 30
alumnos)",174,RED,SMALL_FONT,1,1,2,1);
       tipo = 3;
      };
     if (i == 178) then
      {
       texto("Salón grande (máx. 60
alumnos)",174,RED,SMALL_FONT,1,1,2,1);
       tipo = 4;
      };
     if (i == -1) then
      {
       texto("Salón sin clasificar",174,RED,SMALL_FONT,1,1,2,1);
       tipo = 5;
      };
pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

bioskey(0);
quita(tope);       // Quita ventana gráfica 179
quita(tope);       // Quita ventana gráfica 174
quita(tope);       // Quita ventana gráfica 173
quita(tope);       // Quita ventana gráfica 172
quita(tope);       // Quita ventana gráfica 171
i = 187;
```

```
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;

                case 166: // Horarios
                    pon(tope,171,COPY_PUT);     // Pone ventana gráfica 171
                    pon(tope,172,COPY_PUT);     // Pone ventana gráfica 172
                    texto("Hora de inicio de clase . . .",171,RED,SMALL_FONT,1,1,2,1);
                    k = 1;
                    while (k == 1) do                   // Verifica que el dato sea correcto
                        {
                        lee_cadena(hora1,5,172,RED,SMALL_FONT,1,1,2,1);
                        k = hora_correcta(hora1,hora1,1);
                        }
                    texto(hora1,172,RED,SMALL_FONT,1,1,2,1);
                    pon(tope,173,COPY_PUT);     // Pone ventana gráfica 173
                    pon(tope,174,COPY_PUT);     // Pone ventana gráfica 174
                    texto("Hora de terminación de clase . . .",173,RED,SMALL_FONT,1,1,2,1);
                    k = 1;
                    while (k == 1) do       // Verifica que el dato sea correcto
                        {
                        lee_cadena(hora2,5,174,RED,SMALL_FONT,1,1,2,1);
                        k = hora_correcta(hora1,hora2,2);
                        }
                    texto(hora2,174,RED,SMALL_FONT,1,1,2,1);
                    pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                    texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                    bioskey(0);
                    quita(tope);    // Quita ventana gráfica 179
                    quita(tope);    // Quita ventana gráfica 174
                    quita(tope);    // Quita ventana gráfica 173
                    quita(tope);    // Quita ventana gráfica 172
                    quita(tope);    // Quita ventana gráfica 171
                    i = 166;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;

                case 169: // Días
                    texto("LUNES",180,RED,SMALL_FONT,1,1,2,1);
                    texto("MARTES",181,RED,SMALL_FONT,1,1,2,1);
                    texto("MIERCOLES",182,RED,SMALL_FONT,1,1,2,1);
                    texto("JUEVES",183,RED,SMALL_FONT,1,1,2,1);
                    texto("VIERNES",184,RED,SMALL_FONT,1,1,2,1);
                    texto("SABADO",185,RED,SMALL_FONT,1,1,2,1);
                    texto("Dias seleccionados..",186,RED,SMALL_FONT,1,1,2,1);
                    texto("Presione <ESC> cuando termine de
seleccionar",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    pon(tope,180,NOT_PUT);      // Pone ventana gráfica del cursor
                    m = 187;
                    i = 180;
                    k = 0;
                    while ((i != -1) && (k < 6)) do
                        {
                        i = opciones1(i,180,185,21);
                        switch(i)
                          {
                            case 180: quita(tope);      // Quita ventana gráfica del cursor
                                      texto("Lunes",m,RED,SMALL_FONT,1,1,2,1);
                                      dias[k]=1;
                                      m = m + 1;
                                      k = k + 1;
                                      // Pone ventana gráfica del cursor
                                      pon(tope,i,NOT_PUT);
                                      break;
                            case 181: quita(tope);      // Quita ventana gráfica del cursor
```

```
                                        texto("Martes",m,RED,SMALL_FONT,1,1,2,1);
                                        dias[k]=2;
                                        m = m + 1;
                                        k = k + 1;
                                        // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;
                                case 182: quita(tope);      // Quita ventana gráfica del cursor
                                        texto("Miercoles",m,RED,SMALL_FONT,1,1,2,1);
                                        dias[k]=3;
                                        m = m + 1;
                                        k = k + 1;
                                        // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;
                                case 183: quita(tope);      // Quita ventana gráfica del cursor
                                        texto("Jueves",m,RED,SMALL_FONT,1,1,2,1);
                                        dias[k]=4;
                                        m = m + 1;
                                        k = k + 1;
                                        // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;
                                case 184: quita(tope);      // Quita ventana gráfica del cursor
                                        texto("Viernes",m,RED,SMALL_FONT,1,1,2,1);
                                        dias[k]=5;
                                        m = m + 1;
                                        k = k + 1;
                                        // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;
                                case 185: quita(tope);      // Quita ventana gráfica del cursor
                                        texto("Sábado",m,RED,SMALL_FONT,1,1,2,1);
                                        dias[k]=6;
                                        m = m + 1;
                                        k = k + 1;
                                        // Pone ventana gráfica del cursor
                                        pon(tope,i,NOT_PUT);
                                        break;
                                };
                        };
                pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                dias[k] = 10;
                // Quita ventanas restantes
                quita(tope);        // Quita ventana gráfica del cursor
                for (k = 179; k <= m-1; k++)
                    {
                     setfillstyle(SOLID_FILL,GREEN);
                     bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                    }
                i = 189;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 170: // Número de alumnos
                pon(tope,171,COPY_PUT);     // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);     // Pone ventana gráfica 172
                texto("Total de alumnos inscritos . . .",171,RED,SMALL_FONT,1,1,2,1);
                k = 1;
                while (k == 1) then                 // Verifica que el dato sea correcto
                    {
```

```
                                        lee_cadena(cadena,3,172,RED,SMALL_FONT,1,1,2,1);
                                        k = error_numero_entero(cadena);
                                      }
                                    texto(cadena,172,RED,SMALL_FONT,1,1,2,1);
                                    numalumnos = atoi(cadena);
                                    pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                                    texto("Presione una tecla para
continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);

                                    bioskey(0);
                                    quita(tope);        // Quita ventana gráfica 179
                                    quita(tope);        // Quita ventana gráfica 172
                                    quita(tope);        // Quita ventana gráfica 171
                                    i = 170;
                                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                                    break;

                                  }
    }
   }
// Quitar ventanas
quita(tope);        // Quita ventana gráfica del cursor
for (k = 165; k <= 170; k++)
{
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
}
quita(tope);        // Quita ventana gráfica 164
quita(tope);        // Quita ventana gráfica 161
quita(tope);        // Quita ventana gráfica 160
quita(tope);        // Quita ventana gráfica 150
// Asignar campos al nodo *g
(*g).numgrupo = numgrupo;
(*g).clasegrupo = clasegrupo;
for (k=0; nombreprofe[k]; k++)
   nombreprofe[k] = toupper(nombreprofe[k]);
strcpy((*g).nombreprofe,nombreprofe);
(*g).tipoprofe = tipoprofe;
for (k=0; rfc[k]; k++)
   rfc[k] = toupper(rfc[k]);
strcpy((*g).rfc,rfc);
for (k=0; telofc[k]; k++)
   telofc[k] = toupper(telofc[k]);
strcpy((*g).telofc,telofc);
for (k=0; telcasa[k]; k++)
   telcasa[k] = toupper(telcasa[k]);
strcpy((*g).telcasa,telcasa);
for (k=0; antunam[k]; k++)
   antunam[k] = toupper(antunam[k]);
strcpy((*g).antunam,antunam);
for (k=0; antanep[k]; k++)
   antanep[k] = toupper(antanep[k]);
strcpy((*g).antanep,antanep);
(*g).numsalon = numsalon;
(*g).tipo = tipo;
strcpy((*g).hora1,hora1);
strcpy((*g).hora2,hora2);
for (k=0; dias[k] <= 10; k++)
(*g).dias[k] = dias[k];
(*g).numalumnos = numalumnos;
}


/*    Permite mantener un ciclo dentro del cuál se pueden agregar los nodos que se deseen en la lista enlazada
  GRUPO.*/
int salir_lee_grupo(void)
{
```

```
int c,j;

pon(tope,253,COPY_PUT);        // Pone ventana gráfica 253
texto("AGREGAR UNO MAS",253,RED,SANS_SERIF_FONT,1,2,1,2);
pon(tope,254,COPY_PUT);        // Pone ventana gráfica 254
texto("SALIR",254,RED,SANS_SERIF_FONT,1,2,1,2);
c=0;
j=-1;
pon(tope,253,NOT_PUT);         // Pone ventana gráfica del cursor
while (c == 0)
{
  while (j == -1) do
   j=opciones1(253,253,254,22);
  switch (j)
    {
        case  253 : c = -1;
                    break;
        case  254 : c = -3;
                    break;
    }
}
quita(tope);        // Quita ventana gráfica del cursor
quita(tope);        // Quita ventana gráfica 253
quita(tope);        // Quita ventana gráfica 254
return(c);
}


//     Construye el menú inferior
void ventana_nodo_lista_grupo(void)
{
texto("GRUPO",195,LIGHTBLUE,SMALL_FONT,2,1,2,1);
texto("CLASE DE GRUPO",198,LIGHTBLUE,SMALL_FONT,2,1,2,1);
pon(tope,150,COPY_PUT);        // Pone ventana gráfica
cuadro(150,BLUE);
texto("SALIR",152,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("<F1> AYUDA",153,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("PgUp/PgDn",154,LIGHTRED,SMALL_FONT,2,1,2,1);
}


/*    Despliega en pantalla los datos de los nodos de la lista enlazada GRUPO. Se inicia en la posición inicio hasta la
  posición fin. Regresa la última ventana. */
int ventanas_ponen_grupos(struct GRUPO *g, int inicio, int fin, int *max)
{
 struct GRUPO *r;
 int i,c,k,l;
 char numero[20];

 r=g;
 i = 197;
 while ((*r).num != inicio) do
        r=(*r).apun;
l = *max;
for (k=inicio; k <= fin; k++)
 {
        cuadro(i,CYAN);
        pon(tope,i,COPY_PUT);        // Pone ventana gráfica para i
        itoa((*r).numgrupo,numero,10);
        texto (numero,i,RED,SMALL_FONT,2,1,2,1);
        i=i+1;
        pon(tope,i,COPY_PUT);        // Pone ventana gráfica para i
        c = (*r).clasegrupo;
        switch(c)
         {
```

```
                    case 1: texto ("Grupo de teoria",l,RED,SMALL_FONT,2,1,2,1);
                            break;
                    case 2: texto ("Grupo de laboratorio",l,RED,SMALL_FONT,2,1,2,1);
                            break;
            }
      i=i+1;
      *max=j;
      j=j+1;
      r = (*r).apun;
 }
 return(l);
}


/*    Llama a las funciones que despliegan los nodos de la lista enlazada GRUPO.  Regresa la posición dentro de la
  lista del nodo seleccionado. */
int mover_pantallas_grupos(int l, int min, int salir, struct GRUPO *g, int cuenta)
{
  struct GRUPO *r,*s;
  struct POS{
              int inicio;
              int fin;
             };
  struct POS *pos;
  int c,regresa,inicio,fin,grupo,ultimo,j,quot,rem,numc,k,dato,max;
        // Asigna espacio en memoria para pos
  pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
  if (!pos)
   {
    printf("Error.");
    exit(1);
   }
  r=g;
  s=g;
  dato=0;
  while (r != NULL) do
  {
          ultimo=(*r).num;
          r=(*r).apun;
  }
  quot=(ultimo/4);
  rem=(ultimo%4);
  if (rem != 0) then
    grupo=quot + 1;
  else
    grupo=quot;
  fin=0;
  for (j=1; j <= grupo; j++)
  {
    inicio=fin+1;
    while (( s != NULL) && (fin < (inicio + 3))) do
    {
      fin=(*s).num;
      s=(*s).apun;
    }
    pos[j].inicio=inicio;
    pos[j].fin=fin;
  }
  ventana_nodo_lista_grupo();                          // Construye las ventanas del menú
  max=121;
  j=1;
  inicio=pos[j].inicio;
  fin=pos[j].fin;
  numc=ventanas_ponen_grupos(g,inicio,fin,&max);       // Despliega las ventanas de GRUPO
```

```
c=END;
regresa=0;
pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1: llama_ayuda(25);      // Llama al módulo AYUDA
                 break;
                                       // Desplaza al cursor sobre las ventanas
        case ABAJO:  quita(tope);      // Quita ventana gráfica del cursor
                     i++;
                     if (i == (max + 1)) then
                           i = min;
                     pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                     break;

        case ARRIBA: quita(tope);                // Quita ventana gráfica del cursor
                     i--;
                     if (i == (min - 1)) then
                           i = max;
                     pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                     break;

        case TAB:    if ((i >= min) && (i <= max)) then          // Cambia al menú inferior
                     {
                      regresa=i;
                      i=salir;
                      i=mover_ventana_menu(i,i,salir,regresa);
                      if (i == 152) then
                            {
                             c = ENTER;
                             dato = -2;
                            }
                     }
                     break;

                                       // Cambia a la siguiente página
        case PGDN:   quita(tope);      // Quita ventana gráfica del cursor
                     numc--;
                     for (k=197; k <= numc; k++)
                           quita(tope);       // Quita ventanas gráficas
                     for (k=121; k <= max; k++)
                           {
                            setfillstyle(SOLID_FILL,GREEN);
                            bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                           }
                     j++;
                     if (j == (grupo+1)) then
                      j=1;
                     inicio=pos[j].inicio;
                     fin=pos[j].fin;
                     max=121;
                     numc=ventanas_ponen_grupos(g,inicio,fin,&max);
                     if (i > max) then
                     {
                           i = max;
                           pon(tope,i,NOT_PUT);         // Pone ventana gráfica del cursor
                     }
                     else
                     pon(tope,i,NOT_PUT);               // Pone ventana gráfica del cursor
                     break;

       case PGUP:   quita(tope);                        // Quita ventana gráfica del cursor
```

```
                    numc--;
                    for(k=197; k <= numc; k++)
                        quita(tope);        // Quita ventanas gráficas restantes
                    for (k=121; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                    i--;
                    if (i == 0) then
                     i=grupo;
                    inicio=pos[i].inicio;
                    fin=pos[i].fin;
                    max=121;
                    numc=ventanas_ponen_grupos(g,inicio,fin,&max);
                    if (i > max) then
                     {
                        i = max;
                        pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                     }
                    else
                     pon(tope,i,NOT_PUT);           // Pone ventana gráfica del cursor
                    break;
   }
  }
  if (dato != -2) then
   {
    dato = i - 120;
    dato=dato + 4*(i-1);
   }
  quita(tope);      // Quita ventana gráfica del cursor
  numc--;
  for (k=197; k <= numc; k++)
    quita(tope);    // Quita ventanas gráficas restantes
  for (k=121; k <= max; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  quita(tope);      // Quita ventana gráfica 150
  for (k=152; k <= 154; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  for (k=195; k <= 196; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  farfree(pos);                                 // Libera el espacio reservado para pos
  return(dato);                                 // Regresa la posición de la opción seleccionada
}


//  Permite hacer modificaciones de los datos contenidos en el nodo de la posición dato, de la lista enlazada GRUPO.
void cambio_lista_grupos(struct GRUPO *g, int dato)
{
 struct GRUPO *r;
 char nombreprofe[100], rfc[100], hora1[100], hora2[100], semestre[100],telofc[100], telcasa[100], antunam[100], antsnep[100];

 int i, j, c, regresa, numgrupo, k, clasegrupo, tipo, numalumnos, m, clas[100],tipoprofe,
 long int numsalon;
 char numero[100];
```

```
r=g;
while ((*r).num != dato)                    // Recorre a la lista hasta dato
        r = (*r).apun;
ventana_nodo_lista_materia();               // Construye menú inferior
texto("Seleccione el modulo que desee cambiar . . .",210,LIGHTRED,SMALL_FONT,2,1,2,1);
texto("GRUPO",211,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("CLASE DE GRUPO",212,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("NOMBRE DEL PROFESOR",213,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("NOMBRAMIENTO",214,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("RFC",215,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("ANTIGUEDAD",216,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("TELEFONOS",217,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("SALON",218,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("TIPO DE SALON",219,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("HORARIO DE CLASE",220,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("DIAS",221,LIGHTBLUE,SMALL_FONT,1,1,2,1);
texto("TOTAL DE ALUMNOS",222,LIGHTBLUE,SMALL_FONT,1,1,2,1);
i = 211;
j = 1;
pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
while(j == 1)
 {
   c = bioskey(0);
   switch(c)
    {
        case F1: llama_ayuda(25);           // Llama al módulo AYUDA
                 break;
                                            // Desplaza el cursor sobre el menú de opciones
        case DERECHA:  quita(tope);         // Quita ventana gráfica
                       i++;
                       if (i == 223) then
                          i = 211;
                       pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                       break;

        case IZQUIERDA: quita(tope);                // Quita ventana gráfica
                        i--;
                        if (i == 210) then
                           i = 222;
                        pon(tope,i,NOT_PUT);      // Pone ventana gráfica del cursor
                        break;

        case TAB:     if ((i >= 211) && (i <= 222) then          // Cambia al menú inferior
                          {
                           regresa=i;
                           i=152;
                           i=mover_ventana_menu(i,j,152,regresa);
                           if (i == 152) then
                                 j = 0;
                          }
                        break;

        case ENTER: quita(tope);      // Quita ventana gráfica del cursor
                    switch(i)
                     {
                          case 211: // Despliega el grupo actual
                                    pon(tope,171,COPY_PUT);      // Pone ventana gráfica 171
                                    pon(tope,172,COPY_PUT);      // Pone ventana gráfica 172
                                    texto("Número de grupo actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                                    itoa((*r).numgrupo,numero,10);
                                    texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                                    // Lee el nuevo grupo
                                    pon(tope,173,COPY_PUT);      // Pone ventana gráfica 173
                                    pon(tope,174,COPY_PUT);      // Pone ventana gráfica 174
                                    texto("Nuevo número de grupo . . .",173,RED,SMALL_FONT,1,1,2,1);
```

```
                k = 1;
                while (k == 1) do                       // Verifica que el dato sea correcto
                 {
                 lee_cadena(numero,4,174,RED,SMALL_FONT,1,1,2,1);
                 k = error_grupo_cambiar(g,numero,dato);
                 }
                texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                numgrupo=atoi(numero);
                pon(tope,179,COPY_PUT);         // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                blockey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 172
                quita(tope);            // Quita ventana gráfica 171
                (*r).numgrupo = numgrupo;       // Asigna numgrupo a r
                i=211;
                pon(tope,i,NOT_PUT);            // Pone ventana gráfica del cursor
                break;

        case 212: // Despliega en pantalla la clase de grupo actual
                pon(tope,171,COPY_PUT);         // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);         // Pone ventana gráfica 172
                texto("Clase de grupo actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                c = (*r).clasegrupo;
                switch(c)
                 {
                        case 1: texto ("Grupo de Teoria",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 2: texto ("Grupo de Laboratorio",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                 }
                 // Lee la nueva clase de grupo
                pon(tope,173,COPY_PUT);         // Pone ventana gráfica 173
                pon(tope,193,COPY_PUT);         // Pone ventana gráfica 193
                pon(tope,194,COPY_PUT);         // Pone ventana gráfica 194
                texto("Nueva clase de grupo . . .",173,RED,SMALL_FONT,1,1,2,1);
                texto("Grupo de Teoria",193,RED,SMALL_FONT,1,1,2,1);
                texto("Grupo de Laboratorio",194,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,NOT_PUT);          // Pone ventana gráfica del cursor
                i = -1;
                while (i == -1) do
                 i = opciones1(193,193,194,16);
                quita(tope);            // Quita ventana gráfica del cursor
                quita(tope);            // Quita ventana gráfica 194
                quita(tope);            // Quita ventana gráfica 193
                pon(tope,174,COPY_PUT);         // Pone ventana gráfica 174
                switch(i)
                 {
                        case 193:  texto("Grupo de Teoria",174,RED,SMALL_FONT,1,1,2,1);
                                   clasegrupo = 1;
                                   break;
                        case 194:  texto("Grupo de Laboratorio",174,RED,SMALL_FONT,1,1,2,1);
                                   clasegrupo = 2;
                                   break;
                 }
                pon(tope,179,COPY_PUT);         // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                blockey(0);
                quita(tope);            // Quita ventana gráfica 179
                quita(tope);            // Quita ventana gráfica 174
                quita(tope);            // Quita ventana gráfica 173
                quita(tope);            // Quita ventana gráfica 171
                quita(tope);            // Quita ventana gráfica 172
```

```
                (*r).clasegrupo = clasegrupo;        // Asigna clasegrupo a r
                i=212;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 213: // Despliega en pantalla el nombre del profesor
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Nombre del profesor actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).nombreprofe,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee del teclado el nuevo nombre del profesor
                pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                texto("Nuevo nombre del profesor . . .",173,BLUE,SMALL_FONT,1,1,2,1);
                lee_cadena(nombreprofe,32,174,RED,SMALL_FONT,1,1,2,1);
                texto(nombreprofe,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);        // Quita ventana gráfica 179
                quita(tope);        // Quita ventana gráfica 174
                quita(tope);        // Quita ventana gráfica 173
                quita(tope);        // Quita ventana gráfica 172
                quita(tope);        // Quita ventana gráfica 171
                for (k=0; nombreprofe[k]; k++)
                nombreprofe[k] = toupper(nombreprofe[k]);
                strcpy((*r).nombreprofe,nombreprofe);   // Asigna nombreprofe a r
                i=213;
                pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                break;

        case 214: // Despliega en pantalla el nombramiento actual
                pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                texto("Nombramiento del profesor actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                c = (*r).tipoprofe;
                switch(c)
                 {
                        case 0: texto("No se asignó profesor",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 1: texto("Titular",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 2: texto("Ayudante",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                        case 3: texto("Sin nombramiento",172,BLUE,SMALL_FONT,1,1,2,1);
                                break;
                 }
                 // Selecciona el nuevo nombramiento
                pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                texto("Nuevo nombramiento . . .",173,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,COPY_PUT);        // Pone ventana gráfica 193
                pon(tope,194,COPY_PUT);        // Pone ventana gráfica 194
                texto("TITULAR",193,RED,SMALL_FONT,1,1,2,1);
                texto("AYUDANTE",194,RED,SMALL_FONT,1,1,2,1);
                pon(tope,193,NOT_PUT);        // Pone ventana gráfica del cursor
                i = opciones1(193,193,194,18);
                quita(tope);        // Quita ventana gráfica del cursor
                quita(tope);        // Quita ventana gráfica 194
                quita(tope);        // Quita ventana gráfica 193
                pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                if (i == 193) then
                        {
                         texto("PROFESOR TITULAR",174,RED,SMALL_FONT,1,1,2,1);
                         tipoprofe = 1;
                        }
```

```
                if (i == 194) then
                        {
                         texto("AYUDANTE DE PROFESOR",174,RED,SMALL_FONT,1,1,2,1);
                         tipoprofe = 2;
                        }
                if (i == -1) then
                        {
                         texto("SIN NOMBRAMIENTO",174,RED,SMALL_FONT,1,1,2,1);
                         tipoprofe = 3;
                        }
                pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);       // Quita ventana gráfica 179
                quita(tope);       // Quita ventana gráfica 174
                quita(tope);       // Quita ventana gráfica 173
                quita(tope);       // Quita ventana gráfica 172
                quita(tope);       // Quita ventana gráfica 171
                (*r).tipoprofe = tipoprofe;              // Asigna tipoprofe a r
                . = 214;
                pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                break;

        case 215: // Despliega el actual R.F.C.
                pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
                pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
                texto("RFC del profesor actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
                texto((*r).rfc,172,BLUE,SMALL_FONT,1,1,2,1);
                // Lee el nuevo R.F.C.
                pon(tope,173,COPY_PUT);       // Pone ventana gráfica
                pon(tope,174,COPY_PUT);       // Pone ventana gráfica
                texto("Registro federal de causante . . .",173,RED,SMALL_FONT,1,1,2,1);
                lee_cadena(rfc,14,174,RED,SMALL_FONT,1,1,2,1);
                texto(rfc,174,RED,SMALL_FONT,1,1,2,1);
                pon(tope,179,COPY_PUT);       // Pone ventana gráfica
                texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                bioskey(0);
                quita(tope);       // Quita ventana gráfica 179
                quita(tope);       // Quita ventana gráfica 174
                quita(tope);       // Quita ventana gráfica 173
                quita(tope);       // Quita ventana gráfica 172
                quita(tope);       // Quita ventana gráfica 171
                for (k=0; rfc[k]; k++)
                rfc[k] = toupper(rfc[k]);
                strcpy((*r).rfc,rfc);
                i=215;
                pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                break;

        case 216: // Desplegar la antigüedad en la UNAM Y ENEP actual
                while (i != -1) do
                {
                 pon(tope,171,COPY_PUT);       // Pone ventana gráfica 171
                 pon(tope,172,COPY_PUT);       // Pone ventana gráfica 172
                 texto("Antiguedad del profesor actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                 strcpy(numero,"Ant. UNAM:  ");
                 strcat(numero,(*r).antunam);
                 strcat(numero,"   Ant. ENEP:  ");
                 strcat(numero,(*r).antenep);
                 texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                 pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
                 texto("Presione <ESC> para salir . . .",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                 pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
                 pon(tope,193,COPY_PUT);       // Pone ventana gráfica 193
                 pon(tope,194,COPY_PUT);       // Pone ventana gráfica 194
```

```
                        texto("Seleccione el dato que desea modificar ...",173,RED,SMALL_FONT,1,1,2,1);
                        texto("ANT. UNAM",193,RED,SMALL_FONT,1,1,2,1);
                        texto("ANT. ENEP",194,RED,SMALL_FONT,1,1,2,1);
                        pon(tope,193,NOT_PUT);        // Pone ventana gráfica del cursor
                            i = opciones1(193,193,194,23);
                        quita(tope);        // Quita ventana gráfica del cursor
                        quita(tope);        // Quita ventana gráfica 194
                        quita(tope);        // Quita ventana gráfica 193
                        quita(tope);        // Quita ventana gráfica 173
                        quita(tope);        // Quita ventana gráfica 179
                        quita(tope);        // Quita ventana gráfica 172
                        quita(tope);        // Quita ventana gráfica 171
                        // Lee los nuevos datos
                        if (i != -1) then
                            {
                            switch(i)
                             {
                              case 193:// Antiguedad en la UNAM
                                        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                                        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                                        texto("Nueva antiguedad en la UNAM . .
.",173,RED,SMALL_FONT,1,1,2,1);
                                        lee_cadena(antunam,5,174,RED,SMALL_FONT,1,1,2,1);
                                        texto(antunam,174,RED,SMALL_FONT,1,1,2,1);
                                        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                                        texto("Presione una tecla para
continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                        bioskey(0);
                                        quita(tope);        // Quita ventana gráfica 179
                                        quita(tope);        // Quita ventana gráfica 174
                                        quita(tope);        // Quita ventana gráfica 173
                                        for (k=0; antunam[k]; k++)
                                        antunam[k] = toupper(antunam[k]);
                                        strcpy((*r).antunam,antunam);        // Asigna antunam en r
                                        break;

                              case 194:// Antigüedad en la ENEP
                                        pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
                                        pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
                                        texto("Nueva antigüedad en la ENEP . .
.",173,RED,SMALL_FONT,1,1,2,1);
                                        lee_cadena(antenep,5,174,RED,SMALL_FONT,1,1,2,1);
                                        texto(antenep,174,RED,SMALL_FONT,1,1,2,1);
                                        pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
                                        texto("Presione una tecla para
continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                                        bioskey(0);
                                        quita(tope);        // Quita ventana gráfica 179
                                        quita(tope);        // Quita ventana gráfica 174
                                        quita(tope);        // Quita ventana gráfica 173
                                        for (k=0; antenep[k]; k++)
                                        antenep[k] = toupper(antenep[k]);
                                        strcpy((*r).antenep,antenep);   // Asigna antenep en r
                             }
                            }
                        }
                         i=216;
                         pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                        break;

                    case 217: // Despliega en pantalla el Teléfono de la oficina
                        pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
                        pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
                        texto("Teléfonos actuales . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                        strcpy(numero,"Ofc. ");
```

```
            strcat(numero,(*r).telofc);
            strcat(numero," Casa ");
            strcat(numero,(*r).telcasa);
            texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
            // Lee el nuevo teléfono de la oficina
            pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
            pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
            texto("Nuevo teléfono de la oficina . . .",173,RED,SMALL_FONT,1,1,2,1);
            lee_cadena(telofc,14,174,RED,SMALL_FONT,1,1,2,1);
            texto(telofc,174,RED,SMALL_FONT,1,1,2,1);
            quita(tope);        // Quita ventana gráfica 174
            quita(tope);        // Quita ventana gráfica 173

            // Despliega en pantalla el teléfono de la casa
            pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
            pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
            texto("Nuevo teléfono de la casa . . .",173,RED,SMALL_FONT,1,1,2,1);
            lee_cadena(telcasa,14,174,RED,SMALL_FONT,1,1,2,1);
            texto(telcasa,174,RED,SMALL_FONT,1,1,2,1);
            pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
            texto("Presione una tecla para continuar ...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
            biosKey(0);
            quita(tope);        // Quita ventana gráfica 179
            quita(tope);        // Quita ventana gráfica 174
            quita(tope);        // Quita ventana gráfica 173
            quita(tope);        // Quita ventana gráfica 172
            quita(tope);        // Quita ventana gráfica 171
            strcpy((*r).telofc,telofc);   // Asigna telofc y telcasa en r
            strcpy((*r).telcasa,telcasa);
            i=217;
            pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
            break;

case 218: // Desplegar en pantalla el número de salón
            pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
            texto("Numero de salón actual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            itoa((*r).numsalon,numero,10);
            texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
            // Lee el nuevo número de salón
            pon(tope,173,COPY_PUT);        // Pone ventana gráfica 173
            pon(tope,174,COPY_PUT);        // Pone ventana gráfica 174
            texto("Nuevo número de salón . . .",173,RED,SMALL_FONT,1,1,2,1);
            lee_cadena(numero,5,174,RED,SMALL_FONT,1,1,2,1);
            texto(numero,174,RED,SMALL_FONT,1,1,2,1);
            numsalon = atoi(numero);
            pon(tope,179,COPY_PUT);        // Pone ventana gráfica 179
            texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
            biosKey(0);
            quita(tope);        // Quita ventana gráfica 179
            quita(tope);        // Quita ventana gráfica 174
            quita(tope);        // Quita ventana gráfica 173
            quita(tope);        // Quita ventana gráfica 172
            quita(tope);        // Quita ventana gráfica 171
            (*r).numsalon = numsalon;          // Asigna numsalon a r
            i=218;
            pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
            break;


case 219: // Despliega en pantalla el tipo de salón
            pon(tope,171,COPY_PUT);        // Pone ventana gráfica 171
            pon(tope,172,COPY_PUT);        // Pone ventana gráfica 172
            texto("Tipo de salón acual. . .",171,BLUE,SMALL_FONT,1,1,2,1);
            c = (*r).tipo;
```

```
switch(c)
 {
   case 0: texto ("No se clasificó el salón",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
   case 1: texto ("Salón de dibujo",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
   case 2: texto ("Salón de laboratorio",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
   case 3: texto ("Salón chico (max 30 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
   case 4: texto ("Salón grande (max 60 alumnos)",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
   case 5: texto ("Salón sin clasificar",172,BLUE,SMALL_FONT,1,1,2,1);
           break;
 }
 // Selecciona el nuevo tipo de salón
pon(tope,173,COPY_PUT);       // Pone ventana gráfica 173
texto("Nuevo tipo de salón",173,RED,SMALL_FONT,1,1,2,1);
pon(tope,175,COPY_PUT);       // Pone ventana gráfica 175
pon(tope,176,COPY_PUT);       // Pone ventana gráfica 176
pon(tope,177,COPY_PUT);       // Pone ventana gráfica 177
pon(tope,178,COPY_PUT);       // Pone ventana gráfica 178
texto("D",175,RED,SMALL_FONT,1,1,2,1);
texto("L",176,RED,SMALL_FONT,1,1,2,1);
texto("C",177,RED,SMALL_FONT,1,1,2,1);
texto("G",178,RED,SMALL_FONT,1,1,2,1);
pon(tope,175,NOT_PUT);        // Pone ventana gráfica del cursor
i = opciones1(175,175,178,24);
quita(tope);       // Quita ventana gráfica
quita(tope);       // Quita ventana gráfica 178
quita(tope);       // Quita ventana gráfica 177
quita(tope);       // Quita ventana gráfica 176
quita(tope);       // Quita ventana gráfica 175
pon(tope,174,COPY_PUT);       // Pone ventana gráfica 174
if (i == 175) then
 {
  texto("Salón de dibujo",174,RED,SMALL_FONT,1,1,2,1);
  tipo = 1;
 };
if (i == 176) then
 {
  texto("Salón de laboratorio",174,RED,SMALL_FONT,1,1,2,1);
  tipo = 2;
 };
if (i == 177) then
 {
  texto("Salón chico (máx. 30 alumnos)",174,RED,SMALL_FONT,1,1,2,1);
  tipo = 3;
 };
if (i == 178) then
 {
  texto("Salón grande (máx. 60 alumnos)",174,RED,SMALL_FONT,1,1,2,1);
  tipo = 4;
 };
if (i == -1) then
 {
  texto("Salón sin clasificar",174,RED,SMALL_FONT,1,1,2,1);
  tipo = 5;
 };
pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
bioskey(0);
quita(tope);       // Quita ventana gráfica 179
quita(tope);       // Quita ventana gráfica 174
quita(tope);       // Quita ventana gráfica 173
```

```
                    quita(tope);        // Quita ventana gráfica 172
                    quita(tope);        // Quita ventana gráfica 171
                    (*r).tipo = tipo;           // Asigna tipo a r
                    i = 219;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;

          case 220: // Despliega en pantalla el actual horario de clase
                    pon(tope,171,COPY_PUT);     // Pone ventana gráfica 171
                    pon(tope,172,COPY_PUT);     // Pone ventana gráfica 172
                    texto("Horario de clase actual . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                    strcpy(numero,(*r).hora1);
                    strcat(numero," - ");
                    strcat(numero,(*r).hora2);
                    texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                    // Lee el nuevo hoarario de clase
                    pon(tope,173,COPY_PUT);     // Pone ventana gráfica 173
                    pon(tope,174,COPY_PUT);     // Pone ventana gráfica 174
                    texto("Nueva hora de inicio . . .",173,RED,SMALL_FONT,1,1,2,1);
                    k = 1;
                    while (k == 1) do                   // Verifica que el dato sea correcto
                    {
                      lee_cadena(hora1,5,174,RED,SMALL_FONT,1,1,2,1);
                      k = hora_correcta(hora1,hora1,1);
                    }
                    texto(hora1,174,RED,SMALL_FONT,1,1,2,1);
                    quita(tope);        // Quita ventana gráfica 174
                    quita(tope);        // Quita ventana gráfica 173
                    pon(tope,173,COPY_PUT);     // Pone ventana gráfica 173
                    pon(tope,174,COPY_PUT);     // Pone ventana gráfica 174
                    texto("Nueva hora de terminación . . .",173,RED,SMALL_FONT,1,1,2,1);
                    k = 1;
                    while (k == 1) do                   // Verifica que el dato sea correcto
                    {
                     lee_cadena(hora2,5,174,RED,SMALL_FONT,1,1,2,1);
                     k = hora_correcta(hora1,hora2,2);
                    }
                    texto(hora2,174,RED,SMALL_FONT,1,1,2,1);
                    pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                    texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                    bioskey(0);
                    quita(tope);        // Quita ventana gráfica 179
                    quita(tope);        // Quita ventana gráfica 174
                    quita(tope);        // Quita ventana gráfica 173
                    quita(tope);        // Quita ventana gráfica 172
                    quita(tope);        // Quita ventana gráfica 171
                    strcpy((*r).hora1,hora1);   // Asigna hora1 y hora2 en r
                    strcpy((*r).hora2,hora2);
                    i=220;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;

          case 221: // Despliega en pantalla los Días
                    texto("LUNES",180,RED,SMALL_FONT,1,1,2,1);
                    texto("MARTES",181,RED,SMALL_FONT,1,1,2,1);
                    texto("MIERCOLES",182,RED,SMALL_FONT,1,1,2,1);
                    texto("JUEVES",183,RED,SMALL_FONT,1,1,2,1);
                    texto("VIERNES",184,RED,SMALL_FONT,1,1,2,1);
                    texto("SABADO",185,RED,SMALL_FONT,1,1,2,1);
                    texto("Días seleccionados...",186,RED,SMALL_FONT,1,1,2,1);
                    texto("Presione <ESC> cuando termine de
seleccionar",179,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    pon(tope,180,NOT_PUT);      // Pone ventana gráfica del cursor
                    m = 187;
                    i = 180;
```

```
k = 0;
while (i != -1)
 {
      i = opciones1(i,160,165,21);  // Selección de opciones
      switch(i)
       {
        case 160: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Lunes",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=1;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
        case 161: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Martes",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=2;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
        case 162: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Miercoles",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=3;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
        case 163: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Jueves",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=4;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
        case 164: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Viernes",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=5;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
        case 165: quita(tope);       // Quita ventana gráfica del cursor
                  texto("Sabado",m,RED,SMALL_FONT,1,1,2,1);
                  dias[k]=6;
                  m = m + 1;
                  k = k + 1;
                  pon(tope,i,NOT_PUT);       // Pone ventana gráfica del cursor
                  break;
       };
 };
pon(tope,179,COPY_PUT);       // Pone ventana gráfica 179
texto("Presione una tecla para continuar...",179,LIGHTRED,SMALL_FONT,1,1,2,1);
bioskey(0);
quita(tope);       // Quita ventana gráfica 179
dias[k] = 10;
// Quita ventanas restantes
quita(tope);       // Quita ventana gráfica del cursor
for (k = 179; k <= m-1; k++)
 {
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 }
 for (k=0; dias[k] <= 10; k++)
  (*r).dias[k] = dias[k];
i = 221;
```

```
                        pon(tope,l,NOT_PUT);        // Pone ventana gráfica del cursor
                        break;

                case 222: // Despliega en pantalla el total de alumnos inscritos
                        pon(tope,171,COPY_PUT);     // Pone ventana gráfica 171
                        pon(tope,172,COPY_PUT);     // Pone ventana gráfica 172
                        texto("Total de alumnos inscritos actualmente . . .",171,BLUE,SMALL_FONT,1,1,2,1);
                        itoa((*r).numalumnos,numero,10);
                        texto(numero,172,BLUE,SMALL_FONT,1,1,2,1);
                        // Lee del teclado el nuevo total de alumnos inscritos
                        pon(tope,173,COPY_PUT);     // Pone ventana gráfica 173
                        pon(tope,174,COPY_PUT);     // Pone ventana gráfica 174
                        texto("Nuevo total de alumnos inscritos . . .",173,RED,SMALL_FONT,1,1,2,1);
                        lee_cadena(numero,3,174,RED,SMALL_FONT,1,1,2,1);
                        texto(numero,174,RED,SMALL_FONT,1,1,2,1);
                        numalumnos=atoi(numero);
                        pon(tope,179,COPY_PUT);     // Pone ventana gráfica 179
                        texto("Presione una tecla para continuar",179,LIGHTRED,SMALL_FONT,1,1,2,1);
                        bioskey(0);
                        quita(tope);        // Quita ventana gráfica 179
                        quita(tope);        // Quita ventana gráfica 174
                        quita(tope);        // Quita ventana gráfica 173
                        quita(tope);        // Quita ventana gráfica 172
                        quita(tope);        // Quita ventana gráfica 171
                        (*r).numalumnos = numalumnos;   // Asigna numalumnos a r
                        i=222;
                        pon(tope,l,NOT_PUT);        // Pone ventana gráfica del cursor
                        break;
                }
    }
  }
quita(tope);        // Quita ventana gráfica del cursor
  for (k=210; k <= 222; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  for (k=152; k <= 154; k++)
    {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
quita(tope);        // Quita ventana gráfica 150
}


//  Crea una nueva lista enlazada GRUPO. Inicializa la lista eliminando la lista existente.
void crear_grupos(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct GRUPO m;
 int dato,i,ultimo,k;

 r=p;
 pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
 texto("CREANDO UNA NUEVA LISTA DE GRUPOS",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(21);
  quita(tope);        // Quita ventana gráfica 100
  return;
 }
   i = 0;
   dato = 0;
   while (i != -1) do                   // Selecciona la forma de consultar MATERIA
```

```
    {
          r = p;
          dato = escoje_consulta_materia(r,&l);
          if (l != -1) then
           {
           if (l != 351) then
            {
              while ((*r).num != dato)
                        r = (*r).materias;
              k = 1;
              if ((*r).grupos != NULL) then
                    k = confirmar(1);
              if (k == 0) then
               {
                    quita(tope);        // Quita ventana gráfica 100
                    return;
               }
              lee_nodo_grupos(&m,(*r).grupos);              // Lee los datos para un nuevo nodo
              (*r).grupos = crea_lista_grupos();            // Crea la nueva lista
              pon_grupos((*r).grupos,m);                    // Pone el nuevo nodo en la lista
              while (r != NULL)
                    r = (*r).materias;
            }
           else
            {
              s = p;
              ultimo = 0;
              while (s != NULL) do
                    {
                     ultimo = ultimo + 1;
                     s = (*s).materias;
                    }
              while (dato != -2) do                 // Selección de una materia de la lista MATERIA
              {
                    r = p;
                    dato=mover_pantallas_consulta(120,120,231,p,ultimo);
                    if (dato > 0) then
                     {
                       while ((*r).num != dato)
                                 r = (*r).materias;
                       k = 1;
                       if ((*r).grupos != NULL) then
                             k = confirmar(1);
                       if (k == 0) then
                        {
                         quita(tope);       // Quita ventana gráfica 100
                         return;
                        }
                       lee_nodo_grupos(&m,(*r).grupos);     // Lee los datos para el nuevo nodo
                       (*r).grupos = crea_lista_grupos();   // Crea la nueva lista GRUPO
                       pon_grupos((*r).grupos,m);           // Pone el nuevo nodo en la lista
                       while (r != NULL)
                        r = (*r).materias;
                      }
              }
            }
          }
    }
 quita(tope);       // Quita ventana gráfica 100
}

//    Añade nuevos nodos a la lista enlazada GRUPO.
void agregar_grupos(struct MATERIA *p)
{
 struct MATERIA *r,*s;
```

```
struct GRUPO m;
int dato,c,ultimo,i;

pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
texto("AGREGANDO UN NUEVO GRUPO A LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
if ((*p).num == 0) then
{
llama_error(22);
quita(tope);        // Quita ventana gráfica 100
return;
}
i = 0;
dato = 0;
while (i != -1) do              // Seleccione la forma de consultar MATERIA
 {
   r = p;
   dato = escoje_consulta_materia(r,&i);
   if (i != -1) then
    {
        if (i != 351) then
          {
           while ((*r).num != dato)
                    r = (*r).materias;
           if ((*r).grupos == NULL) then
           {
            llama_error(25);
           }
           else
            {
                  while (dato != -3) do
                   {
                     lee_nodo_grupos(&m,(*r).grupos);          // Lee los datos del nuevo nodo
                     pon_grupos((*r).grupos,m);                // Añade el nuevo nodo en la lista
                     dato = salir_lee_grupo();
                   }
            }
           while (r != NULL)
                    r = (*r).materias;
          }
        else
         {
          s = p;
          ultimo = 0;
          while (s != NULL) do
           {
                 ultimo = ultimo + 1;
                 s = (*s).materias;
           }
          while (dato != -2) do                  // Selección de una materia de la lista MATERIA
          {
           r = p;
            dato=mover_pantallas_consulta(120,120,231,p,ultimo);
            if (dato > 0) then
                  {
                   while ((*r).num != dato)
                            r = (*r).materias;
                   if ((*r).grupos == NULL) then
                   {
                    llama_error(25);
                   }

                   else
                    {
                      while (dato != -3) do
                           {
```

```
                                lee_nodo_grupos(&m,(*r).grupos);    // Lee los datos de un nuevo nodo
                                pon_grupos((*r).grupos,m);          // Añade un nuevo nodo en la lista
                                dato = salir_lee_grupo();
                              }
                        }
                        while (r != NULL)
                                r = (*r).materias;
                      }
              }
            }
      }
    }
 quita(tope);       // Quita ventana gráfica 100
}


//    Llama a las funciones que eliminan un nodo dentro de la lista enlazada GRUPO.
void eliminar_grupos(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct GRUPO *g;
 int dato,c,ultimo,i,k;

 pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
 texto("ELIMINANDO UN GRUPO DE LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(23);
  quita(tope);        // Quita ventana gráfica 100
  return;
 }
 i = 0;
 dato = 0;
 while (i != -1) do                     // Selecciona la forma de consultar MATERIA
   {
    r = p;
    dato = escoje_consulta_materia(r,&i);
    if (i != -1) then
     {
         if (i != 351) then
           {
            while ((*r).num != dato)
                    r = (*r).materias;
            if ((*r).grupos == NULL) then
            {
              llama_error(26);
            }
            else
             {
                    k = 0;
                    g = (*r).grupos;
                    while (g != NULL) do
                    {
                      k = k + 1;
                      g = (*g).apun;
                    }
                    c = 0;
                    while (c != -2) do              // Selecciona un nodo de la lista GRUPO
                    {
                      c = mover_pantallas_grupos(121,121,152,(*r).grupos,k);

                      if (c > 0) then
                       {
                        k = confirmar(2);
                        if (k == 1) then
```

```
                    {
                     c = quita_grupos((*r).grupos,c);        // Elimina el nodo de la lista
                     if (c == 0) then
                      {
                       (*r).grupos = NULL;
                       c = -2;
                       i = -1;
                      }
                    }
                }
              }
    }
    while (r != NULL)
            r = (*r).materias;
  }
 else
  {
   s = p;
   ultimo = 0;
   while (s != NULL) do
    {
          ultimo = ultimo + 1;
          s = (*s).materias;
    }
   while (dato != -2) do                    // Consulta toda la lista MATERIA
   {
    r = p;
    dato=mover_pantallas_consulta(120,120,231,p,ultimo);
    if (dato > 0) then
          {
           while ((*r).num != dato)
                    r = (*r).materias;
           if ((*r).grupos == NULL) then
           {
            llama_error(26);
           }
           else
           {
             k = 0;
             g = (*r).grupos;
             while (g != NULL) do
                   {
                    k = k + 1;
                    g = (*g).apun;
                   }
             c = 0;
             while (c != -2) do                    // Selecciona un nodo en la lista GRUPO
                   {
                    c = mover_pantallas_grupos(121,121,152,(*r).grupos,k);
                    if (c > 0) then
                    {
                     k = confirmar(2);
                     if (k == 1) then
                      {
                            c = quita_grupos((*r).grupos,c);        // Elimina el nodo de la lista
                           if (c == 0) then
                            {
                             (*r).grupos = NULL;
                             c = -2;
                             dato = -2;
                             i = -1;
                            }
                      }
                    }
                   }
```

```
                    }
                    while (r != NULL)
                            r = (*r).materias;
                  }
              }
            }
      }
    }
 quita(tope);       // Quita ventana gráfica 100
}


//   Llama a las funciones que modifican la información de los nodos de la lista enlazada GRUPO.
void cambiar_grupos(struct MATERIA *p)
{
 struct MATERIA *r,*s;
 struct GRUPO *g;
 int dato,c,i,ultimo,k;

 pon(tope,100,COPY_PUT);        // Pone ventana gráfica 100
 texto("CAMBIANDO DATOS DE UN GRUPO DE LA LISTA",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 if ((*p).num == 0) then
 {
  llama_error(24);
  quita(tope);       // Quita ventana gráfica 100
  return;
 }
 i = 0;
 dato = 0;
 while (i != -1) do                    // Selecciona la forma de consultar MATERIA
   {
    r = p;
    dato = escoje_consulta_materia(r,&i);
    if (i != -1) then
     {
         if (i != 351) then
           {
            while ((*r).num != dato)
                    r = (*r).materias;
            if ((*r).grupos == NULL) then
            {
             llama_error(27);
            }
            else
             {
              k = 0;
              g = (*r).grupos;
              while (g != NULL) do
                    {
                      k = k + 1;
                      g = (*g).apun;
                    }
              c = 0;
              while (c != -2) do                    // Selecciona un nodo de la lista GRUPO
                    {
                      c = mover_pantallas_grupos(121,121,152,(*r).grupos,k);
                      if (c > 0) then
                        cambio_lista_grupos((*r).grupos,c); // Cambia los elementos del nodo
                    }
             }

            while (r != NULL)
                  r = (*r).materias;
           }
         else
```

```
                {
                 s = p;
                 ultimo = 0;
                 while (s != NULL) do
                  {
                        ultimo = ultimo + 1;
                        s = (*s).materias;
                  }
                 while (dato != -2) do                    // Consulta toda la lista MATERIA
                 {
                  r = p;
                   dato=mover_pantallas_consulta(120,120,231,p,ultimo);
                   if (dato > 0) then
                         {
                          while ((*r).num != dato)
                                  r = (*r).materias;
                          if ((*r).grupos == NULL) then
                          {
                           llama_error(27);
                          }
                          else
                          {
                           k = 0;
                           g = (*r).grupos;
                           while (g != NULL) do
                           {
                                 k = k + 1;
                                 g = (*g).apun;
                           }
                           c = 0;
                           while (c != -2) do          // Selecciona un nodo de la lista GRUPO
                                {
                                 c = mover_pantallas_grupos(121,121,152,(*r).grupos,k);
                                 if (c > 0) then
                                   cambio_lista_grupos((*r).grupos,c); // Cambia los elementos del nodo
                                }
                          }
                          while (r != NULL)
                                  r = (*r).materias;
                         }
                 }
                }
        }
    }
 quita(tope);       // Quita ventana gráfica 100
}

//   Llama a las funciones que realizan operaciones sobre la lista enlazada GRUPO.
void grupos(int i, struct MATERIA *lista)
{
 cuadro(61,GREEN);
 switch(i)
   {
   case 44: crear_grupos(lista);          // Crea una nueva lista enlazada GRUPO
            break;
   case 45: agregar_grupos(lista);        // Añade nuevos nodos en la lista enlazada GRUPO
            break;
   case 46: eliminar_grupos(lista);       // Elimina nodos de la lista enlazada GRUPO
            break;
   case 47: cambiar_grupos(lista);        // Cambia datos de los nodos de la lista enlazada GRUPO
            break;
   }
 cuadro(61,LIGHTBLUE);
}
```

B.3.9. G_TODOS.CCP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\grupos.h"
#include "d:\g_todos.h"
#include "d:\consulta.h"
#include "d:\errores.h"
#include "d:\ayuda.h"

#define  then


//    Busca un elemento  materia  en cada nodo de la lista enlazada MATERIA
struct MATERIA * busca_materia(struct MATERIA *p, int materia)
{
 while (p != NULL) do
  {
   if ((*p).num == materia) then
     return(p);
   else
     p = (*p).materias;
  }
 return(NULL);
}

/*    Despliega en pantalla los datos de la lista enlazada GRUPO. Inicia en la posición inicio hasta la posición fin.
  Regresa la última ventana colocada. */
int ventanas_ponen_datos(struct GRUPO *g, int inicio, int fin, char nombreprofe[100])
{
 struct GRUPO *r;
 int i,c,k,n;
 char numero[20],cadena[100];

 r=g;
 i = 274;
 while ((*r).num != inicio) do
          r=(*r).apun;
for (k=inicio; k <= fin; k++)
{
  if (!strcmp(nombreprofe,(*r).nombreprofe)) then
   {
          pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
          itoa((*r).numgrupo,numero,10);
          texto(numero,i,LIGHTBLUE,SMALL_FONT,1,1,2,1);
          i=i+1;
          pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
          itoa((*r).numsalon,numero,10);
          texto(numero,i,LIGHTBLUE,SMALL_FONT,1,1,2,1);
          i=i+1;
```

```
            pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
            strcpy(cadena,"... ");
            for (n = 0; (*r).dias[n] <= 10; n++)
            {
             if ((*r).dias[n] == 10) then
              break;
             c = (*r).dias[n];
             switch (c)
              {
                case 1: strcat(cadena," Lu");
                        break;
                case 2: strcat(cadena," Ma");
                        break;
                case 3: strcat(cadena," Mi");
                        break;
                case 4: strcat(cadena," Ju");
                        break;
                case 5: strcat(cadena," Vi");
                        break;
                case 6: strcat(cadena," Sá");
                        break;
              }
             }
            texto(cadena,i,LIGHTBLUE,SMALL_FONT,1,1,2,1);
            i=i+1;
            pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
            strcpy(numero,(*r).hora1);
            strcat(numero," a ");
            strcat(numero,(*r).hora2);
            texto(numero,i,LIGHTBLUE,SMALL_FONT,1,1,2,1);
            i = i+1;
    }
    r = (*r).apun;
  }
  return(i);
}

/*     Despliega en pantalla los datos de la lista enlazada GRUPO. Inicia en la posición inicio hasta la posición fin.
  Regresa la última ventana colocada. */
int ventanas_ponen_datos_salones(struct GRUPO *g, int inicio, int fin, long int numsalon)
{
 struct GRUPO *r;
 int i,c,k,n;
 char numero[20],cadena[100];

 r=g;
 i = 294;
 while ((*r).num != inicio) do
        r=(*r).apun;
for (k=inicio; k <= fin; k++)
 {
  if (numsalon == (*r).numsalon) then
   {
        pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
        itoa((*r).numgrupo,numero,10);
        texto(numero,i,RED,SMALL_FONT,1,1,2,1);
        i=i+1;
        pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
        texto((*r).nombreprofe,i,RED,SMALL_FONT,1,1,2,1);
        i=i+1;
        pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
        strcpy(cadena,"... ");
        for (n = 0; (*r).dias[n] <= 10; n++)
        {
         if ((*r).dias[n] == 10) then
```

```
                    break;
                  c = (*r).dias[n];
                  switch (c)
                  {
                    case 1: strcat(cadena," Lu");
                            break;
                    case 2: strcat(cadena," Ma");
                            break;
                    case 3: strcat(cadena," MI");
                            break;
                    case 4: strcat(cadena," Ju");
                            break;
                    case 5: strcat(cadena," Vi");
                            break;
                    case 6: strcat(cadena," Sá");
                            break;
                  }
                 }
                 texto(cadena,l,RED,SMALL_FONT,1,1,2,1);
                 l=l+1;
                 pon(tope,l,COPY_PUT);          // Pone ventana gráfica para l
                 strcpy(numero,(*r).hora1);
                 strcat(numero," a ");
                 strcat(numero,(*r).hora2);
                 texto(numero,l,RED,SMALL_FONT,1,1,2,1);
                 l = l+1;
      }
      r = (*r).apun;
   }
   return(i);
}

//      Busca numero en la lista enlazada GRUPO
void busca_grupo_tipo(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).clasegrupo == clase) then
            {
             if ((*s).numgrupo == numero) then
              {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
              }
            }
     s = (*s).apun;
     }
  r = (*r).materias;
 }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
 arreglo[j].numero = -2;
}
```

```
//      Busca numero en la lista enlazada GRUPO
void busca_grupo_cadena_tipo(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],grupo[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
  s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).clasegrupo == clase) then
            {
             dato = 1;
            itoa((*s).numgrupo,grupo,10);
            for (n=0; n < k && dato == 1; n++)
              {
               if (num[n] == grupo[n]) then
                 dato =1;
               else
                 dato = 0;
              }
            if (dato == 1) then
              {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
              }
            }
     s = (*s).apun;
     }
  r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

//      Busca clase en la lista enlazada GRUPO
void busca_grupo_todo_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
  s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).clasegrupo == clase) then
            {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
```

```
            }
        s = (*s).apun;
       }
     r = (*r).materias;
    }
    arreglo[j].grupo = -2;
    arreglo[j].materia = -2;
    arreglo[j].numero = -2;
}

//      Busca clase en la lista enlazada GRUPO
void busca_grupo_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
     if ((*s).clasegrupo == clase) then
            {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
            }
     s = (*s).apun;
     }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

//      Busca numero en la lista enlazada GRUPO
void busca_grupo_teolab(struct MATERIA *p, int numero, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            if ((*s).numgrupo == numero) then
             {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
             }
     s = (*s).apun;
     }
  r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}
```

```
//      Busca numero en la lista enlazada GRUPO
void busca_grupo_cadena_teolab(struct MATERIA *p, int numero, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],grupo[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            itoa((*s).numgrupo,grupo,10);
            for (n=0; n < k && dato == 1; n++)
              {
              if (num[n] == grupo[n]) then
                dato =1;
              else
                dato = 0;
              }
            if (dato == 1) then
              {
               arreglo[j].grupo = (*s).num;
               arreglo[j].materia = (*r).num;
               arreglo[j].numero = (*s).numgrupo;
               j = j + 1;
              }
     s = (*s).apun;
     }
  r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

void busca_grleolab_todo(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
              arreglo[j].grupo = (*s).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*s).numgrupo;
              j = j + 1;
     s = (*s).apun;
     }
  r = (*r).materias;
 }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
```

```
  arreglo[j].numero = -2
}

void busca_gleolab_materia(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
          arreglo[j].grupo = (*s).num;
          arreglo[j].materia = (*r).num;
          arreglo[j].numero = (*s).numgrupo;
          j = j + 1;
          s = (*s).apun;
     }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

//  Cuenta los elementos del arreglo ARREGLO
void busca_grupo_tipo_cuenta(struct MATERIA *p, int numero, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).clasegrupo == clase) then
            {
             if ((*s).numgrupo == numero) then
               j = j + 1;
            }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

// Busca clase en la lista enlazada GRUPO
void busca_grupo_cadena_tipo_cuenta(struct MATERIA *p, int numero, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],grupo[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
```

```
   s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).clasegrupo == clase) then
            {
            dato = 1;
            itoa((*s).numgrupo,grupo,10);
            for (n=0; n < k && dato == 1; n++)
              {
              if (num[n] == grupo[n]) then
                dato =1;
              else
                dato = 0;
              }
            if (dato == 1) then
               j = j + 1;
            }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

// Busca clase en la lista enlazada GRUPO
void busca_grupo_todo_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).clasegrupo == clase) then
               j = j + 1;
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

// Busca clase en la lista enlazada GRUPO
void busca_grupo_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).clasegrupo == clase) then
               j = j + 1;
      s = (*s).apun;
     }
 *cuenta = j;
}
```

```
// Busca numero en la lista enlazada GRUPO
void busca_grupo_teolab_cuenta(struct MATERIA *p, int numero, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            if ((*s).numgrupo == numero) then
              j = j + 1;
     s = (*s).apun;
     }
  r = (*r).materias;
  }
 *cuenta = j;
}

// Busca numero en la lista enlazada GRUPO
void busca_grupo_cadena_teolab_cuenta(struct MATERIA *p, int numero, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],grupo[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            itoa((*s).numgrupo,grupo,10);
            for (n=0; n < k && dato == 1; n++)
              {
               if (num[n] == grupo[n]) then
                 dato =1;
               else
                 dato = 0;
              }
            if (dato == 1) then
              j = j + 1;
     s = (*s).apun;
     }
  r = (*r).materias;
  }
 *cuenta = j;
}

void busca_gteolab_todo_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
```

```
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
          j = j + 1;
          s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

void busca_gteolab_materia_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
          j = j + 1;
          s = (*s).apun;
     }
 *cuenta = j;
}

//     Crea ventanas de grupos
void cuadros_grupos_todos(int tipo)
{

  switch(tipo)
   {
          case 310: texto("GRUPOS DE TEORIA",316,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    texto("MATERIA CORRESPONDIENTE AL GRUPO",319,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    break;
          case 311: texto("GRUPOS DE LAB.",318,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    texto("MATERIA CORRESPONDIENTE AL GRUPO DE LAB.",319,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    break;
          case 312: texto("GRUPOS",318,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    texto("MATERIA CORRESPONDIENTE AL GRUPO",319,LIGHTBLUE,SMALL_FONT,1,1,2,1);
                    break;
   }
  texto("PAG. 1",240,BLUE,SANS_SERIF_FONT,1,2,1,2);
  cuadro(150,BLUE);
  texto("Seleccione un grupo para obtener más información",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("<TAB> cambia menú",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("PgUp/PgDn",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
}

//  Despliega en pantalla los datos de la lista enlazada GRUPO. Regresa la posición de la última ventana colocada.
int ventanas_grupos_todos(struct MATERIA *p,int inicio, int fin, struct ARREGLO *arreglo,int *max,int tipo)
{
 struct MATERIA *s,*r;
 struct GRUPO *g;
 char numero[100],cadena[100];
 int i,j,k,c;

 r = p;
 i = 259;
 j = *max;
```

```
 for (k=inicio; k <= fin; k++)
   {
    s = busca_materia(r,arreglo[k].materia);
    g = (*s).grupos;
    while ((*g).num != arreglo[k].grupo) do
            g = (*g).apun;
    cuadro(j,CYAN);
    pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
    itoa((*g).numgrupo,numero,10);
    if (tipo == 312) then
     {
          if ((*g).clasegrupo == 1) then
            {
             strcpy(cadena,numero);
             strcat(cadena," Teo.");
            }
          if ((*g).clasegrupo == 2) then
            {
             strcpy(cadena,numero);
             strcat(cadena," Lab.");
            }
           texto(cadena,i,RED,SMALL_FONT,3,2,2,1);
         }
     else
          texto(numero,i,RED,SMALL_FONT,3,2,2,1);
    i = i + 1;
    if (s != NULL) then
         {
          pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
          texto((*s).nombre,i,RED,SMALL_FONT,3,2,2,1);
         }
    i = i + 1;
    *max=j;
    j = j+1;
   }
 return(j);
}

/*   Llama a las funciones que despliegan los datos de la lista enlazada GRUPO. Regresa la posición del nodo de la
  lista GRUPO seleccionado.*/
int grupos_teoria(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int b, int tipo, int cuenta)
{
 struct MATERIA *r;
 struct POS
          {
           int inicio;
           int fin;
          };
 int k, i,regresa,numc,o,ultimo,grupo,cuot,rem,j,a,inicio,fin,dato,max;
 char sig[10],cad[10];
 struct POS *pos;

 // Reserva espacio en memoria para pos
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
  {
   printf("Error.");
   exit(1);
  }
 arreglo[0].grupo = 0;
 arreglo[0].materia = 0;
 arreglo[0].numero = 0;
 r = p;
 switch(tipo)
  {
```

```
        case 310: if (b == 0) then
                    busca_grupo_tipo(r,numero,arreglo,1);
                  if (b == 1) then
                    busca_grupo_cadena_tipo(r,numero,arreglo,1);
                  if (b == 2) then
                    busca_grupo_todo_tipo(r,arreglo,1);
                  if (b == 3) then
                    busca_grupo_materia_tipo(r,arreglo,1);
                  break;
        case 311: if (b == 0) then
                    busca_grupo_tipo(r,numero,arreglo,2);
                  if (b == 1) then
                    busca_grupo_cadena_tipo(r,numero,arreglo,2);
                  if (b == 2) then
                    busca_grupo_todo_tipo(r,arreglo,2);
                  if (b == 3) then
                    busca_grupo_materia_tipo(r,arreglo,2);
                  break;
        case 312: if (b == 0) then
                    busca_grupo_teolab(r,numero,arreglo);
                  if (b == 1) then
                    busca_grupo_cadena_teolab(r,numero,arreglo);
                  if (b == 2) then
                    busca_gteolab_todo(r,arreglo);
                  if (b == 3) then
                    busca_gteolab_materia(r,arreglo);
                  break;
 }
ultimo = 0;
for (k=0; arreglo[k].grupo != -2; k++)
    ultimo = ultimo + 1;
quot = (ultimo/3);
rem = (ultimo%3);
ultimo = ultimo - 1;
if (ultimo == -1) then
  ultimo = 0;
ordena_arreglo(arreglo,ultimo);          // Ordena el arreglo antes de desplegarlo en pantalla
if (rem != 0) then
  grupo = quot + 1;
else
  grupo = quot;
if (grupo == 0) then
 {
  llama_error(29);
  return(-2);
 }
k = 0;
for (j=1; j <= grupo; j++)
 {
  for (a=0; arreglo[k].grupo != -2 && a < 3; k++, a++)
       {
        fin = k;
        if (a == 0) then
          inicio = k;
       }
  pos[j].inicio = inicio;
  pos[j].fin = fin;
 }
cuadros_grupos_todos(tipo);
j = 1;
max = 122;
inicio = pos[j].inicio;
fin = pos[j].fin;
numc = ventanas_grupos_todos(p,inicio,fin,arreglo,&max,tipo);
i = 122;
```

```
c=END;
regresa=0;
pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1: llama_ayuda(52);     // Llama al módulo AYUDA
                 break;
                                      // Permite desplazar el cursor sobre las ventanas
        case ABAJO:  quita(tope);          // Quita ventana gráfica del cursor
                     i++;
                     if (i == (max + 1)) then
                           i = 122;
                     pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     break;
        case ARRIBA: quita(tope);          // Quita ventana gráfica del cursor
                     i--;
                     if (i == (122 - 1)) then
                           i = max;
                     pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     break;
        case TAB:    if ((i >= 122) && (i <= max)) then              // Cambia al menú inferior
                     {
                      regresa=i;
                      i=237;
                      i=mover_ventana_menu(i,i,237,regresa);
                      if (i == 237) then
                            {
                             c = ENTER;
                             dato = -2;
                            }
                     }
                     break;
                                                    // Cambia a la siguiente página
        case PGDN:   quita(tope);          // Quita ventana gráfica del cursor
                     numc--;
                     for (k=259; k <= numc; k++)
                           quita(tope);          // Quita ventanas gráficas restantes
                     for (k = 122; k <= max; k++)
                           {
                            setfillstyle(SOLID_FILL,GREEN);
                            bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                           }
                     j++;
                     if (j == (grupo+1)) then
                      j=1;
                     inicio=pos[j].inicio;
                     fin=pos[j].fin;
                     max=122;
                     numc = ventanas_grupos_todos(p,inicio,fin,arreglo,&max,tipo);
                     strcpy(sig,"PAG. ");
                     itoa(j,cad,10);
                     strcat(sig,cad);
                     texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                     if (i > max) then
                      {
                       i = max;
                       pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                      }
                     else
                      pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     break;
                                                    // Cambia a la página anterior
```

```
        case PGUP:  quita(tope);        // Quita ventana gráfica // cursor
                    numc--;
                    for (k=259; k <= numc; k++)
                        quita(tope);        // Quita ventanas gráficas restantes
                    for (k = 122; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                    j--;
                    if (j == 0) then
                     j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=122;
                    numc = ventanas_grupos_todos(p,inicio,fin,arreglo,&max,tipo);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                    if (i > max) then
                    {
                     i = max;
                     pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    }
                    else
                     pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
    }
   }
   quita(tope);        // Quita ventana gráfica del cursor
   numc--;
   for (k=259; k <= numc; k++)
     quita(tope);        // Quita ventanas gráficas restantes
   for (k = 316; k <= 319; k++)
     {
      setfillstyle(SOLID_FILL,GREEN);
      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
   for (k = 122; k <= max; k++)
     {
      setfillstyle(SOLID_FILL,GREEN);
      bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
     }
   texto("MATERIA SELECCIONADA",240,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
   if (dato != -2) then
    {
     dato = i - 122;
     a = pos[j].inicio;
     a = a + dato;
    }
   else
    a = dato;
   farfree(pos);                    // Libera el espacio asignado para pos
   return(a);                       // Regresa la posición del nodo seleccionado
}


//  Muestra en pantalla los datos del nodo en la posición a de la lista enlazada GRUPO.
void consulta_datos_grupos_todos(struct MATERIA *p, struct ARREGLO *arreglo,int a, int b, char semestre[10])
{
 struct GRUPO *s;
 struct MATERIA *r;
 int c, k;
 char cadena[100], numero[100];
```

```
r = p;
while ((*r).num != arreglo[a].materia) do          // Se recorre la lista hasta la posició a
    r = (*r).materias;
s = (*r).grupos;
while ((*s).num != arreglo[a].grupo) do
        s = (*s).apun;
if (b == 3) then
  quita(tope);          // Quita ventana gráfica 240
for (k=236; k <= 240; k++)
   {
        setfillstyle(SOLID_FILL,GREEN);
        bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
c = DERECHA;
while (c != ESC) do
{
  // Grupo
  pon(tope,247,COPY_PUT);          // Pone ventana gráfica 247
  strcpy(cadena,"GRUPO . . .  ");
  itoa((*s).numgrupo,numero,10);
  strcat(cadena,numero);
  texto(cadena,247,LIGHTRED,SMALL_FONT,2,1,2,1);
  // Clase de grupo
  pon(tope,248,COPY_PUT);          // Pone ventana gráfica 248
  c = (*s).clasegrupo;
  switch (c)
  {
    case 1: strcpy(cadena,"GRUPO DE TEORIA");
            strcat(cadena,"  SEMESTRE ... ");
            strcat(cadena,semestre);
            texto(cadena,248,LIGHTRED,SMALL_FONT,1,1,2,1);
            break;
    case 2: strcpy(cadena,"GRUPO DE LABORATORIO");
            strcat(cadena,"  SEMESTRE ... ");
            strcat(cadena,semestre);
            texto(cadena,248,LIGHTRED,SMALL_FONT,1,1,2,1);
            break;
  }
  cuadro(286,CYAN);
  // Materia
  pon(tope,249,COPY_PUT);          // Pone ventana gráfica 249
  texto((*r).nombre,249,LIGHTBLUE,SMALL_FONT,1,1,2,1);
  // Clave
  pon(tope,250,COPY_PUT);          // Pone ventana gráfica 250
  strcpy(cadena,"CLAVE ...  ");
  itoa((*r).clave,numero,10);
  grupo_correcto(numero);
  strcat(cadena,numero);
  texto(cadena,250,LIGHTBLUE,SMALL_FONT,1,1,2,1);
  // Profesor
  pon(tope,251,COPY_PUT);          // Pone ventana gráfica 251
  strcpy(cadena,"PROF... ");
  strcat(cadena,(*s).nombreprofe);
  texto(cadena,251,LIGHTBLUE,SMALL_FONT,1,1,2,1);
  // R.F.C.
  pon(tope,252,COPY_PUT);          // Pone ventana gráfica 252
  strcpy(cadena,"RFC...  ");
  strcat(cadena,(*s).rfc);
  texto(cadena,252,LIGHTBLUE,SMALL_FONT,1,1,2,1);
  // Salón
  pon(tope,253,COPY_PUT);          // Pone ventana gráfica 253
  strcpy(cadena,"Salón . . .  ");
  itoa((*s).numsalon,numero,10);
  strcat(cadena,numero);
  texto(cadena,253,LIGHTBLUE,SMALL_FONT,1,1,2,1);
```

```
    // Total de alumnos inscritos
    pon(tope,254,COPY_PUT);          // Pone ventana gráfica 254
    strcpy(cadena,"Alumnos inscritos . . . ");
    itoa((*s).numalumnos,numero,10);
    strcat(cadena,numero);
    texto(cadena,254,LIGHTBLUE,SMALL_FONT,1,1,2,1);
    // Dias
    pon(tope,255,COPY_PUT);          // Pone ventana gráfica 255
    strcpy(cadena,"Dias... ");
    for (k = 0; (*s).dias[k] <= 10; k++)
    {
    if ((*s).dias[k] == 10) then
     break;
    c = (*s).dias[k];
    switch (c)
     {
           case 1: strcat(cadena," Lun");
                   break;
           case 2: strcat(cadena," Mar");
                   break;
           case 3: strcat(cadena," Mie");
                   break;
           case 4: strcat(cadena," Jue");
                   break;
           case 5: strcat(cadena," Vie");
                   break;
           case 6: strcat(cadena," Sáb");
                   break;
     }
    }
    texto(cadena,255,LIGHTBLUE,SMALL_FONT,1,1,2,1);
    // Horas
    pon(tope,256,COPY_PUT);          // Pone ventana gráfica 256
    strcpy(cadena,"Horario . . . ");
    strcat(cadena,(*s).hora1);
    strcat(cadena," a ");
    strcat(cadena,(*s).hora2);
    texto(cadena,256,LIGHTBLUE,SMALL_FONT,1,1,2,1);
    pon(tope,246,COPY_PUT);          // Pone ventana gráfica 246
    texto("Presione < ESC > para salir . . . . ",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
    c = bioskey(0);
  }
 quita(tope);          // Quita ventana gráfica 246
 quita(tope);          // Quita ventana gráfica 256
 quita(tope);          // Quita ventana gráfica 255
 quita(tope);          // Quita ventana gráfica 254
 quita(tope);          // Quita ventana gráfica 253
 quita(tope);          // Quita ventana gráfica 252
 quita(tope);          // Quita ventana gráfica 251
 quita(tope);          // Quita ventana gráfica 250
 quita(tope);          // Quita ventana gráfica 249
 k = 200;
 setfillstyle(SOLID_FILL,GREEN);
 bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 quita(tope);          // Quita ventana gráfica 248
 quita(tope);          // Quita ventana gráfica 247
 ventana_menu_consulta();
 if (b == 3) then
  {
   pon(tope,240,COPY_PUT);          // Pone ventana gráfica 240
   texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
  }
}
```

```
//   Busca numero en la lista enlazada GRUPO.
int busca_salon_arreglo(struct MATERIA *p, struct ARREGLO *arreglo, long int numero, int j)
{
 struct GRUPO *g;
 struct MATERIA *r, *s;
 int a,k;

 r = p;
 for (k=0, a=0; k <= (j - 1) && a != 1; k++)
  {
    s = busca_materia(r,arreglo[k].materia);
    g = (*s).grupos;
    while ((*g).num != arreglo[k].grupo) do
            g = (*g).apun;
    if ((*g).numsalon == numero) then
            a = 1;
  }
 return(a);
}

//   Construye el menú inferior y etiquetas, para datos de salones.
void cuadros_salones_todos(int tipo)
{
  switch(tipo)
    {
          case 332: texto("SALONES DE DIBUJO",338,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
          case 333: texto("SALONES DE LABORATORIO",338,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
          case 334: texto("SALONES CHICOS (MAX. 30 ALUMNOS)",338,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
          case 335: texto("SALONES GRANDES (MAX. 60 ALUMNOS)",338,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
          case 336: texto("S A L O N E S",338,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                    break;
    }
  texto("PAG. 1",240,BLUE,SANS_SERIF_FONT,1,2,1,2);
  cuadro(150,BLUE);
  texto("Seleccione un salón para obtener más información",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("<TAB> cambia menú",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  texto("PgUp/PgDnr",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
}

//   Busca numero en la lista enlazada GRUPO.
void busca_salon_tipo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).tipo == clase) then
            {
             if ((*s).numsalon == numero) then
              {
               if (j > 0) then
                    a = busca_salon_arreglo(g,arreglo,numero,j);
```

```
                if (a == 0) then
                 {
                        arreglo[].grupo = (*s).num;
                        arreglo[].materia = (*r).num;
                        arreglo[].numero = (*s).numsalon;
                        j = j + 1;
                 }
                }
               }
      s = (*s).apun;
     }
  r = (*r).materias;
 }
 arreglo[].grupo = -2;
 arreglo[].materia = -2;
 arreglo[].numero = -2;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_tipo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int k,n,j,dato,a;
 char num[100],salon[100];

 r = p;
 g = p;
 a = 0;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
   while (s != NULL) do
    {
     if ((*s).tipo == clase) then
           {
           dato = 1;
           itoa((*s).numsalon,salon,10);
           for (n=0; n < k && dato == 1; n++)
            {
             if (num[n] == salon[n]) then
               dato =1;
             else
               dato = 0;
            }
           if (dato == 1) then
            {
             if (j > 0) then
                  a = busca_salon_arreglo(g,arreglo,(*s).numsalon,j);
             if (a == 0) then
             {
                    arreglo[].grupo = (*s).num;
                    arreglo[].materia = (*r).num;
                    arreglo[].numero = (*s).numsalon;
                    j = j + 1;
             }
            }
           }
     s = (*s).apun;
    }
  r = (*r).materias;
 }
```

```
  arreglo[].grupo = -2;
  arreglo[].materia = -2;
  arreglo[].numero = -2;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_todo_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if ((*s).tipo == clase) then
            {
              if (j > 0) then
                 a = busca_salon_arreglo(g,arreglo,(*s).numsalon,j);
              if (a == 0) then
                {
                      arreglo[].grupo = (*s).num;
                      arreglo[].materia = (*r).num;
                      arreglo[].numero = (*s).numsalon;
                      j = j + 1;
                }
            }
       s = (*s).apun;
      }
   r = (*r).materias;
  }
  arreglo[].grupo = -2;
  arreglo[].materia = -2;
  arreglo[].numero = -2;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j,a;

 r = p;
 a = 0;
 j = 0;
 s = (*r).grupos;
     while (s != NULL) do
      {
       if ((*s).tipo == clase) then
            {
              if (j > 0) then
                 a = busca_salon_arreglo(r,arreglo,(*s).numsalon,j);
              if (a == 0) then
                {
                      arreglo[].grupo = (*s).num;
                      arreglo[].materia = (*r).num;
                      arreglo[].numero = (*s).numsalon;
                      j = j + 1;
```

```
            }
          }
      s = (*s).apun;
    }
  arreglo[].grupo = -2;
  arreglo[].materia = -2;
  arreglo[].numero = -2;
}

//   Busca numero en la lista enlazada GRUPO.
void busca_salon(struct MATERIA *p, long int numero, struct ARREGLO *arreglo)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            if ((*s).numsalon == numero) then
             {
              if (j > 0) then
                  a = busca_salon_arreglo(g,arreglo,numero,j);
              if (a == 0) then
               {
                    arreglo[j].grupo = (*s).num;
                    arreglo[j].materia = (*r).num;
                    arreglo[j].numero = (*s).numsalon;
                    j = j + 1;
               }
             }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  arreglo[].grupo = -2;
  arreglo[].materia = -2;
  arreglo[].numero = -2;
}

//   Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_todo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int k,n,j,dato,a;
 char num[100],salon[100];

 r = p;
 g = p;
 a = 0;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
```

```
            dato = 1;
            itoa((*s).numsalon,salon,10);
            for (n=0; n < k && dato == 1; n++)
              {
              if (num[n] == salon[n]) then
                dato =1;
              else
                dato = 0;
              }
            if (dato == 1) then
              {
              if (j > 0) then
                    a = busca_salon_arreglo(g,arreglo,(*s).numsalon,j);
              if (a == 0) then
               {
                    arreglo[j].grupo = (*s).num;
                    arreglo[j].materia = (*r).num;
                    arreglo[j].numero = (*s).numsalon;
                    j = j + 1;
               }
              }
      s = (*s).sgun;
      }
   r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}

void busca_salon_todo(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
          if (j > 0) then
            a = busca_salon_arreglo(g,arreglo,(*s).numsalon,j);
          if (a == 0) then
            {
              arreglo[j].grupo = (*s).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*s).numsalon;
              j = j + 1;
            }
      s = (*s).sgun;
     }
   r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
}
```

```
void busca_salon_materia_todo(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j,a;

 r = p;
 a = 0;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
          if (j > 0) then
             a = busca_salon_arreglo(r,arreglo,(*s).numsalon,j);
          if (a == 0) then
            {
              arreglo[j].grupo = (*s).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*s).numsalon;
              j = j + 1;
            }
      s = (*s).apun;
     }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
 arreglo[j].numero = -2;
}

// Cuenta el numero de elementos del arreglo ARREGLO
void busca_salon_tipo_cuenta(struct MATERIA *p, long int numero, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipo == clase) then
             {
              if ((*s).numsalon == numero) then
                        j = j + 1;
             }
       s = (*s).apun;
      }
   r = (*r).materias;
  }
  *cuenta = j;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_tipo_cuenta(struct MATERIA *p, long int numero, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],salon[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
```

```
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipo == clase) then
             {
              dato = 1;
              itoa((*s).numsalon,salon,10);
              for (n=0; n < k && dato == 1; n++)
                {
                 if (num[n] == salon[n]) then
                   dato =1;
                 else
                   dato = 0;
                }
              if (dato == 1) then
                        j = j + 1;
             }
      s = (*s).apun;
      }
   r = (*r).materias;
  }
  *cuenta = j;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_todo_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipo == clase) then
                        j = j + 1;
      s = (*s).apun;
      }
   r = (*r).materias;
  }
  *cuenta = j;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipo == clase) then
                        j = j + 1;
      s = (*s).apun;
```

```
        }
      *cuenta = j;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cuenta(struct MATERIA *p, long int numero, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
     {
            if ((*s).numsalon == numero) then
                    j = j + 1;
      s = (*s).apun;
     }
  r = (*r).materias;
 }
 *cuenta = j;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_todo_cuenta(struct MATERIA *p, long int numero, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],salon[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            itoa((*s).numsalon,salon,10);
            for (n=0; n < k && dato == 1; n++)
              {
               if (num[n] == salon[n]) then
                 dato =1;
               else
                 dato = 0;
              }
            if (dato == 1) then
                    j = j + 1;
      s = (*s).apun;
     }
  r = (*r).materias;
 }
 *cuenta = j;
}

void busca_salon_todo_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
```

```
struct GRUPO *s;
int j;

r = p;
j = 0;
while (r != NULL) do
 {
  s = (*r).grupos;

   while (s != NULL) do
    {
         j = j + 1;
         s = (*s).apun;
    }
  r = (*r).materias;
 }
 *cuenta = j;
}

void busca_salon_materia_todo_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
    {
         j = j + 1;
         s = (*s).apun;
    }
 *cuenta = j;
}

//   Busca numero en la lista enlazada GRUPO.
void busca_salon_tipo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
    {
     if ((*s).tipo == clase) then
          {
           if ((*s).numsalon == numero) then
           {
                salida[j].grupo = (*s).num;
                salida[j].materia = (*r).num;
                salida[j].numero = (*s).numgrupo;
                j = j + 1;
           }
          }
    s = (*s).apun;
    }
 r = (*r).materias;
}
salida[j].grupo = -2;
```

```
  salida[j].materia = -2;
  salida[j].numero = -2;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_tipo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],salon[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).tipo == clase) then
            {
             dato = 1;
             itoa((*s).numsalon,salon,10);
             for (n=0; n < k && dato == 1; n++)
              {
               if (num[n] == salon[n]) then
                 dato =1;
               else
                 dato = 0;
              }
             if (dato == 1) then
               {
                        salida[j].grupo = (*s).num;
                        salida[j].materia = (*r).num;
                        salida[j].numero = (*s).numgrupo;
                        j = j + 1;
               }
            }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_todo_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).tipo == clase) then
            {
```

```
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    j = j + 1;
            }
        s = (*s).apun;
       }
     r = (*r).materias;
    }
    salida[j].grupo = -2;
    salida[j].materia = -2;
    salida[j].numero = -2;
}

//    Busca clase en la lista enlazada GRUPO.
void busca_salon_materia_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
    {
     if ((*s).tipo == clase) then
          {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    j = j + 1;
          }
     s = (*s).apun;
    }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
    {
          if ((*s).numsalon == numero) then
           {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    j = j + 1;
           }
     s = (*s).apun;
    }
  r = (*r).materias;
 }
```

```
  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
}

//    Busca numero en la lista enlazada GRUPO.
void busca_salon_cadena_todo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char num[100],salon[100];

 r = p;
 itoa(numero,num,10);
 k = strlen(num);
 j = 0;
 while (r != NULL) do
  {
  s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            itoa((*s).numsalon,salon,10);
            for (n=0; n < k && dato == 1; n++)
             {
              if (num[n] == salon[n]) then
               dato =1;
              else
               dato = 0;
             }
            if (dato == 1) then
             {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    j = j + 1;
             }
     s = (*s).apun;
     }
  r = (*r).materias;
  }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
}

void busca_salon_todo_imprime(struct MATERIA *p, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
     {
              salida[j].grupo = (*s).num;
              salida[j].materia = (*r).num;
              salida[j].numero = (*s).numgrupo;
              j = j + 1;
     s = (*s).apun;
```

```
    }
  r = (*r).materias;
 }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
}

void busca_salon_materia_todo_imprime(struct MATERIA *p, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
              salida[j].grupo = (*s).num;
              salida[j].materia = (*r).num;
              salida[j].numero = (*s).numgrupo;
              j = j + 1;
     s = (*s).apun;
     }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
}

//  Despliega en pantalla los datos de salones de la lista enlazada GRUPO. Regresa la última ventana colocada.
int ventanas_salones_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo, int *max, int tipo)
{
 struct MATERIA *s, *r;
 struct GRUPO *g;
 char numero[100],cadena[100];
 int i,j,k,c;

 i = 339;
 j = *max;
 r = p;
 for (k=inicio; k <= fin; k++)
 {
  s = busca_materia(r,arreglo[k].materia);
  g = (*s).grupos;
  while ((*g).num != arreglo[k].grupo) do
            g = (*g).apun;
  if (g != NULL) then
   {
    cuadro(j,CYAN);
    pon(tope,i,COPY_PUT);          // Pone ventana gráfica para i
    itoa((*g).numsalon,numero,10);
    if (tipo == 336) then
     {
         strcpy(cadena,numero);
         c = (*g).tipo;
         switch(c)
          {
           case 1: strcat(cadena,"  Salón de dibujo            ");
                   break;
           case 2: strcat(cadena,"  Salón de laboratorio       ");
                   break;
           case 3: strcat(cadena,"  Salón chico(máx. 30 alumnos) ");
                   break;
           case 4: strcat(cadena,"  Salón grande(máx. 60 alumnos)");
```

```
                    break;
            case 5: strcat(cadena,"  Salón sin clasificar      ");
                    break;
           }
          texto(cadena,i,RED,SMALL_FONT,3,2,2,1);
     }
    if (tipo == 333) then
          texto(numero,i,RED,SMALL_FONT,3,2,2,1);
    if ((tipo == 332) || (tipo == 334) || (tipo == 335)) then
          {
           strcpy(cadena,"A - ");
           strcat(cadena,numero);
           texto(cadena,i,RED,SMALL_FONT,3,2,2,1);
          }
    i = i + 1;
    *max = j;
    j = j + 1;
   }
 }
 return(l);
}

// Muestra en pantalla los datos de salones en la lista enlazada GRUPO. Regresa la posición del nodo seleccionado.
int salones_todos(struct MATERIA *p, long int numero, struct ARREGLO * arreglo, int b, int tipo, int cuenta)
{
 struct MATERIA *r;
 struct POS
          {
          int inicio;
          int fin;
          };
 struct POS *pos;
 int k, i,regresa,numc,c,ultimo,grupo,quot,rem,j,a,inicio,fin,dato,max;
 char sig[10],cad[10];

 // Reserva espacio en memoria para pos
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
  {
    printf("Error.");
    exit(1);
  }
 arreglo[0].grupo = 0;
 arreglo[0].materia = 0;
 arreglo[0].numero = 0;
 r = p;
 switch(tipo)
  {
        case 332: if (b == 0) then
                    busca_salon_tipo(r,numero,arreglo,1);
                  if (b == 1) then
                    busca_salon_cadena_tipo(r,numero,arreglo,1);
                  if (b == 2) then
                    busca_salon_todo_tipo(r,arreglo,1);
                  if (b == 3) then
                    busca_salon_materia_tipo(r,arreglo,1);
                  break;
        case 333: if (b == 0) then
                    busca_salon_tipo(r,numero,arreglo,2);
                  if (b == 1) then
                    busca_salon_cadena_tipo(r,numero,arreglo,2);
                  if (b == 2) then
                    busca_salon_todo_tipo(r,arreglo,2);
                  if (b == 3) then
                    busca_salon_materia_tipo(r,arreglo,2);
```

```
                    break;
          case 334: if (b == 0) then
                      busca_salon_tipo(r,numero,arreglo,3);
                    if (b == 1) then
                      busca_salon_cadena_tipo(r,numero,arreglo,3);
                    if (b == 2) then
                      busca_salon_todo_tipo(r,arreglo,3);
                    if (b == 3) then
                      busca_salon_materia_tipo(r,arreglo,3);
                    break;
          case 335: if (b == 0) then
                      busca_salon_tipo(r,numero,arreglo,4);
                    if (b == 1) then
                      busca_salon_cadena_tipo(r,numero,arreglo,4);
                    if (b == 2) then
                      busca_salon_todo_tipo(r,arreglo,4);
                    if (b == 3) then
                      busca_salon_materia_tipo(r,arreglo,4);
                    break;
          case 336: if (b == 0) then
                      busca_salon(r,numero,arreglo);
                    if (b == 1) then
                      busca_salon_cadena_todo(r,numero,arreglo);
                    if (b == 2) then
                      busca_salon_todo(r,arreglo);
                    if (b == 3) then
                      busca_salon_materia_todo(r,arreglo);
                    break;
  }
ultimo = 0;
for (k=0; arreglo[k].grupo != -2; k++)
     ultimo = ultimo + 1;
quot = (ultimo/3);
rem = (ultimo%3);
ultimo = ultimo - 1;
if (ultimo == -1) then
  ultimo = 0;
ordena_arreglo(arreglo,ultimo);          // Ordena arreglo antes de desplegar en pantalla
if (rem != 0) then
  grupo = quot + 1;
else
  grupo = quot;
 if (grupo == 0) then
  {
   llama_error(16);
   return(-2);
  }
 k = 0;
 for (j=1; j <= grupo; j++)
   {
     for (a=0; arreglo[k].grupo != -2 && a < 3; k++, a++)
          {
           fin = k;
           if (a == 0) then
             inicio = k;
          }
    pos[j].inicio = inicio;
    pos[j].fin = fin;
  }
cuadros_salones_todos(tipo);                    // Pone ventanas de etiquetas y menú inferior
j = 1;
max = 122;
inicio = pos[j].inicio;
fin = pos[j].fin;
numc = ventanas_salones_todos(p,inicio,fin,arreglo,&max,tipo);
```

```
i = 122;
c=END;
regresa=0;
pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1: llama_ayuda(54);                  // Llama al módulo AYUDA
                 break;
                                                   // Permite desplazar al cursor sobre las ventanas
        case ABAJO:  quita(tope);          // Quita ventana gráfica del cursor
                     i++;
                     if (i == (max + 1)) then
                           i = 122;
                     pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     break;

        case ARRIBA: quita(tope);          // Quita ventana gráfica del cursor
                     i--;
                     if (i == (122 - 1)) then
                           i = max;
                     pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     break;

        case TAB:    if ((i >= 122) && (i <= max)) then              // Cambia al menú inferior
                     {
                      regresa=i;
                      i=237;
                      i=mover_ventana_menu(i,i,237,regresa);
                      if (i == 237) then
                            {
                             c = ENTER;
                             dato = -2;
                            }
                     }
                     break;
                           // Cambia a la siguiente página
        case PGDN:   quita(tope);          // Quita ventana gráfica // cursor
                     numc--;
                     for (k=399; k <= numc; k++)
                           quita(tope);                // Quita ventanas gráficas restantes
                     for (k = 122; k <= max; k++)
                           {
                            setfillstyle(SOLID_FILL,GREEN);
                            bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                           }
                     j++;
                     if (j == (grupo+1)) then
                      j=1;
                     inicio=pos[j].inicio;
                     fin=pos[j].fin;
                     max=122;
                     numc = ventanas_salones_todos(p,inicio,fin,arreglo,&max,tipo);
                     strcpy(sig,"PAG. ");
                     itoa(j,cad,10);
                     strcat(sig,cad);
                     texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                     if (i > max) then
                     {
                      i = max;
                      pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
                     }
                     else
```

```
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
                                                // Cambia a la página anterior
          case PGUP:  quita(tope);          // Quita ventana gráfica cursor
                    numc--;
                    for (k=339; k <= numc; k++)
                        quita(tope);          // Quita ventanas gráficas restantes
                    for (k = 122; k <= max; k++)
                        {
                         setfillstyle(SOLID_FILL,GREEN);
                         bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                        }
                    j--;
                    if (j == 0) then
                      j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=122;
                    numc = ventanas_salones_todos(p,inicio,fin,arreglo,&max,tipo);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                    if (i > max) then
                     {
                      i = max;
                      pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                     }
                    else
                      pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
    }
  }
  quita(tope);          // Quita ventana gráfica del cursor
  numc--;
  for (k=339; k <= numc; k++)
    quita(tope);          // Quita ventanas gráficas restantes
  k = 338;
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  for (k = 122; k <= max; k++)
    {
     setfillstyle(SOLID_FILL,GREEN);
     bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
  texto("MATERIA SELECCIONADA",240,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
  if (dato != -2) then
    {
     dato = i - 122;
     a = pos[j].inicio;
     a = a + dato;
    }
  else
   a = dato;
  farfree(pos);                                     // Libera el espacio reservado a pos
  return(a);                                        // Regresa la posición del nodo
}


//  Despliega en pantalla la información completa de los datos de salones de la lista enlazada GRUPO.
int mover_datos_salones_todos(struct GRUPO *g, long int numsalon, int cuenta)

{ struct GRUPO *r,*s;
  struct POS{
            int inicio;
```

```
            int fin;
          };
struct POS *pos;
int c,inicio,fin,grupo,ultimo,j,quot,rem,numc,k,a,dato;
char sig[20], cad[20];

// Reserva espacio en memoria para pos
pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
if (!pos)
  {
   printf("Error.");
   exit(1);
  }
r=g;
s=g;
ultimo = 0;
while (r != NULL) do
{
  if (numsalon == (*r).numsalon) then
        ultimo=ultimo + 1;
  r=(*r).apun;
}
quot=(ultimo/3);
rem=(ultimo%3);

if (rem != 0) then
  grupo=quot + 1;
else
  grupo=quot;
fin=0;
for (j=1; j <= grupo; j++)
{
 inicio=fin+1;
 a = 0;
 while (( s != NULL) && (a < 3)) do
 {
  if (numsalon == (*s).numsalon) then
   {
        fin=(*s).num;
        a = a + 1;
   }
  s=(*s).apun;
 }
 pos[j].inicio=inicio;
 pos[j].fin=fin;
}
j=1;
inicio=pos[j].inicio;
fin=pos[j].fin;
numc=ventanas_ponen_datos_salones(g,inicio,fin,numsalon);
c=END;
while (c != ESC) do
{
 c=bioskey(0);
 switch(c)
 {
        case F1: llama_ayuda(56);              // Llama al módulo AYUDA
                 break;
                                               // Cambia a la página siguiente
        case PGDN:  numc--;
                    for (k=294; k <= numc; k++)
                        quita(tope);           // Quita ventanas gráficas restantes
                    j++;
                    if (j == (grupo+1)) then
                      j=1;
```

```
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_ponen_datos_salones(g,inicio,fin,numsalon);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,289,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;
                                                    // Cambia a la página anterior
        case PGUP:  numc--;
                    for(k=294; k <= numc; k++)
                        quita(tope);        // Quita ventanas gráficas restantes
                    j--;
                    if (j == 0) then
                      j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_ponen_datos_salones(g,inicio,fin,numsalon);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,289,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;

        case DERECHA:  dato = -3;          // Asigna -3 para desplegar los datos de salones
                       c = ESC;              // de la siguiente materia.
                       break;

        case IZQUIERDA: dato = -4;                  // Asigna -4 para desplegar los datos de salones
                       c = ESC;              // de la materia anterior.
                       break;

        case ESC :     dato = -2;                   // Salir.
                       break;
   }
  }
  numc--;
  for (k=294; k <= numc; k++)
    quita(tope);        // Quita ventanas gráficas
  farfree(pos);                                     // Libera el espacio reservado para pos
  return(dato);
}

//   Busca salon en la lista enlazada GRUPO.
int busca_salon_resultado(struct MATERIA *p, long int salon, int materia[50])
{
 struct MATERIA *r;
 struct GRUPO *g;
 int j,a;

 r = p;
 j = 0;
 while (r != NULL) do
  {
    g = (*r).grupos;
    a = 0;

    while ((g != NULL) && (a == 0)) do
      {
          if ((*g).numsalon == salon) then
              a = 1;
          g = (*g).apun;
      }
    if (a == 1) then
     {
```

```
      materia[j] = (*r).num;
      j = j + 1;
      }
     r = (*r).materias;
    }
  materia[j] = -2;
  return(j);
}

// Llama a las funciones que despliegan en pantalla los datos de salones para cada materia.
void consulta_datos_salones_todos(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10])
{
 struct MATERIA *r,*m,*s;
 struct GRUPO *g,*gpo;
 int k,c,b,materia[50],ultimo,n,e,num;
 char numero[100],cadena[100];

 r = p;
 m = p;
 if (y == 3) then
 quita(tope);          // Quita ventana gráfica 240
 for (k=236; k <= 240; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 while ((*r).num != arreglo[a].materia) do
          r = (*r).materias;
 g = (*r).grupos;
 while ((*g).num != arreglo[a].grupo) do
          g = (*g).apun;
 if (y == 3) then
   texto("Presione < ESC > para ir al menú . . . . ",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
 else
  {
   texto("Las flechas de posición (IZQ y DER) cambian materia",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
   texto("<ESC> para salir",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
   texto("PgUp/PgDn",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  }
 // Salón
 itoa((*g).numsalon,numero,10);
 if (((*g).tipo == 2) || ((*g).tipo == 5)) then
   texto(numero,255,LIGHTRED,SMALL_FONT,3,2,2,1);
  else
  {
   strcpy(cadena,"A - ");
   strcat(cadena,numero);
   texto(cadena,255,LIGHTRED,SMALL_FONT,3,2,2,1);
  }
 c=(*g).tipo;
 switch(c)
  {
          case 0: strcpy(cadena,"Salón sin clasificar");
                  strcat(cadena,"    Semestre ... ");
                  strcat(cadena,semestre);
                  texto(cadena,255,LIGHTRED,SMALL_FONT,1,1,2,1);
                  break;
          case 1: strcpy(cadena,"Salón de dibujo");
                  strcat(cadena,"    Semestre ... ");
                  strcat(cadena,semestre);
                  texto(cadena,255,LIGHTRED,SMALL_FONT,1,1,2,1);
                  break;
          case 2: strcpy(cadena,"Salón de laboratorio");
                  strcat(cadena,"    Semestre ... ");
                  strcat(cadena,semestre);
```

```
                texto(cadena,266,LIGHTRED,SMALL_FONT,1,1,2,1);
                break;
        case 3: strcpy(cadena,"Salón chico (máx. 30 alumnos)");
                strcat(cadena,"    Semestre ... ");
                strcat(cadena,semestre);
                texto(cadena,266,LIGHTRED,SMALL_FONT,1,1,2,1);
                break;
        case 4: strcpy(cadena,"Salón grande (máx. 60 alumnos)");
                strcat(cadena,"    Semestre ... ");
                strcat(cadena,semestre);
                texto(cadena,266,LIGHTRED,SMALL_FONT,1,1,2,1);
                break;
        case 5: strcpy(cadena,"Salón sin asignación");
                strcat(cadena,"    Semestre ... ");
                strcat(cadena,semestre);
                texto(cadena,266,LIGHTRED,SMALL_FONT,1,1,2,1);
                break;
 }
cuadro(266,CYAN);
// Materia
texto("MAT. 1",267,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
texto((*r).nombre,286,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2);
texto("Pag. 1",289,RED,SANS_SERIF_FONT,1,2,1,2);
// Etiquetas
texto("GRUPO",290,BLUE,SMALL_FONT,1,1,2,1);
texto("PROFESOR",291,BLUE,SMALL_FONT,1,1,2,1);
texto("DIAS",292,BLUE,SMALL_FONT,1,1,2,1);
texto("HORARIO",293,BLUE,SMALL_FONT,1,1,2,1);
gpo = (*r).grupos;
k = 0;
while (gpo != NULL) do
 {
  if ((*gpo).numsalon == (*g).numsalon) then
        k = k + 1;
  gpo = (*gpo).apun;
 }
                                        // Muestra la información completa de salones
if (y == 3) then
 {
  e = 1;
  while (e != -2) do
    e = mover_datos_salones_todos((*r).grupos,(*g).numsalon,k);
 }
else
 {
  b = busca_salon_resultado(m,(*g).numsalon,materia);
  for (n=0; materia[n] != -2; n++)
        ultimo = n;
  e = 1;
  s = p;
  b = 0;
  while ((*s).num != materia[b]) do
        s = (*s).materias;
  while (e != -2) do                    // Despliega en pantalla los datos de salones para la siguiente materia
       {
        e = mover_datos_salones_todos((*s).grupos,(*g).numsalon,k);
        if (e == -3) then
          {
          b++;
          if (b == (ultimo + 1)) then
                b = 0;
          num = b + 1;
          s = busca_materia(m,materia[b]);
          strcpy(cadena,"MAT. ");
          itoa(num,numero,10);
```

```
                strcat(cadena,numero);
                texto(cadena,287,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
                texto((*s).nombre,298,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2);
                e = 1;
               }
              if (e == -4) then              // Despliega en pantalla los datos de salones para la materia anterior
               {
               b--;
               if (b == -1) then
                     b = ultimo;
               num = b + 1;
               s = busca_materia(m,materia[b]);
               strcpy(cadena,"MAT.  ");
               itoa(num,numero,10);
               strcat(cadena,numero);
               texto(cadena,287,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
               texto((*s).nombre,298,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2);
               e = 1;
              }
            }
   }
  for (k=287; k <= 293; k++)
  {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 k = 296;
 setfillstyle(SOLID_FILL,GREEN);
 bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
 if (y == 3) then
  {
    k = 248;
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 for (k=295; k <= 298; k++)
  {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 ventana_menu_consulta();              // Pone el menú principal de Consulta de Salones
 if (y == 3) then
  {
   pon(tope,240,COPY_PUT);          // Pone ventana gráfica 240
   texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
  }
}
```

B.3.10. IMPRIME.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include <time.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\graficos.h"
#include "d:\archivos.h"
#include "d:\g_todos.h"
#include "d:\p_todos.h"
#include "d:\consulta.h"
#include "d:\errores.h"

#define  then


//    Verifica que la salida a impresora sea correcta.
int verificar(char texto[100])
{
 int valor, estado, k;

 valor = 0;
 for (k = 0; texto[k]; k++)
  {
   estado = bioprint(2,texto[k],0);
   if (estado & 0x08) then                  // Entrada/Salida
     valor = 1;
   if (estado & 0x20) then                  // Falta de papel
            valor = 1;
   if (estado & 0x40) then                  // Verifica impresora
     valor = 1;
  }
 if (valor == 1) then
  llama_error(54);
 return(valor);
}


//   Envía a impresora la información contenida en ARREGLO. Imprime listados por GRUPO.
void imprime_grupo(struct MATERIA *p, struct ARREGLO *arreglo, int condicion, char semestre[10])
{
struct MATERIA *r, *s;
struct GRUPO *g;
int k, ntimes, b, i,c,a, m,j,n, compara;
char *texto[80], numero[100];

for (i = 0; i <= 59; i++)
{                       //   Reserva espacio en memoria para texto
texto[i] = (char *) farcalloc(180,sizeof(char));
if (!texto[i]) then
  {
  outtextxy(100,100,"No hay espacio en memoria. Funcion imprime_estrecord");
  exit(1);
```

```
    }
  }
  r = p;
  s = busca_materia(r,arreglo[0].materia);
  g = (*s).grupos;
  while ((*g).num != arreglo[0].grupo) do
        g = (*g).apun;
  compara = (*g).numgrupo;
  nlineas = 0;
  b = 0;
  i = 0;
  while (arreglo[i].materia != -2) do
   {
    if (b == 0) then
      {
       nlineas = 0;
                                                        // imprime los encabezados
       strcpy(texto[0],"");
       for (k=0; k<=50; k++)
            strcat(texto[0],"");
       strcat(texto[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
       texto[1][0] = '\n';
       texto[1][1] = '\r';
       texto[1][2] = '\0';
       strcpy(texto[2],"");
       for (k=0; k<=44; k++)
            strcat(texto[2],"");
       strcat(texto[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
       texto[3][0] = '\n';
       texto[3][1] = '\r';
       texto[3][2] = '\0';

       strcpy(texto[4],"");
       for (k=0; k<=55; k++)
            strcat(texto[4],"");
       strcat(texto[4],"Ingenieria Mecanica Electrica");
       texto[5][0] = '\n';
       texto[5][1] = '\n';
       texto[5][2] = '\r';
       texto[5][3] = '\0';

       strcpy(texto[6],"");
       for (k=0; k<=62; k++)
            strcat(texto[6],"");
       strcat(texto[6],"HORARIOS POR GRUPO");
       texto[7][0] = '\n';
       texto[7][1] = '\n';
       texto[7][2] = '\r';
       texto[7][3] = '\0';

       strcpy(texto[8],"");
       for (k=0; k<=44; k++)
            strcat(texto[8],"");
       strcat(texto[8],"PERIODO:  ");
       strcat(texto[8],semestre);
       texto[9][0] = '\n';
       texto[9][1] = '\r';
       texto[9][2] = '\0';

strcpy(texto[10],"┌──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┐");
└──────────┴──────────┘");

strcat(texto[10],"┌──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┐");
└──────────┴──────────┘");
```

```
      texto[11][0] = '\n';
      texto[11][1] = '\r';
      texto[11][2] = '\0';

      strcpy(texto[12]," | CMAT |     NOMBRE DE LA MATERIA      |        NOMBRE DEL PROFESOR        ");
      strcat(texto[12]," |      RFC.     | GPO. |  LUN  |  MAR  |  MIE  |  JUE  |  VIE  |  SAB  | SALON | CUPO | ");
      texto[13][0] = '\n';
      texto[13][1] = '\r';
      texto[13][2] = '\0';

strcpy(texto[14]," └──────┴────────────────────────────────┴──────────────
──────────────────────┘");

strcat(texto[14],"─────────────────┴───────┴───────┴───────┴───────┴───────┴───────┴─────
───┴───────┴──────┘ ");
      texto[15][0] = '\r';
      texto[15][1] = '\0';
      j = 15;
      b = 1;
    }
   s = busca_materia(r,arreglo[i].materia);
   g = (*s).grupos;
   while ((*g).num != arreglo[i].grupo) do
            g = (*g).apun;
   if (compara != (*g).numgrupo) then              // Sólo imprime 14 líneas por hoja
    {
          nlinea = 14;
          compara = (*g).numgrupo;
    }
   if (nlinea > 13) then
    {
          for (k=0; k <= j; k++)
           {
            a = 1;
            while (a == 1) do
              a = verificar(texto[k]);
            for (n=0; texto[k][n]; n++)
              biosprint(0,texto[k][n],0);
           }
      biosprint(0,'\f',0);                          // Salto de página
      b = 0;
    }
   else                                             // Imprime información completa
    {
          j = j+1;

strcpy(texto[j]," ├──────┼────────────────────────────────┼──────────────
──────────────────────┤");

strcat(texto[j],"─────────────────┼───────┼───────┼───────┼───────┼───────┼───────┼─────
───┼───────┼──────┤ ");
      j = j + 1;
      texto[j][0] = '\n';
      texto[j][1] = '\r';
      texto[j][2] = '\0';
          // Clave de la materia
          j = j + 1;
          strcpy(texto[j]," | ");
          itoa((*s).clave,numero,10);
          for (k=0; numero[k]; k++);
            while (k < 4) do
             {
              strcat(texto[j],"0");
              k = k+1;
             }
```

```
        strcat(texto[],numero);
        strcat(texto[],' | ');

        // Nombre de la materia
        strcat(texto[],(*e).nombre);
        for (k=0; (*e).nombre[k]; k++);
          while (k < 28) do
            {
              strcat(texto[],' ');
              k = k+1;
            }
        strcat(texto[],' | ');

        //Nombre del profesor
        strcat(texto[],(*g).nombreprofe);
        for (k=0; (*g).nombreprofe[k]; k++);
          while (k < 32) do
            {
              strcat(texto[],' ');
              k = k+1;
            }
        strcat(texto[],' | ');

// RFC del profesor
if (condicion == 254) then
        {
        strcat(texto[],(*g).rfc);
        for (k=0; (*g).rfc[k]; k++);
          while (k < 12) do
            {
              strcat(texto[],' ');
              k = k+1;
            }
        }
else
        strcat(texto[],'            ');
strcat(texto[],' | ');

// Número de grupo
itoa((*g).numgrupo,numero,10);
for (k=0; numero[k]; k++);
        while (k < 4) do
          {
            strcat(texto[],'0');
            k = k+1;
          }
strcat(texto[],numero);
strcat(texto[],' | ');

// Días
c = 0;
m = 1;
k = 0;
while ((*g).dias[k] != 10) do
        {
        if ((*g).dias[c] == m) then
          {
            strcat(texto[],(*g).hora1);
            strcat(texto[],' | ');
            c = c + 1;
            k = k + 1;
          }
        else
          strcat(texto[],'      | ');
        m = m + 1;
```

```
        }
        while (m < 7) do
        {
         strcat(texto[],"      | ");
         m = m+1;
        }

// Salón
itoa((*g).numsalon,numero,10);
for (k=0; numero[k]; k++);
        while (k < 5) do
        {
          strcat(texto[]," ");
          k = k+1;
        }
strcat(texto[],numero);
strcat(texto[]," | ");

// Cupo, tipo de salón
c = (*g).tipo;
switch(c)
        {
         case 0: strcat(texto[],"    ");
                 break;
         case 1: strcat(texto[],"    ");
                 break;
         case 2: strcat(texto[],"    ");
                 break;
         case 3: strcat(texto[]," 30 ");
                 break;
         case 4: strcat(texto[]," 60 ");
                 break;
         case 5: strcat(texto[],"    ");
                 break;
        }
strcat(texto[]," | ");

j = j + 1;
texto[][0] = '\n';
texto[][1] = 'r';
texto[][2] = '\0';

j = j + 1;
// Poner siguiente línea
strcpy(texto[]," |      |                          | ");
strcat(texto[],"                        |            |     | ");

m = 1;
k = 0;
c = 0;
while ((*g).clase[k] != 10)
       {
        if ((*g).clase[c] == m) then
          {
           strcat(texto[],(*g).hora2);
           strcat(texto[]," | ");
           k = k + 1;
           c = c + 1;
          }
        else
         strcat(texto[],"      | ");
        m = m + 1;
       }
       while (m < 7) do
       {
```

```
            strcat(texto[j],"      | ");
            m = m+1;
          }
      strcat(texto[j],"    |      | ");
      j = j + 1;
      texto[j][0] = '\n';
      texto[j][1] = '\r';
      texto[j][2] = '\0';
      j = j + 1;

strcpy(texto[j]," └──────┴──────────────────────────────────────┴──────────────────────────────────
──────────────────────────────┘");

strcat(texto[j],"──────────────────────────────┴──────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────
──┴──────────┴──────────┴ ");
      j = j + 1;
      texto[j][0] = '\r';
      texto[j][1] = '\0';

      nlinea = nlinea + 1;
      i = i +1;
     }
   }
  for (k=0; k <= j; k++)                        // Manda el arreglo a impresora
  {
   a = 1;
   while (a == 1) do
     a = verificar(texto[k]);
   for (n=0; texto[k][n]; n++)
     biosprint(0,texto[k][n],0);
  }
 biosprint(0,'\f',0);                              // Salto de página
for (i = 0; i <= 59; i++)                          // Libera el espacio reservado para texto
   farfree(texto[i]);
}


// Envía a impresora la información contenida en ARREGLO. Imprime listados por MATERIA.
void imprime_materias(struct MATERIA *p, struct ARREGLO *arreglo, int condicion, char semestre[10])
{
 struct MATERIA *r, *s;
 struct GRUPO *g;
 int k, nlinea, b, i,c,a, m,j,n, compara;
 char *texto[60], numero[100];
                                                            // Reserva espacio en memoria para texto
 for (i = 0; i <= 59; i++)
 {
 texto[i] = (char *) farcalloc(180,sizeof(char));
 if (!texto[i]) then
   {
   outtextxy(100,100,"No hay espacio en memoria. Funcion imprime_extraord");
   exit(1);
   }
}

r = p;
s = busca_materia(r,arreglo[0].materia);
g = (*s).grupos;
while ((*g).num != arreglo[0].grupo) do
            g = (*g).apun;

compara = (*g).numgrupo;
nlinea = 0;
b = 0;
i = 0;
```

```
while (arreglo[i].materia != -2) do
  {
   if (b == 0) then
     {
      nlinea = 0;
                                                          // imprime encabezados
      strcpy(texto[0]," ");
      for (k=0; k<=50; k++)
            strcat(texto[0]," ");
      strcat(texto[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
      texto[1][0] = '\n';
      texto[1][1] = '\r';
      texto[1][2] = '\0';

      strcpy(texto[2]," ");
      for (k=0; k<=44; k++)
            strcat(texto[2]," ");
      strcat(texto[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
      texto[3][0] = '\n';
      texto[3][1] = '\r';
      texto[3][2] = '\0';

      strcpy(texto[4]," ");
      for (k=0; k<=55; k++)
            strcat(texto[4]," ");
      strcat(texto[4],"Ingenieria Mecanica Electrica");
      texto[5][0] = '\n';
      texto[5][1] = '\n';
      texto[5][2] = '\r';
      texto[5][3] = '\0';

      strcpy(texto[6]," ");
      for (k=0; k<=62; k++)
            strcat(texto[6]," ");
      strcat(texto[6],"HORARIOS POR MATERIA");
      texto[7][0] = '\n';
      texto[7][1] = '\n';
      texto[7][2] = '\r';
      texto[7][3] = '\0';

      strcpy(texto[8]," ");
      for (k=0; k<=44; k++)
            strcat(texto[8]," ");
      strcat(texto[8],"PERIODO:  ");
      strcat(texto[8],semestre);
      texto[9][0] = '\n';
      texto[9][1] = '\r';
      texto[9][2] = '\0';

strcpy(texto[10]," ┌──────────┬──────────────────────────────────────────────────────┬──────────────────────────
──────────────────────────┐");

strcat(texto[10],"────────────────────────────┬───────┬───────┬───────┬───────┬───────┬───────┬───────
────┬───────┬──────────┐ ");
      texto[11][0] = '\n';
      texto[11][1] = '\r';
      texto[11][2] = '\0';

      strcpy(texto[12]," | CMAT |     NOMBRE DE LA MATERIA     |       NOMBRE DEL PROFESOR        ");
      strcat(texto[12]," |      RFC.     | GPO. | LUN | MAR | MIE | JUE | VIE | SAB | SALON | CUPO | ");
      texto[13][0] = '\n';
      texto[13][1] = '\r';
      texto[13][2] = '\0';

strcpy(texto[14]," └──────────┴──────────────────────────────────────────────────────┴──────────────────────────
```

```
                                                    ");
strcat(texto[14],"
                                                    ");
      texto[15][0] = '\r';
      texto[15][1] = '\0';
      j = 15;
      b = 1;
     }
   s = busca_materia(r,arreglo[i].materia);
   g = (*s).grupos;
   while ((*g).num != arreglo[i].grupo) do
            g = (*g).apun;
   if (compara != (*g).numgrupo) then              // Imprime 14 líneas por hoja
    {
         nlineas = 14;
         compara = (*g).numgrupo;
    }
   if (nlineas > 13) then
    {
         for (k=0; k <= j; k++)
         {
           a = 1;
           while (a == 1) do
             a = verificar(texto[k]);
          for (n=0; texto[k][n]; n++)
           biosprint(0,texto[k][n],0);
         }
    biosprint(0,'\r',0);
    b = 0;
   }
   else                                            // Imprime la información completa
    {
        j = j+1;

strcpy(texto[]," |
                                                    ");

strcat(texto[]."
                                                    ");
    j = j + 1;
    texto[][0] = '\n';
    texto[][1] = '\r';
    texto[][2] = '\0';
        // Clave de la materia
        j = j + 1;
        strcpy(texto[]," | ");
        itoa((*s).clave,numero,10);
        for (k=0; numero[k]; k++);
          while (k < 4) do
            {
              strcat(texto[],"0");
              k = k+1;
            }
        strcat(texto[],numero);
        strcat(texto[]," | ");

        // Nombre de la materia
        strcat(texto[],(*s).nombre);
        for (k=0; (*s).nombre[k]; k++);

          while (k < 28) do
            {
              strcat(texto[]," ");
              k = k+1;
```

```
        }
      strcat(texto[],"  | ");

      //Nombre del profesor
      strcat(texto[],(*g).nombreprofe);
      for (k=0; (*g).nombreprofe[k]; k++);
        while (k < 32) do
          {
            strcat(texto[]," ");
            k = k+1;
          }
      strcat(texto[],"  | ");

// RFC del profesor
if (condicion == 254) then
      {
       strcat(texto[],(*g).rfc);
       for (k=0; (*g).rfc[k]; k++);
         while (k < 12) do
           {
             strcat(texto[]," ");
             k = k+1;
           }
      }
else
      strcat(texto[],"            ");
strcat(texto[],"  | ");

// Número de grupo
itoa((*g).numgrupo,numero,10);
for (k=0; numero[k]; k++);
      while (k < 4) do
       {
         strcat(texto[],"0");
         k = k+1;
       }
strcat(texto[],numero);
strcat(texto[],"  | ");

// Días
c = 0;
m = 1;
k = 0;
while ((*g).dias[k] != 10) do
      {
       if ((*g).dias[c] == m) then
         {
          strcat(texto[],(*g).hora1);
          strcat(texto[],"  | ");
          c = c + 1;
          k = k + 1;
         }
       else
        strcat(texto[],"      | ");
       m = m + 1;
      }
      while (m < 7) do
      {
       strcat(texto[],"      | ");
       m = m+1;
      }

// Salón
itoa((*g).numsalon,numero,10);
for (k=0; numero[k]; k++);
```

```
        while (k < 5) do
          {
           strcat(texto[],"  ");
           k = k+1;
          }
 strcat(texto[],numero);
 strcat(texto[]," | ");

 // Cupo, tipo de salón
 c = (*g).tipo;
 switch(c)
        {
         case 0: strcat(texto[],"    ");
                 break;
         case 1: strcat(texto[],"    ");
                 break;
         case 2: strcat(texto[],"    ");
                 break;
         case 3: strcat(texto[],"30 ");
                 break;
         case 4: strcat(texto[],"60 ");
                 break;
         case 5: strcat(texto[],"    ");
                 break;
        }
 strcat(texto[]," | ");

 i = i + 1;
 texto[i][0] = '\n';
 texto[i][1] = 'r';
 texto[i][2] = '\0';

 i = i + 1;
 // Poner siguiente linea
 strcpy(texto[]," |      |                          | ");
 strcat(texto[],"                                |          |     | ");

 m = 1;
 k = 0;
 c = 0;
 while ((*g).dias[k] != 10)
       {
        if ((*g).dias[c] == m) then
          {
           strcat(texto[],(*g).hora2);
           strcat(texto[]," | ");
           k = k + 1;
           c = c + 1;
          }
        else
         strcat(texto[],"      | ");
        m = m + 1;
       }
       while (m < 7) do
        {
         strcat(texto[],"      | ");
         m = m+1;
        }
 strcat(texto[],"      |     | ");
 i = i + 1;
 texto[i][0] = '\n';
 texto[i][1] = 'r';
 texto[i][2] = '\0';
 i = i + 1;
```

```
strcpy(texto[j],"
                                                                    ");
strcat(texto[j],"
                                                                    ");
      j = j + 1;
      texto[j][0] = '\r';
      texto[j][1] = '\0';

      nlinea = nlinea + 1;
      i = i +1;
     }
   }
  for (k=0; k <= j; k++)                          // Manda el arreglo impresora
   {
    a = 1;
    while (a == 1) do
      a = verificar(texto[k]);
    for (n=0; texto[k][n]; n++)
      biosprint(0,texto[k][n],0);
   }
 biosprint(0,'\f',0);                             // Salto de página
for (i = 0; i <= 59; i++)                         // Libera el espacio reservado para texto
   farfree(texto[i]);
}


// Envía a impresora la información contenida en ARREGLO. Imprime listados por SALON.
void imprime_salon(struct MATERIA *p, struct SALIDA *salida, int condicion, char semestre[10])
{
 struct MATERIA *r, *s;
 struct GRUPO *g;
 int k, nlinea, b, i, c, a, m, j, n, cuenta;
 long int comsalon;
 char *texto[60], numero[100];
                                                  // Reserva espacio en memoria para texto
 for (i = 0; i <= 59; i++)
 {
 texto[i] = (char *) farcalloc(180,sizeof(char));
 if (!texto[i]) then
   {
   outtextxy(100,100,"No hay espacio en memoria. Funcion imprime_edraord");
   exit(1);
   }
}
cuenta = 0;
for (k=0; salida[k].grupo <= -2; k++)
    cuenta = cuenta + 1;
cuenta = cuenta - 1;
ordena_salida(salida,cuenta);                     // Ordena el archivo de salida
r = p;
s = busca_materia(r,salida[0].materia);
g = (*s).grupos;
while ((*g).num != salida[0].grupo) do
            g = (*g).apun;
comsalon = (*g).numsalon;
nlinea = 0;
b = 0;
i = 0;
while (salida[i].materia != -2) do
 {
  if (b == 0) then
    {
     nlinea = 0;
                                                  // Imprime encabezados
```

```
        strcpy(tasdo[0]," ");
        for (k=0; k<=50; k++)
              strcat(tasdo[0]," ");
        strcat(tasdo[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
        tasdo[1][0] = '\n';
        tasdo[1][1] = '\r';
        tasdo[1][2] = '\0';

        strcpy(tasdo[2]," ");
        for (k=0; k<=44; k++)
              strcat(tasdo[2]," ");
        strcat(tasdo[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
        tasdo[3][0] = '\n';
        tasdo[3][1] = '\r';
        tasdo[3][2] = '\0';

        strcpy(tasdo[4]," ");
        for (k=0; k<=55; k++)
              strcat(tasdo[4]," ");
        strcat(tasdo[4],"Ingeniería Mecanica Electrica");
        tasdo[5][0] = '\n';
        tasdo[5][1] = '\n';
        tasdo[5][2] = '\r';
        tasdo[5][3] = '\0';

        strcpy(tasdo[6]," ");
        for (k=0; k<=62; k++)
              strcat(tasdo[6]," ");
        strcat(tasdo[6],"HORARIOS POR SALON");
        tasdo[7][0] = '\n';
        tasdo[7][1] = '\n';
        tasdo[7][2] = '\r';
        tasdo[7][3] = '\0';

        strcpy(tasdo[8]," ");
        for (k=0; k<=44; k++)
              strcat(tasdo[8]," ");
        strcat(tasdo[8],"PERIODO:  ");
        strcat(tasdo[8],semestre);
        tasdo[9][0] = '\n';
        tasdo[9][1] = '\r';
        tasdo[9][2] = '\0';

strcpy(tasdo[10],"┌──────┬──────────────────────────────┬──────────────────────────────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
──────────────────────────┐");

strcat(tasdo[10],"
└──────┴──────────────────────────────┴──────────────────────────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘ ");
        tasdo[11][0] = '\n';
        tasdo[11][1] = '\r';
        tasdo[11][2] = '\0';

        strcpy(tasdo[12]," | CMAT |     NOMBRE DE LA MATERIA     |      NOMBRE DEL PROFESOR      ");
        strcat(tasdo[12],"|    RFC.    | GPO. | LUN  | MAR  | MIE  | JUE  | VIE  | SAB  | SALON | CUPO | ");
        tasdo[13][0] = '\n';
        tasdo[13][1] = '\r';
        tasdo[13][2] = '\0';

strcpy(tasdo[14]," └──────┴──────────────────────────────┴──────────────────────────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘
──────────────────────────┘");

strcat(tasdo[14],"
└──────┴──────────────────────────────┴──────────────────────────────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┘ ");
        tasdo[15][0] = '\r';
        tasdo[15][1] = '\0';
```

```
      j = 15;
      b = 1;
     }
   s = busca_materia(r,salida[i].materia);
   g = (*s).grupos;
   while ((*g).num != salida[i].grupo) do
              g = (*g).apun;
   if (comsesion != (*g).numsesion) then                          // imprime 14 líneas por hoja
     {
           nlinea = 14;
           comsesion = (*g).numsesion;
     }
   if (nlinea > 13) then
     {
           for (k=0; k <= j; k++)
             {
              a = 1;
              while (a == 1) do
                 a = verificar(texto[k]);
             for (n=0; texto[k][n]; n++)
               biosprint(0,texto[k][n],0);
             }
      biosprint(0,'\f',0);
      b = 0;
     }
   else
     {                                                            // imprime la información completa
           j = j+1;

strcpy(texto[j]," ├──────┼─────────────────────────────────────┼──────────────
────────────────────────┼");

strcat(texto[j],"───────────────────┼───────┼───────┼───────┼───────┼───────┼───────┼──────
──┼───────┼──────┤ ");
      j = j + 1;
      texto[j][0] = '\n';
      texto[j][1] = '\r';
      texto[j][2] = '\0';

          // Clave de la materia
          j = j + 1;
          strcpy(texto[j]," | ");
          itoa((*s).clave,numero,10);
          for (k=0; numero[k]; k++);
            while (k < 4) do
              {
               strcat(texto[j],"0");
               k = k+1;
              }
          strcat(texto[j],numero);
          strcat(texto[j]," | ");

          // Nombre de la materia
          strcat(texto[j],(*s).nombre);
          for (k=0; (*s).nombre[k]; k++);
            while (k < 28) do
              {
               strcat(texto[j]," ");
               k = k+1;
              }
          strcat(texto[j]," | ");

          //Nombre del profesor
          strcat(texto[j],(*g).nombreprofe);
          for (k=0; (*g).nombreprofe[k]; k++);
```

```
        while (k < 32) do
          {
           strcat(texto[],' ');
           k = k+1;
          }
       strcat(texto[],' | ');

// RFC del profesor
if (condicion == 254) then
       {
        strcat(texto[],(*g).rfc);
        for (k=0; (*g).rfc[k]; k++);
          while (k < 12) do
           {
            strcat(texto[],' ');
            k = k+1;
           }
       }
else
       strcat(texto[],'            ');
strcat(texto[],' | ');

// Número de grupo
itoa((*g).numgrupo,numero,10);
for (k=0; numero[k]; k++);
       while (k < 4) do
        {
         strcat(texto[],'0');
         k = k+1;
        }
strcat(texto[],numero);
strcat(texto[],' | ');

// Días
c = 0;
m = 1;
k = 0;
while ((*g).dias[k] != 10) do
       {
        if ((*g).dias[c] == m) then
          {
           strcat(texto[],(*g).hora1);
           strcat(texto[],' | ');
           c = c + 1;
           k = k + 1;
          }
        else
          strcat(texto[],'      | ');
        m = m + 1;
       }
       while (m < 7) do
        {
         strcat(texto[],'      | ');
         m = m+1;
        }

// Salón
itoa((*g).numsalon,numero,10);
for (k=0; numero[k]; k++);
       while (k < 5) do
        {
         strcat(texto[],' ');
         k = k+1;
        }
strcat(texto[],numero);
```

```
        strcat(texto[],"  | ");

        // Cupo, tipo de salón
        c = (*g).tipo;
        switch(c)
              {
               case 0: strcat(texto[],"    ");
                       break;
               case 1: strcat(texto[],"    ");
                       break;
               case 2: strcat(texto[],"    ");
                       break;
               case 3: strcat(texto[],"30 ");
                       break;
               case 4: strcat(texto[],"60 ");
                       break;
               case 5: strcat(texto[],"    ");
                       break;
              }
        strcat(texto[],"  | ");

        j = j + 1;
        texto[][0] = '\n';
        texto[][1] = '\r';
        texto[][2] = '\0';

        j = j + 1;
        // Poner siguiente linea
        strcpy(texto[],"  |      |                              | ");
        strcat(texto[],"                                 |              |      | ");

        m = 1;
        k = 0;
        c = 0;
        while ((*g).dias[k] != 10)
              {
               if ((*g).dias[c] == m) then
                 {
                  strcat(texto[],(*g).hora2);
                  strcat(texto[],"  | ");
                  k = k + 1;
                  c = c + 1;
                 }
               else
                strcat(texto[],"      | ");
               m = m + 1;
              }
              while (m < 7) do
              {
               strcat(texto[],"      | ");
               m = m+1;
              }
        strcat(texto[],"      |      | ");

        j = j + 1;
        texto[][0] = '\n';
        texto[][1] = '\r';
        texto[][2] = '\0';

        j = j + 1;
strcpy(texto[],"  └──────┴──────────────────────────────┴──────────────┴──────┴──────┘");

strcat(texto[],"
```

```
                                                    ");
      j = j + 1;
      texto[j][0] = '\r';
      texto[j][1] = '\0';

      nlinea = nlinea + 1;
      i = i +1;
     }
   }
  for (k=0; k <= j; k++)                          // Manda a impresora el arreglo
   {
    a = 1;
    while (a == 1) do
          a = verificar(texto[k]);
   for (n=0; texto[k][n]; n++)
    biosprint(0,texto[k][n],0);
   }
 biosprint(0,'\F',0);                             // Salto de página
for (i = 0; i <= 59; i++)                         // Libera el espacio reservado
   farfree(texto[i]);
}


//   Devuelve la diferencia de horas entre cadena1 y cadena2, el resultado se devuelve en cadena2
void diferencia_horas(char cadena1[100], char cadena2[100])
{
 int horas, minutos, minuto1, minuto2, k, a, hora1, hora2,c;
 char cadena[100], min1[100];

 hora1=atoi(cadena1);
 hora2=atoi(cadena2);
 min1[0] = '\0';
 a = 0;
 for (k=0; cadena1[k]; k++)
  {
   if (!isdigit(cadena1[k]))
         a = k + 1;
  }
 if (a > 0)
  {
   for (c=0; cadena1[a]; a++,c++)
         min1[c] = cadena1[a];
  }
 minuto1=atoi(min1);
 a = 0;
 for (k=0; cadena2[k]; k++)
  {
   if (!isdigit(cadena2[k]))
    a = k + 1;
  }
 if (a > 0)
  {
   for (c=0; cadena2[a]; a++,c++)
    min1[c] = cadena2[a];
  }
 minuto2=atoi(min1);
 if (minuto1 > minuto2)
  {
   minuto1 = 60 - minuto1;
   minuto2 = 60 - minuto2;
   hora1 = hora1 + 1;
   minutos = 60 - (minuto2 - minuto1);
   horas = hora2 - hora1;
  }
 else
```

```
    {
      horas = abs(horas2 - hora1);
      minutos = minuto2 - minuto1;
    }
  itoa(horas,min1,10);
  for (k=0; min1[k]; k++);
  if (k == 1)
    {
     strcpy(cadena,"0");
     strcat(cadena,min1);
    }
  else
    strcpy(cadena,min1);
  strcat(cadena,":");
  itoa(minutos,min1,10);
  for (k=0; min1[k]; k++);
  if (k == 1)
    {
     strcat(cadena,"0");
     strcat(cadena,min1);
    }
  else
    strcat(cadena,min1);
  strcpy (cadena2,cadena);
}

//    Realiza la operación de adición de horas. Suma cadena1 y cadena2, el resultado se coloca en cadena2.
void suma_horas(char cadena1[100], char cadena2[100])
{
  int k, a, hora1, hora2,c, minuto1,minuto2,horas,minutos;
  char  min1[100], cadena[100];

  hora1=atoi(cadena1);
  hora2=atoi(cadena2);
  min1[0] = '\0';
  a = 0;
  for (k=0; cadena1[k]; k++)
    {
     if (!isdigit(cadena1[k]))
          a = k + 1;
    }
  if (a > 0)
    {
     for (c=0; cadena1[a]; a++,c++)
          min1[c] = cadena1[a];
    }
  minuto1=atoi(min1);
  a = 0;
  for (k=0; cadena2[k]; k++)
    {
     if (!isdigit(cadena2[k]))
       a = k + 1;
    }
  if (a > 0)
    {
     for (c=0; cadena2[a]; a++,c++)
      min1[c] = cadena2[a];
    }
  minuto2=atoi(min1);
  minutos = minuto1 + minuto2;

  if (minutos >= 60)
    {
      hora1 = hora1 + 1;
```

```
      minutos = minutos - 60;
    }
  horas = hora1 + hora2;
  itoa(horas,min1,10);
  for (k=0; min1[k]; k++);
  if (k == 1)
    {
     strcpy(cadena,"0");
     strcat(cadena,min1);
    }
  else
    strcpy(cadena,min1);
  strcat(cadena,":");
  itoa(minutos,min1,10);
  for (k=0; min1[k]; k++);
  if (k == 1)
    {
     strcat(cadena,"0");
     strcat(cadena,min1);
    }
  else
    strcat(cadena,min1);
  strcpy (cadena2,cadena);
}


//     Envía a impresora la información contenida en ARREGLO. Imprime listados por PROFESOR.
void imprime_profesor(struct MATERIA *p, struct SALIDA *salida, char semestre[10])
{
 struct MATERIA *r, *s;
 struct GRUPO *g, *e;
 int k, l, b, nlineas, pone, c,n,j;
 char *texto[60], numero[100], compara[100], almacena[100], suma[100];
                                                    // Reserva espacio en memoria para texto
 for (l = 0; l <= 59; l++)
  {
  texto[l] = (char *) farcalloc(160,sizeof(char));
  if (!texto[l]) then
    {
    outtextxy(100,100,"No hay espacio en memoria. Funcion imprime_extraord");
    exit(1);
    }
  }
 r = p;
 s = busca_materia(r,salida[0].materia);
 g = (*s).grupos;
 while ((*g).num != salida[0].grupo) do
         g = (*g).apun;
 strcpy(compara,(*g).nombreprofe);
 nlineas = 0;
 b = 0;
 l = 0;
 while (salida[l].materia != -2) do
  {
   if (b == 0) then
     {
          nlineas = 0;
                                                    // Imprime encabezados
          strcpy(texto[0],"  ");
          for (k=0; k<=38; k++)
            strcat(texto[0]," ");
          strcat(texto[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
          texto[1][0] = '\n';
          texto[1][1] = '\r';
          texto[1][2] = '\0';
```

```
        strcpy(texto[2]," ");
        for (k=0; k<=32; k++)
          strcat(texto[2]," ");
        strcat(texto[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
        texto[3][0] = '\n';
        texto[3][1] = 'r';
        texto[3][2] = '\0';

        strcpy(texto[4]," ");
        for (k=0; k<=45; k++)
          strcat(texto[4]," ");
        strcat(texto[4],"Ingeniería Mecánica Eléctrica");
        texto[5][0] = '\n';
        texto[5][1] = '\n';
        texto[5][2] = 'r';
        texto[5][3] = '\0';

        strcpy(texto[6]," ");
        for (k=0; k<=49; k++)
          strcat(texto[6]," ");
        strcat(texto[6],"HORARIOS POR PROFESOR");
        texto[7][0] = '\n';
        texto[7][1] = '\n';
        texto[7][2] = 'r';
        texto[7][3] = '\0';

        strcpy(texto[8]," ");
        for (k=0; k<=32; k++)
          strcat(texto[8]," ");
      strcat(texto[8],"PERIODO:  ");
      strcat(texto[8],semestre);
      texto[9][0] = '\n';
      texto[9][1] = 'r';
      texto[9][2] = '\0';

strcpy(texto[10],"┌──────────┬──────────────────────────────┬──────┬──────┬──────┬──────┬──────┐
┬");

strcat(texto[10],"
──────────┘ ");
      texto[11][0] = '\n';
      texto[11][1] = 'r';
      texto[11][2] = '\0';

      strcpy(texto[12],"|    RFC.    |       NOMBRE DEL PROFESOR        | CMAT | ");
      strcat(texto[12],"   NOMBRE DE LA MATERIA    | GPO. |  SALON  |  DIAS  |   HORARIO   | HORAS | ");
      texto[13][0] = '\n';
      texto[13][1] = 'r';
      texto[13][2] = '\0';

strcpy(texto[14],"|──────────┼──────────────────────────────┼──────┼──────┼──────┼──────┼──────┤
┤");
strcat(texto[14],"
──────────┘ ");
      texto[15][0] = '\n';
      texto[15][1] = 'r';
      texto[15][2] = '\0';
      i = 15;
      b = 1;
      pone = 1;
      }

      s = busca_materia(r,salida[i].materia);
      g = (*s).grupos;
      while ((*g).num != salida[i].grupo) do
```

```
        g = (*g).apun;
if (strcmp(compara,(*g).nombreprofe)) then  // Cuenta 14 lineas por hoja
      {
       nlineas = 14;
       strcpy(compara,(*g).nombreprofe);
      }
if  (nlineas > 13) then
      {
      // Anexar datos de teléfono, antigüedad
      j = j + 1;
      texto[][0] = '\r';
      texto[][1] = '\0';
      // Teléfono
      j = j + 1;
      strcpy(texto[],"└");
      for (k = 0; k<=45; k++)
        strcat(texto[],"—");
      j = j + 1;
      texto[][0] = '\n';
      texto[][1] = '\r';
      texto[][2] = '\0';
      // Suma total de horas
      j = j + 1;
      strcpy(texto[],"");
      for (k=0; k<=94; k++)
        strcat(texto[]," ");
      strcat(texto[],"TOTAL DE HORAS: ");
      strcat(texto[],almacena);
      j = j + 1;
      texto[][0] = '\n';
      texto[][1] = '\r';
      texto[][2] = '\0';
      j = j + 1;
      strcpy(texto[],"");
      strcat(texto[],"TELEFONO DE LA OFICINA: ");
      strcat(texto[],(*e).telofc);
      for (k = 0; (*e).telofc[k]; k++);
       while (k <= 14) do
        {
         strcat(texto[]," ");
         k = k+1;
        }
      strcat(texto[],"      TELEFONO DE LA CASA: ");
      strcat(texto[],(*e).telcasa);
      j = j + 1;
      texto[][0] = '\n';
      texto[][1] = '\r';
      texto[][2] = '\0';

      // Antigüedad
      j = j + 1;
      strcpy(texto[],"");
      strcat(texto[],"ANTIGUEDAD EN LA UNAM: ");
      strcat(texto[],(*e).antiunam);
      for (k = 0; (*e).antiunam[k]; k++);
       while (k <= 5) do
        {
         strcat(texto[]," ");
         k = k+1;
        }
      strcat(texto[],"      ANTIGUEDAD EN LA ENEP: ");
      strcat(texto[],(*e).antenep);
      j = j + 1;
      texto[][0] = '\n';
      texto[][1] = '\r';
```

```
texto[j][2] = '\0';

for (k=0; k <= j; k++)                          // Manda a impresión el arreglo
  {
   o = 1;
   while (o == 1) do
           o = verificar(texto[k]);
  for (n=0; texto[k][n]; n++)
    biosprint(0,texto[k][n],0);
  }
biosprint(0,'\f',0);                            // Salto de página
b = 0;
}
else
{
 e = g;
                                                // Imprime la información completa
if (pone == 1) then
  {
   j = j + 1;
   texto[j][0] = '\r';
   texto[j][1] = '\0';
   j = j + 1;
   strcpy(texto[j]," | ");
   // RFC del profesor
   strcat(texto[j],(*g).rfc);
   for (k=0; (*g).rfc[k]; k++);
     while (k < 12) do
           {
            strcat(texto[j]," ");
            k = k+1;
           }
   strcat(texto[j]," | ");

   //Nombre del profesor
   strcat(texto[j],(*g).nombreprofe);
   for (k=0; (*g).nombreprofe[k]; k++);
     while (k < 32) do
           {
            strcat(texto[j]," ");
            k = k+1;
           }
   strcat(texto[j]," | ");

   // Salta el renglón
   for (k=0; k <= 70; k++)
           strcat(texto[j]," ");
   strcat(texto[j]," | ");

   j = j + 1;
   texto[j][0] = '\n';
   texto[j][1] = '\r';
   texto[j][2] = '\0';

   j = j + 1;
strcpy(texto[j]," ├──────────────────┴──────────────────────────────────────┼────────
┬");

strcat(texto[j],"────────────────────────────────────────┬──────┬──────────┬────────┬──────
──────────┬───────────┤ ");
   j = j + 1;
   texto[j][0] = '\n';
   texto[j][1] = '\r';
   texto[j][2] = '\0';
```

```
        }
       else
        {
         j = j + 1;
         strcpy(texto[j]," |                                   |──────┬─");

strcat(texto[j],"──────────────────────────────────────────────────────────────┬────────┬──────────────┬───────────┬─────
──────────────┬────────────────| ");

         j = j + 1;
         texto[j][0] = '\n';
         texto[j][1] = '\r';
         texto[j][2] = '\0';
        }
       j = j + 1;
       strcpy(texto[j]," |             ");
       strcat(texto[j],"                                  | ");

       // Clave de la materia
       itoa((*s).clave,numero,10);
       for (k=0; numero[k]; k++);
         while (k < 4) do
          {
                strcat(texto[j],"0");
                k = k+1;
          }
       strcat(texto[j],numero);
       strcat(texto[j]," | ");

       // Nombre de la materia
       strcat(texto[j],(*s).nombre);
       for (k=0; (*s).nombre[k]; k++);
         while (k < 28) do
          {
                strcat(texto[j]," ");
                k = k+1;
          }
       strcat(texto[j]," | ");

      // Número de grupo
      itoa((*g).numgrupo,numero,10);
      for (k=0; numero[k]; k++);
        while (k < 4) do
         {
          strcat(texto[j],"0");
          k = k+1;
         }
      strcat(texto[j],numero);
      strcat(texto[j]," | ");

      // Salón
       itoa((*g).numsalon,numero,10);
      if ((*g).tipo == 2) then
        strcat(texto[j],"L-");
       else
        strcat(texto[j],"A-");
       for (k=0; numero[k]; k++);
        while (k < 5) do
         {
          strcat(texto[j]," ");
          k = k+1;
         }
      if ((*s).numsalon == 0) then
       {
        strcat(texto[j]," ");
```

```
        }
        else
           strcat(texto[],numero);
        strcat(texto[]," | ");

       // Dias
       if ((*g).dias[0] == 10) then
        {
         strcat(texto[],"      ");
         c = 0;
        }
       else
        {
         for (k=0; (*g).dias[k] != 10; k++)
          {
           c = (*g).dias[k];
           switch(c)
                 {
                  case 1: strcat(texto[],"L ");
                          break;
                  case 2: strcat(texto[],"M ");
                          break;
                  case 3: strcat(texto[],"M ");
                          break;
                  case 4: strcat(texto[],"J ");
                          break;
                  case 5: strcat(texto[],"V ");
                          break;
                  case 6: strcat(texto[],"S ");
                          break;
                 }
          }
          c = k - 1;
         if (k == 2) then
           strcat(texto[]," ");
         if (k == 1) then
           strcat(texto[],"  ");
         while (k < 3) do
          {
           strcat(texto[]," ");
           k = k + 1;
          }
        }
       strcat(texto[]," | ");

       // Horario
       strcat(texto[],(*g).hora1);
       strcat(texto[]," ");
       strcat(texto[],(*g).hora2);
       strcat(texto[]," | ");

// Total horas
diferencia_horas((*g).hora1,(*g).hora2);
strcpy(suma,(*g).hora2);
for (k=0; k < c; k++)
      suma_horas((*g).hora2,suma);
if (c == 0) then
     strcpy(suma,"00:00");
 strcat(texto[],suma);
if (pone == 1) then
     strcpy(almacena,"00:00");
suma_horas(almacena,suma);

strcpy(almacena,suma);
pone = 0;
```

```
    for (k=0; suma[k]; k++);
          while (k < 5) do
           {
           strcat(texto[],' ');
           k = k + 1;
           }
    strcat(texto[],' | ');

    j = j + 1;
    texto[][0] = '\n';
    texto[][1] = '\r';
    texto[][2] = '\0';

    j = j + 1;
    strcpy(texto[],' ');
    for (k=0; k<= 44; k++)
          strcat(texto[],' ');

strcat(texto[],'└──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┘');

    j = j + 1;
    texto[][0] = '\r';
    texto[][1] = '\0';
    nlinea = nlinea + 1;
    i = i + 1;
   }
  }
  j = j + 1;
  texto[][0] = '\r';
  texto[][1] = '\0';
  // Tel4fono
  j = j + 1;
  strcpy(texto[],' ┌');
  for (k = 0; k<=45; k++)
    strcat(texto[],'─');
  j = j + 1;
  texto[][0] = '\n';
  texto[][1] = '\r';
  texto[][2] = '\0';
  // Suma total de horas
  j = j + 1;
  strcpy(texto[],' ');
  for (k=0; k<=94; k++)
    strcat(texto[],' ');
  strcat(texto[],'TOTAL DE HORAS:  ');
  strcat(texto[],almacena);
  j = j + 1;
  texto[][0] = '\n';
  texto[][1] = '\r';
  texto[][2] = '\0';

  j = j + 1;
  strcpy(texto[],' ');
  strcat(texto[],'TELEFONO DE LA ORCINA:  ');
  strcat(texto[],(*e).telofc);
  for (k = 0; (*e).telofc[k]; k++);
   while (k <= 14) do
    {
     strcat(texto[],' ');
     k = k+1;
    }
  strcat(texto[],'      TELEFONO DE LA CASA:  ');
  strcat(texto[],(*e).telcasa);
  j = j + 1;
  texto[][0] = '\n';
```

```
    tado[j][1] = '\r';
    tado[j][2] = '\0';

    // Antigüedad
    j = j + 1;
    strcpy(tado[j]," ");
    strcat(tado[j],"ANTIGUEDAD EN LA UNAM:  ");
    strcat(tado[j],(*e).antunam);
    for (k = 0; (*e).antunam[k]; k++);
     while (k <= 5) do
       {
        strcat(tado[j]," ");
       k = k+1;
       }
    strcat(tado[j],"      ANTIGUEDAD EN LA ENEP:  ");
    strcat(tado[j],(*e).antenep);
    j = j + 1;
    tado[j][0] = '\n';
    tado[j][1] = '\r';
    tado[j][2] = '\0';

    for (k=0; k <= j; k++)                          // Manda el arreglo a impresión
     {
       c = 1;
       while (c == 1) do
             c = verificar(tado[k]);
      for (n=0; tado[k][n]; n++)
       biosprint(0,tado[k][n],0);
     }
   biosprint(0,'\f',0);                             // Salto de página
   for (i = 0; i <= 99; i++)                        // Libera el espacio reservado para tado
      farfree(tado[i]);
}


//     Con el número día, mes y año, devuelve el día de la semana.
void dia_semana(int dia, int mes, int ano, char numero[100])
{
 struct tm t;

 ano = ano - 1900;
 mes = mes - 1;
 strcpy(numero,"  ");
 t.tm_year = ano;
 t.tm_mon = mes;
 t.tm_mday = dia;
 mktime(&t);
 switch(t.tm_wday)
  {
   case 0: strcpy(numero,"DOM");
           break;
   case 1: strcpy(numero,"LUN");
           break;
   case 2: strcpy(numero,"MAR");
           break;
   case 3: strcpy(numero,"MIE");
           break;
   case 4: strcpy(numero,"JUE");
           break;
   case 5: strcpy(numero,"VIE");
           break;
   case 6: strcpy(numero,"SAB");
           break;
  }
}
```

```
//   Envía a impresora la información contenida en ARREGLO. Imprime listados de EXAMEN EXTRAORDINARIO.
void imprime_extraord(struct MATERIA *p, struct ARREGLO *arreglo, int tipo, char semestre[10])
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 char *texto[60], numero[100], cadena[100];
 int k,i,b,ntinea,j,n,c;
                                                    // Reserva espacio en memoria para texto
 for (i = 0; i <= 59; i++)
 {
 texto[i] = (char *) farcalloc(160,sizeof(char));
 if (!texto[i]) then
   {
   outtextxy(100,100,"No hay espacio en memoria. Funcion imprime_extraord");
   exit(1);
   }
 }
 i=0;
 r=p;
 b=0;
 ntinea=0;
 s = busca_materia(r,arreglo[0].materia);
 e = (*s).examen;
 while ((*e).num != arreglo[0].grupo) do
         e = (*e).sig;
 while (arreglo[i].materia != -2) do
  {
   if (b == 0) then
        {
         ntinea = 0;
                                                    // Imprime encabezados
         strcpy(texto[0]," ");
         for (k=0; k<=36; k++)
           strcat(texto[0]," ");
         strcat(texto[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
         texto[1][0] = '\n';
         texto[1][1] = '\r';
         texto[1][2] = '\0';

         strcpy(texto[2]," ");
         for (k=0; k<=32; k++)
           strcat(texto[2]," ");
         strcat(texto[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
         texto[3][0] = '\n';
         texto[3][1] = '\r';
         texto[3][2] = '\0';

         strcpy(texto[4]," ");
         for (k=0; k<=43; k++)
           strcat(texto[4]," ");
         strcat(texto[4],"Ingeniería Mecanica Electrica");
         texto[5][0] = '\n';
         texto[5][1] = '\n';
         texto[5][2] = '\r';
         texto[5][3] = '\0';

         strcpy(texto[6]," ");
         for (k=0; k<=35; k++)
           strcat(texto[6]," ");
         strcat(texto[6],"EXAMEN EXTRAORDINARIO 2a. VUELTA PERIODO: ");
         strcat(texto[6],semestre);
         texto[7][0] = '\n';
         texto[7][1] = '\n';
         texto[7][2] = '\n';
         texto[7][3] = '\r';
```

```
        texto[7][4] = '\0';

strcpy(texto[8]," ┌──────────┬────────────────────────────────────────────────────┬──────────┬──────────────────────
──────────────────────────────────┬");
        strcat(texto[8],"────────────────┬──────────┬────────┬────────────┬──────────────┬────────────────────┐ ");
        texto[9][0] = '\n';
        texto[9][1] = '\r';
        texto[9][2] = '\0';

        strcpy(texto[10]," | CMAT |     NOMBRE DE LA MATERIA     | EXAM |      NOMBRE DEL PROFESOR        | ");
        strcat(texto[10],"   RFC.    | GPO. | DIA | HORA  |  SALON  |  FECHA   | ");
        texto[11][0] = '\n';
        texto[11][1] = '\r';
        texto[11][2] = '\0';

strcpy(texto[12]," └──────────┴────────────────────────────────────────────────────┴──────────┴──────────────────────
──────────────────────────────────┴");
        strcat(texto[12],"────────────────┴──────────┴────────┴────────────┴──────────────┴────────────────────┘ ");
        texto[13][0] = '\r';
        texto[13][1] = '\0';
        j = 13;
        b = 1;
      }
   s = busca_materia(r,arreglo[i].materia);
   e = (*s).examen;
   while ((*e).num != arreglo[0].grupo) do
          e = (*e).sig;
   if (nlinea > 3) then
     {
                                                  // Manda a impresión el arreglo
        for (k=0; k <= j; k++)
         {
          c = 1;
          while (c == 1) do
            c = verificar(texto[k]);
         for (n=0; texto[k][n]; n++)
           biosprint(0,texto[k][n],0);
         }
        biosprint(0,'\f',0);                      // Salto de página
        b = 0;
     }
   else
     {
                                                  // Imprime la información completa
        j = j + 1;

strcpy(texto[j]," ├──────────┼────────────────────────────────────────────────────┼──────────┼──────────────────────
──────────────────────────────────┼");
        strcat(texto[j],"────────────────┼──────────┼────────┼────────────┼──────────────┼────────────────────┤ ");
        j = j+1;
        texto[j][0] = '\n';
        texto[j][1] = '\r';
        texto[j][2] = '\0';
        j = j+1;
        strcpy(texto[j]," |      |                              |    |                                 | ");
        strcat(texto[j],"             |    |     |       |          |        | ");
        j = j+1;
        texto[j][0] = '\n';
        texto[j][1] = '\r';
        texto[j][2] = '\0';

        // Clave de la materia
        j = j+1;
        strcpy(texto[j]," | ");
        itoa((*s).clave,numero,10);
```

```
for (k=0; numero[k]; k++);
while (k < 4) do
 {
  strcat(texto[],"0");
  k = k + 1;
 }
strcat(texto[],numero);
strcat(texto[]," | ");

// Nombre de la materia
strcat(texto[],(*e).nombre);
for (k=0; (*e).nombre[k]; k++);
 while (k < 28) do
  {
   strcat(texto[]," ");
   k = k+1;
  }
strcat(texto[]," | ");

// Examen
strcat(texto[],"EXT2");
strcat(texto[]," | ");

//Nombre del profesor
strcat(texto[],(*e).nombreprofe);
for (k=0; (*e).nombreprofe[k]; k++);
 while (k < 32) do
  {
   strcat(texto[]," ");
   k = k+1;
  }
strcat(texto[]," | ");

// RFC del profesor
if (tipo == 254) then
 {
  strcat(texto[],(*e).rfc);
  for (k=0; (*e).rfc[k]; k++);
   while (k < 12) do
        {
         strcat(texto[]," ");
         k = k+1;
        }
 }
else
 strcat(texto[],"          ");
strcat(texto[]," | ");

// Número de grupo
itoa((*e).numgrupo,numero,10);
for (k=0; numero[k]; k++);
 while (k < 4) do
  {
   strcat(texto[],"0");
   k = k+1;
  }
strcat(texto[],numero);
strcat(texto[]," | ");

// Día
strcat(texto[],"MIE");
strcat(texto[]," | ");

// Hora
strcat(texto[],(*e).hora1);
```

```
strcat(texto[],"  | ");

// Salón
itoa((*e).numesion,numero,10);
if ((*e).tiposion == 2) then
  strcat(texto[],"L-");
else
  strcat(texto[],"A-");
for (k=0; numero[k]; k++);
  while (k < 5) do
    {
      strcat(texto[]," ");
      k = k+1;
    }
if ((*e).numesion == 0) then
  {
   strcat(texto[]," ");
  }
else
    strcat(texto[],numero);
strcat(texto[]," | ");

// Fecha
itoa((*e).dia1,numero,10);
for (k=0; numero[k]; k++);
while (k < 2) do
 {
   strcat(texto[],"0");
   k = k+1;
 }
strcat(texto[],numero);
strcat(texto[],"/");
itoa((*e).mes1,numero,10);
for (k=0; numero[k]; k++);
while (k < 2) do
 {
   strcat(texto[],"0");
   k = k+1;
 }
strcat(texto[],numero);
strcat(texto[],"/");
itoa((*e).ano1,numero,10);
strcat(texto[],numero);
strcat(texto[]," | ");
// Siguiente renglón
j = j+1;
texto[j][0] = '\n';
texto[j][1] = 'r';
texto[j][2] = '\0';

j = j+1;
strcpy(texto[]," |     |                    | ");
// Examen 2
strcat(texto[],"EXT2");
strcat(texto[]," | ");

//Nombre del profesor suplente
strcat(texto[],(*e).profe);
for (k=0; (*e).profe[k]; k++);
  while (k < 32) do
    {
      strcat(texto[]," ");
      k = k+1;
    }
strcat(texto[]," | ");
```

```
// RFC del profesor suplente
if (tipo == 254) then
 {
  strcat(texto[],(*e).rfc2);
  for (k=0; (*e).rfc2[k]; k++);
    while (k < 12) do
          {
           strcat(texto[]," ");
           k = k+1;
          }
 }
else
  strcat(texto[],"           ");
strcat(texto[]," | ");

// Número de grupo
itoa((*e).numgrupo,numero,10);
for (k=0; numero[k]; k++);
  while (k < 4) do
    {
     strcat(texto[],"0");
     k = k+1;
    }
strcat(texto[],numero);
strcat(texto[]," | ");

// Día
if (((*e).dia1 == 0) || ((*e).mes1 == 0) || ((*e).ano1 == 0)) then
  strcat(texto[],"   ");
else
 {
  dia_semana((*e).dia1,(*e).mes1,(*e).ano1,numero);
  strcat(texto[],numero);
 }
strcat(texto[]," | ");

// Hora
strcat(texto[],(*e).hora1);
strcat(texto[]," | ");

// Salón
itoa((*e).numsalon,numero,10);
if ((*e).tposalon == 2) then
  strcat(texto[],"L-");
else
  strcat(texto[],"A-");
for (k=0; numero[k]; k++);
  while (k < 5) do
    {
     strcat(texto[]," ");
     k = k+1;
    }
if ((*e).numsalon == 0) then
  {
   strcat(texto[]," ");
  }
else
    strcat(texto[],numero);
strcat(texto[]," | ");

// Fecha
itoa((*e).dia1,numero,10);
for (k=0; numero[k]; k++);
while (k < 2) do
 {
```

```
          strcat(texto[j],"0");
          k = k+1;
         }
        strcat(texto[j],numero);
        strcat(texto[j],"/");
        itoa((*e).mes1,numero,10);
        for (k=0; numero[k]; k++);
        while (k < 2) do
         {
          strcat(texto[j],"0");
          k = k+1;
         }
        strcat(texto[j],numero);
        strcat(texto[j],"/");
        if ((*e).ano1 == 0) then
          strcat(texto[j],"00");
        else
         {
         itoa((*e).ano1,numero,10);
         n = 0;
         for (k=0; numero[k]; k++)
          {
           if (k >= 2) then
            {
             cadena[n] = numero[k];
             n = n + 1;
            }
          }
          cadena[n] = '\0';
         strcat(texto[j],cadena);
         }
        strcat(texto[j]," | ");
        j = j+1;
        texto[j][0] = '\n';
        texto[j][1] = '\r';
        texto[j][2] = '\0';
        j = j+1;

strcpy(texto[j]," └────────┴──────────────────────────────────────┴────────┴
──────────────────────────────────────────────┘");
        strcat(texto[j],"──────────┴──────────┴─────────┴──────────────┴──────────────────┴─────────────────┘");
        j = j+1;
        texto[j][0] = '\r';
        texto[j][1] = '\0';

        nlinea = nlinea + 1;
        l = l + 1;
    }
  }
                                          // Imprimir último dato
  for (k=0; k <= j; k++)
   {
    c = 1;
    while (c == 1) do
        c = verificar(texto[k]);
   for (n=0; texto[k][n]; n++)
        biosprint(0,texto[k][n],0);
   }
  biosprint(0,'\f',0);                     // Salto de página
for (i = 0; i <= 59; i++)                  // Libera el espacio reservado para texto
  farfree(texto[i]);
}
```

```
// Envía a impresora la información contenida en ARREGLO. Imprime listados de EXAMEN FINAL
void imprime_final(struct MATERIA *p, struct ARREGLO *arreglo, int tipo, char semestre[10])
{
 struct MATERIA *r,*s;
 struct EXAMEN *e;
 char *tasdo[80], numero[100], cadena[100];
 int k,l,b,nlineas,j,n,compara,pon,cuenta,c;
                                                    // Reserva espacio en memoria para tasdo
 for (l = 0; l <= 59; l++)
 {
 tasdo[l] = (char *) farcalloc(180,sizeof(char));
 if (!tasdo[l]) then
   {
   outtasdoxy(100,100,"No hay espacio en memoria. Funcion imprime_extraord");
   exit(1);
   }
 }
 l=0;
 r=p;
 b=0;
 nlineas=0;
 s = busca_materia(r,arreglo[0].materia);
 e = (*s).examen;
 while ((*e).num != arreglo[0].grupo) do
         e = (*e).sig;
 compara = (*e).clave;
 while (arreglo[l].materia != -2) do
  {
   if (b == 0) then
        {
         nlineas = 0;
                                                    // Imprime encabezados
         strcpy(tasdo[0]," ");
         for (k=0; k<=44; k++)
           strcat(tasdo[0]," ");
         strcat(tasdo[0],"UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO");
         tasdo[1][0] = '\n';
         tasdo[1][1] = '\r';
         tasdo[1][2] = '\0';

         strcpy(tasdo[2]," ");
         for (k=0; k<=39; k++)
           strcat(tasdo[2]," ");
         strcat(tasdo[2],"ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ARAGON"");
         tasdo[3][0] = '\n';
         tasdo[3][1] = '\r';
         tasdo[3][2] = '\0';

         strcpy(tasdo[4]," ");
         for (k=0; k<=50; k++)
           strcat(tasdo[4]," ");
         strcat(tasdo[4],"Ingeniería Mecanica Electrica");
         tasdo[5][0] = '\n';
         tasdo[5][1] = '\n';
         tasdo[5][2] = '\r';
         tasdo[5][3] = '\0';

         strcpy(tasdo[6]," ");

         for (k=0; k<=50; k++)
           strcat(tasdo[6]," ");
         strcat(tasdo[6],"EXAMENES FINALES  PERIODO: ");
         strcat(tasdo[6],semestre);
         tasdo[7][0] = '\n';
         tasdo[7][1] = '\n';
```

```
          texto[7][2] = '\n';
          texto[7][3] = '\r';
          texto[7][4] = '\0';

strcpy(texto[8]," ┌──────────┬──────────────────────────────────────────────────────────┬──────────────────────────────
──────────────────────────────────┬");

strcat(texto[8],"──────────────────────────────────────┬──────────┬──────────────┬──────────┬──────────┬──────────────────────┬─────────┬
──────────────────────┐ ");
          texto[9][0] = '\n';
          texto[9][1] = '\r';
          texto[9][2] = '\0';

          strcpy(texto[10]," | CMAT |      NOMBRE DE LA MATERIA      |        NOMBRE DEL PROFESOR        | ");
          strcat(texto[10],"   RFC.     | GPO. |  SALON  | HORA  | DIA | FECHA 1a.v | DIA | FECHA 2a.v | ");
          texto[11][0] = '\n';
          texto[11][1] = '\r';
          texto[11][2] = '\0';

strcpy(texto[12]," └──────────┴──────────────────────────────────────────────────────────┴──────────────────────────────
──────────────────────────────────┴");

strcat(texto[12],"──────────────────────────────────────┴──────────┴──────────────┴──────────┴──────────┴──────────────────────┴─────────┴
──────────────────────┘ ");
          texto[13][0] = '\r';
          texto[13][1] = '\0';
          j = 13;
          b = 1;
          pon = 0;
          cuenta = 0;
        }
    s = busca_materia(r,arreglo[i].materia);
    e = (*s).examen;
   while ((*e).num != arreglo[i].grupo) do
          e = (*e).sig;
   if (compara != (*s).clave) then
    {
     compara = (*s).clave;
     pon = 0;
     cuenta = cuenta + 1;
    }
   if ((nlineas > 27) || (cuenta > 1)) then
    {
                                                            // Imprime última línea
         j = j+1;
         strcpy(texto[j]," └──────────┴─────────────────────────────────────────────────────────────┘");
                                                            // Manda el arreglo a impresora
         for (k=0; k <= j; k++)
          {
            c = 1;
            while (c == 1) do
               c = verificar(texto[k]);
           for (n=0; texto[k][n]; n++)
             biosprint(0,texto[k][n],0);
          }
         biosprint(0,'\f',0);                               // Salto de página
         b = 0;
    }

   else
    {
                                                            // Imprime la información completa
         j = j + 1;
         if (pon == 0) then
          {
```

```
strcpy(texto[],"                                                    ");
strcat(texto[],"                                                    ");
        }
        else
        {
         strcpy(texto[],"                                           ");
strcat(texto[],"                                                    ");
        }
        i = i+1;
        texto[][0] = '\n';
        texto[][1] = '\r';
        texto[][2] = '\0';
        i = i+1;
        strcpy(texto[],"                                            ");
        strcat(texto[],"                                            ");
        i = i+1;
        texto[][0] = '\n';
        texto[][1] = '\r';
        texto[][2] = '\0';

        // Clave de la materia
        i = i+1;
        if (pon == 0) then
        {
         strcpy(texto[]," ");
         itoa((*s).clave,numero,10);
         for (k=0; numero[k]; k++);
         while (k < 4) do
          {
                strcat(texto[],"0");
                k = k + 1;
          }
         strcat(texto[],numero);
         strcat(texto[]," ");

         // Nombre de la materia
         strcat(texto[],(*s).nombre);
         for (k=0; (*s).nombre[k]; k++);
          while (k < 28) do
                {
                 strcat(texto[]," ");
                 k = k+1;
                }
         strcat(texto[]," ");
        pon = 1;
        }
        else
        {
         strcpy(texto[]," ");
         strcat(texto[],"                                           ");
        }


        //Nombre del profesor
        strcat(texto[],(*e).nombreprofe);
        for (k=0; (*e).nombreprofe[k]; k++);
         while (k < 32) do
          {
           strcat(texto[]," ");
```

```
      k = k+1;
    }
 strcat(texto[],' | ');

 // RFC del profesor
 if (tipo == 254) then
  {
    strcat(texto[],(*e).rfc);
    for (k=0; (*e).rfc[k]; k++);
      while (k < 12) do
            {
              strcat(texto[],' ');
              k = k+1;
            }
  }
 else
  strcat(texto[],'            ');
 strcat(texto[],' | ');

 // Número de grupo
 itoa((*e).numgrupo,numero,10);
 for (k=0; numero[k]; k++);
   while (k < 4) do
     {
       strcat(texto[],'0');
       k = k+1;
     }
 strcat(texto[],numero);
 strcat(texto[],' | ');

 // Salón
 itoa((*e).numsalon,numero,10);
 if ((*e).tiposalon == 2) then
   strcat(texto[],'L-');
 else
   strcat(texto[],'A-');

 for (k=0; numero[k]; k++);
   while (k < 5) do
     {
       strcat(texto[],' ');
       k = k+1;
     }
 if ((*e).numsalon == 0) then
   {
    strcat(texto[],' ');
   }
 else
     strcat(texto[],numero);
 strcat(texto[],' | ');

 // Hora
 strcat(texto[],(*e).hora1);
 strcat(texto[],' | ');

 // Día
 if (((*e).dia1 == 0) || ((*e).mes1 == 0) || ((*e).ano1 == 0)) then
   strcat(texto[],'   ');

 else
  {
   dia_semana((*e).dia1,(*e).mes1,(*e).ano1,numero);
   strcat(texto[],numero);
  }
 strcat(texto[],' | ');
```

```
// Fecha
itoa((*e).dia1,numero,10);
for (k=0; numero[k]; k++);
strcat(texto[],"");
while (k < 2) do
 {
  strcat(texto[],"0");
  k = k+1;
 }
strcat(texto[],numero);
strcat(texto[],"/");
itoa((*e).mes1,numero,10);
for (k=0; numero[k]; k++);
while (k < 2) do
 {
  strcat(texto[],"0");
  k = k+1;
 }
strcat(texto[],numero);
strcat(texto[],"/");
if ((*e).ano1 == 0) then
  strcat(texto[],"00");
else
 {
  itoa((*e).ano1,numero,10);
  n = 0;
  for (k=0; numero[k]; k++)
   {
    if (k >= 2) then
     {
      cadena[n] = numero[k];
      n = n + 1;
     }
   }
   cadena[n] = '\0';
  strcat(texto[],cadena);
 }
strcat(texto[],"  | ");

// Dia 2
if ((((*e).dia2 == 0) || ((*e).mes2 == 0) || ((*e).ano2 == 0)) then
  strcat(texto[],"   ");
else
 {
  dia_semana((*e).dia2,(*e).mes2,(*e).ano2,numero);
  strcat(texto[],numero);
 }
strcat(texto[],"  | ");

// Fecha 2
itoa((*e).dia2,numero,10);
for (k=0; numero[k]; k++);
strcat(texto[],"");
while (k < 2) do
 {
  strcat(texto[],"0");
  k = k+1;
 }
strcat(texto[],numero);
strcat(texto[],"/");
itoa((*e).mes2,numero,10);
for (k=0; numero[k]; k++);
while (k < 2) do
 {
  strcat(texto[],"0");
```

```
               k = k+1;
              }
             strcat(texto[j],numero);
             strcat(texto[j],"/");
             if ((*e).ano2 == 0) then
               strcat(texto[j],"00");
             else
              {
               itoa((*e).ano2,numero,10);
               n = 0;
               for (k=0; numero[k]; k++)
                {
                 if (k >= 2) then
                  {
                   cadena[n] = numero[k];
                   n = n + 1;
                  }
                }
               cadena[n] = '\0';
              strcat(texto[j],cadena);
             }
            strcat(texto[j]," │ ");

            j = j+1;
            texto[j][0] = '\n';
            texto[j][1] = '\r';
            texto[j][2] = '\0';
            j = j+1;
            strcpy(texto[j],"                    └────────────────────────────────────────────────┘");

strcat(texto[j],"──────────────────┴────────┴───────────┴──────┴──────┴────────────────┴─────────
────────────────┘ ");
            j = j+1;
            texto[j][0] = '\r';
            texto[j][1] = '\0';
            nlinea = nlinea + 1;
            i = i + 1;
     }
   }

   // Imprime última línea
   j = j+1;
   strcpy(texto[j]," └────────┴──────────────────────────────────────────────────┘");
                                                        // Imprimir último dato
   for (k=0; k <= j; k++)
    {
     c = 1;
     while (c == 1) do
            c = verificar(texto[k]);
    for (n=0; texto[k][n]; n++)
          biosprint(0,texto[k][n],0);
    }
   biosprint(0,'\f',0);                                 // Salto de página
for (i = 0; i <= 59; i++)                               // Libera el espacio reservado en memoria para texto
   farfree(texto[i]);
}


//  Llama a las funciones que mandan a impresión un archivo. Se obtiene el listado según la variable tipo.
void imprimir(struct MATERIA *p, struct ARREGLO *arreglo, int tipo, char semestre[10])
{
 int i;

 pon(tope,352,COPY_PUT);                   // Pone ventana gráfica 352
 texto("Confirme la impresión . . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
```

```
 pon(tope,259,COPY_PUT);                // Pone ventana gráfica 259
 pon(tope,254,COPY_PUT);                // Pone ventana gráfica 254
 texto("NO",259,BLUE,SANS_SERIF_FONT,1,2,2,2);
 texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
 pon(tope,259,NOT_PUT);                 // Pone ventana gráfica del cursor
 i = opciones1(253,253,254,9);
 quita(tope);              // Quita ventana gráfica del cursor
 quita(tope);              // Quita ventana gráfica 254
 quita(tope);              // Quita ventana gráfica 253
 quita(tope);              // Quita ventana gráfica 352
 if (i == 254) then
  {
     pon(tope,352,COPY_PUT);                // Pone ventana gráfica 352
     texto("Desea imprimir el R.F.C. del profesor . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
     pon(tope,253,COPY_PUT);                // Pone ventana gráfica 253
     pon(tope,254,COPY_PUT);                // Pone ventana gráfica 254
     texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
     texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
     i = -1;
     pon(tope,253,NOT_PUT);                 // Pone ventana gráfica del cursor
     while (i == -1) do
          i = opciones1(253,253,254,9);
     quita(tope);              // Quita ventana gráfica del cursor
     quita(tope);              // Quita ventana gráfica 254
     quita(tope);              // Quita ventana gráfica 253
     quita(tope);              // Quita ventana gráfica 352
   pon(tope,156,COPY_PUT);                // Pone ventana gráfica 156
   cuadro(156,BLACK);
   pon(tope,157,COPY_PUT);                // Pone ventana gráfica 157
   texto("I M P R I M I E N D O . . .",157,DARKGRAY,TRIPLEX_SCR_FONT,1,2,2,2);
                                                  // Llama a la función de impresión según tipo
   switch(tipo)
   {
    case 1:imprime_grupo(p,arreglo,i,semestre);
              break;
    case 2:imprime_materias(p,arreglo,i,semestre);
              break;
    case 4:imprime_extraord(p,arreglo,i,semestre);
              break;
    case 5:imprime_final(p,arreglo,i,semestre);
              break;
   }
   quita(tope);              // Quita ventana gráfica 157
   quita(tope);              // Quita ventana gráfica 156
  }
}


/*    Llama a las funciones que buscan la información de profesor para crear el archivo salida, según el valor de la
  variable tip. */
void imprimir_profe_salida(struct MATERIA *p, char semestre[10], int b, int tip, char profesor[100], int cuenta)
{
 struct MATERIA *r;
 struct SALIDA *salida;
 int i,k,ultimo;
                                                  // Reserva el espacio en memoria para salida
 salida = (struct SALIDA *) farcalloc(cuenta+1,sizeof(struct SALIDA));

  if (!salida)
    {
     printf("Error.");
     exit(1);
    }
 salida[0].grupo = 0;
 salida[0].materia = 0;
```

```
salida[0].numero = 0;
r = p;
pon(tope,352,COPY_PUT);                    // Pone ventana gráfica 352
texto("Confirme la impresión . . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
pon(tope,253,COPY_PUT);                    // Pone ventana gráfica 253
pon(tope,254,COPY_PUT);                    // Pone ventana gráfica 254
texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
pon(tope,233,NOT_PUT);                     // Pone ventana gráfica del cursor
i = opciones1(253,253,254,9);
quita(tope);              // Quita ventana gráfica del cursor
quita(tope);              // Quita ventana gráfica 254
quita(tope);              // Quita ventana gráfica 253
quita(tope);              // Quita ventana gráfica 352
if (i == 254) then
 {
  pon(tope,156,COPY_PUT);                  // Pone ventana gráfica 156
  cuadro(156,BLACK);
  pon(tope,157,COPY_PUT);                  // Pone ventana gráfica
  texto("I M P R I M I E N D O  . . .",157,DARKGRAY,TRIPLEX_SCR_FONT,1,2,2,2);
  switch(tip)
   {
        case 342: if (b == 0) then
                    busca_profe_tipo_imprime(r,profesor,salida,1);
                  if (b == 1) then
                    busca_profe_cadena_tipo_imprime(r,profesor,salida,1);
                  if (b == 2) then
                    busca_profe_todos_tipo_imprime(r,salida,1);
                  if (b == 3) then
                    busca_profe_materia_tipo_imprime(r,salida,1);
                  break;
        case 343: if (b == 0) then
                    busca_profe_tipo_imprime(r,profesor,salida,2);
                  if (b == 1) then
                    busca_profe_cadena_tipo_imprime(r,profesor,salida,2);
                  if (b == 2) then
                    busca_profe_todos_tipo_imprime(r,salida,2);
                  if (b == 3) then
                    busca_profe_materia_tipo_imprime(r,salida,2);
                  break;
        case 344: if (b == 0) then
                    busca_profe_tipo_imprime(r,profesor,salida,3);
                  if (b == 1) then
                    busca_profe_cadena_tipo_imprime(r,profesor,salida,3);
                  if (b == 2) then
                    busca_profe_todos_tipo_imprime(r,salida,3);
                  if (b == 3) then
                    busca_profe_materia_tipo_imprime(r,salida,3);
                  break;
        case 345: if (b == 0) then
                    busca_profe_imprime(r,profesor,salida);
                  if (b == 1) then
                    busca_profe_cadena_imprime(r,profesor,salida);
                  if (b == 2) then
                    busca_profe_todos_imprime(r,salida);
                  if (b == 3) then
                    busca_profe_todos_materia_imprime(r,salida);
                  break;
   }
  ultimo = 0;
  for (k=0; salida[k].grupo != -2; k++)
        ultimo = ultimo + 1;
  ultimo = ultimo - 1;
  if (ultimo == -1) then
    ultimo = 0;
```

```
    ordena_nombre_salida(salida,ultimo);                    // Ordena el archivo salida
    imprime_profesor(p,salida,semestre);                    // Imprime el archivo salida
    quita(tope);            // Quita ventana gráfica 157
    quita(tope);            // Quita ventana gráfica 156
  }
  farfree(salida);                                          // Libera el espacio reservado para salida
}

//    Busca la información de salón, para crear el archivo de impresión salida, según el valor de la variable tip.
void imprimir_salon_salida(struct MATERIA *p, char semestre[10], int b, int tip, long int numsalon,     int cuenta)
{
 struct MATERIA *r;
 struct SALIDA *salida;
 int i,k,ultimo;
                                                            // Reserva espacio en memoria para salida
 salida = (struct SALIDA *) farcalloc(cuenta+1,sizeof(struct SALIDA));
 if (!salida)
   {
    printf("Error.");
    exit(1);
   }
 salida[0].grupo = 0;
 salida[0].materia = 0;
 salida[0].numero = 0;
 r = p;
 pon(tope,352,COPY_PUT);            // Pone ventana gráfica 352
 texto("Confirme la impresión . . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
 pon(tope,253,COPY_PUT);            // Pone ventana gráfica 253
 pon(tope,254,COPY_PUT);            // Pone ventana gráfica 254
 texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
 texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
 pon(tope,253,NOT_PUT);             // Pone ventana gráfica del cursor
 i = opciones1(253,253,254,9);
 quita(tope);           // Quita ventana gráfica del cursor
 quita(tope);           // Quita ventana gráfica 254
 quita(tope);           // Quita ventana gráfica 253
 quita(tope);           // Quita ventana gráfica 352
 if (i == 254) then
  {
    pon(tope,352,COPY_PUT);             // Pone ventana gráfica 352
    texto("Desea imprimir el RFC. del profesor . . .",352,LIGHTRED,SANS_SERIF_FONT,1,2,2,2);
    pon(tope,253,COPY_PUT);             // Pone ventana gráfica 253
    pon(tope,254,COPY_PUT);             // Pone ventana gráfica 254
    texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
    texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
    i = -1;
    pon(tope,253,NOT_PUT);              // Pone ventana gráfica del cursor
    while (i == -1) do
        i = opciones1(253,253,254,9);
    quita(tope);            // Quita ventana gráfica del cursor
    quita(tope);            // Quita ventana gráfica 254
    quita(tope);            // Quita ventana gráfica 253
    quita(tope);            // Quita ventana gráfica 352
  pon(tope,156,COPY_PUT);               // Pone ventana gráfica 156
  cuadro(156,BLACK);
  pon(tope,157,COPY_PUT);               // Pone ventana gráfica 157
  texto("I M P R I M I E N D O . . .",157,DARKGRAY,TRIPLEX_SCR_FONT,1,2,2,2);
  switch(tip)
   {
          case 332: if (b == 0) then
                      busca_salon_tipo_imprime(r,numsalon,salida,1);
                    if (b == 1) then
                      busca_salon_cadena_tipo_imprime(r,numsalon,salida,1);
                    if (b == 2) then
                      busca_salon_todo_tipo_imprime(r,salida,1);
```

```
                    if (b == 3) then
                      busca_salon_materia_tipo_imprime(r,salida,1);
                    break;
          case 333: if (b == 0) then
                      busca_salon_tipo_imprime(r,numsalon,salida,2);
                    if (b == 1) then
                      busca_salon_cadena_tipo_imprime(r,numsalon,salida,2);
                    if (b == 2) then
                      busca_salon_todo_tipo_imprime(r,salida,2);
                    if (b == 3) then
                      busca_salon_materia_tipo_imprime(r,salida,2);
                    break;
          case 334: if (b == 0) then
                      busca_salon_tipo_imprime(r,numsalon,salida,3);
                    if (b == 1) then
                      busca_salon_cadena_tipo_imprime(r,numsalon,salida,3);
                    if (b == 2) then
                      busca_salon_todo_tipo_imprime(r,salida,3);
                    if (b == 3) then
                      busca_salon_materia_tipo_imprime(r,salida,3);
                    break;
          case 335: if (b == 0) then
                      busca_salon_tipo_imprime(r,numsalon,salida,4);
                    if (b == 1) then
                      busca_salon_cadena_tipo_imprime(r,numsalon,salida,4);
                    if (b == 2) then
                      busca_salon_todo_tipo_imprime(r,salida,4);
                    if (b == 3) then
                      busca_salon_materia_tipo_imprime(r,salida,4);
                    break;
          case 336: if (b == 0) then
                      busca_salon_imprime(r,numsalon,salida);
                    if (b == 1) then
                      busca_salon_cadena_todo_imprime(r,numsalon,salida);
                    if (b == 2) then
                      busca_salon_todo_imprime(r,salida);
                    if (b == 3) then
                      busca_salon_materia_todo_imprime(r,salida);
                    break;
      }
    ultimo = 0;
    for (k=0; salida[k].grupo != -2; k++)
            ultimo = ultimo + 1;
    ultimo = ultimo - 1;
    if (ultimo == -1) then
      ultimo = 0;
    ordena_salida(salida,ultimo);                    // Ordena el archivo salida
    imprime_salon(p,salida,l,semestre);     // Manda a impresión el archivo salida
    quita(tope);            // Quita ventana gráfica 157
    quita(tope);            // Quita ventana gráfica 156
  }
 farfree(salida);              // Libera el espacio reservado para salida
}
```

B.3.11 MENU.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\consulta.h"
#include "d:\grupos.h"
#include "d:\examen.h"
#include "d:\ventdos.h"
#include "d:\ayuda.h"
extern unsigned _stklen = 65519;
#define  then                                          // Definición de las notas
#define          DO     1
#define          RE     2
#define          MI     3
#define          FA     4
#define          SOL    5
#define          LA     6
#define          SI     7
#define          DO_    8
#define          RE_    9
#define          MI_    10
#define          FA_    11
#define          SOL_   12
#define          LA_    13
#define          SI_    14
unsigned frecuencia[20], tiempo;

//     Establece la frecuencia.
void inicio(void)
{
 int        i;

 frecuencia[0] = 40;
 for (i = 1; i <= 14; i++)
    frecuencia[i] = 100 + frecuencia[0]*i;
 tiempo = 1600;
}

// Establece el tipo de nota, el tiempo y la duración (bemol ó sostenido).
void F(int nota, double t, double tipo)
{
 unsigned           frec, c;
 frec = frecuencia[nota] + (unsigned) tipo*frecuencia[0];
 sound(frec);
 c = (unsigned ) tiempo*t;
 delay(c);
 nosound();
}
```

```
// Construye la melodía. "Pequeña Suite de Juan Sebastián Bach"
void suite(void)
{
 double              r;
 r = 1;
 F(RE_,r/4,0);
 F(SOL,r/8,0);
 F(LA,r/8,0);
 F(SI,r/8,0);
 F(DO_,r/8,0);

 F(RE_,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(SOL,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(SOL,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(MI_,r/4,0);
 F(DO_,r/8,0);
 F(RE_,r/8,0);
 F(MI_,r/8,0);
 F(FA_,r/8,r/2);

 F(SOL_,r/4,0);
 F(SOL,r/4,0);
 F(SOL,r/4,0);

 F(DO_,r/4,0);
 F(RE_,r/8,0);
 F(DO_,r/8,0);
 F(SI,r/8,0);
 F(LA,r/8,0);

 F(SI,r/4,0);
 F(DO_,r/8,0);
 F(SI,r/8,0);
 F(LA,r/8,0);
 F(SOL,r/8,0);
 // 1.
 F(FA,r/4,0);
 F(SOL,r/8,0);
 F(LA,r/8,0);
 F(SI,r/8,0);
 F(SOL,r/8,0);

 F(SI,r/4,0);
 F(LA,r/2,0);
 //:
 F(RE_,r/4,0);
 F(SOL,r/8,0);
 F(LA,r/8,0);
 F(SI,r/8,0);
 F(DO_,r/8,0);

 F(RE_,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(SOL,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(SOL,r/8,0);
 delay(unsigned (tiempo*r/8));
 F(MI_,r/4,0);
 F(DO_,r/8,0);
 F(RE_,r/8,0);
 F(MI_,r/8,0);
 F(FA_,r/8,r/2);
```

```
 F(SOL_,r/4,0);
 F(SOL,r/4,0);
 F(SOL,r/4,0);

 F(DO_,r/4,0);
 F(RE_,r/8,0);
 F(DO_,r/8,0);
 F(SI,r/8,0);
 F(LA,r/8,0);

 F(SI,r/4,0);
 F(DO_,r/8,0);
 F(SI,r/8,0);
 F(LA,r/8,0);
 F(SOL,r/8,0);
 // 2
 F(LA,r/4,0);
 F(SI,r/8,0);
 F(LA,r/8,0);
 F(SOL,r/8,0);
 F(FA,r/8,0);

 F(SOL,r/2,0);
 delay(unsigned (tiempo*r/4));
}

//   Llama a las funciones que construyen la melodía
void musica(void)
{
 inicio();
 suite();
 nosound();
}

//   Establece el nuevo período de clase
void pon_semestre(char semestre[10])
{
 int i,k;

 pon(tope,100,COPY_PUT);              // Pone ventana gráfica 100
 texto("CONSULTA DEL SEMESTRE",100,BLUE,SANS_SERIF_FONT,3,4,1,1);
 i = 0;
 while ((i != -1) && (i != 253)) do
  {                                   // Despliega en pantalla el semestre actual
   pon(tope,249,COPY_PUT);            // Pone ventana gráfica 249
   pon(tope,250,COPY_PUT);            // Pone ventana gráfica 250
   texto("Semestre actual ...",249,LIGHTRED,SMALL_FONT,3,2,2,1);
   texto(semestre,250,LIGHTRED,SMALL_FONT,3,2,2,1);
   pon(tope,252,COPY_PUT);            // Pone ventana gráfica  252
   pon(tope,253,COPY_PUT);            // Pone ventana gráfica  253
   pon(tope,254,COPY_PUT);            // Pone ventana gráfica  254
   texto("Desea modificar el semestre ...",252,LIGHTBLUE,SANS_SERIF_FONT,1,2,2,2);
   texto("NO",253,BLUE,SANS_SERIF_FONT,1,2,2,2);
   texto("SI",254,BLUE,SANS_SERIF_FONT,1,2,2,2);
   pon(tope,150,COPY_PUT);            // Pone ventana gráfica 150
   cuadro(150,BLUE);
   pon(tope,246,COPY_PUT);            // Pone ventana gráfica 246
   texto("Presione < ESC > para salir ...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
   pon(tope,255,NOT_PUT);             // Pone ventana gráfica del cursor
   i = opciones1(253,253,254,7);
   quita(tope);          // Quita ventana gráfica del cursor
   quita(tope);          // Quita ventana gráfica 246
   quita(tope);          // Quita ventana gráfica 150
   quita(tope);          // Quita ventana gráfica 254
   quita(tope);          // Quita ventana gráfica 253
```

```
    quita(tope);          // Quita ventana gráfica 352
    if (i == 254) then
     {                                  // Lee del teclado el nuevo semestre
      pon(tope,150,COPY_PUT);           // Pone ventana gráfica 150
      cuadro(150,BLUE);
      pon(tope,246,COPY_PUT);           // Pone ventana gráfica 246
      texto("Presione <ENTER> cuando termine de escribir el semestre ...",246,LIGHTBLUE,SMALL_FONT,3,2,2,1);
      pon(tope,251,COPY_PUT);           // Pone ventana gráfica 251
      pon(tope,252,COPY_PUT);           // Pone ventana gráfica 252
      texto("Nuevo semestre ...",251,LIGHTBLUE,SMALL_FONT,3,2,2,1);
      lee_cadena(semestre,5,252,LIGHTBLUE,SMALL_FONT,3,2,2,1);
      for (k=0; semestre[k]; k++)
            semestre[k] = toupper(semestre[k]);
      strcpy(semestre,semestre);
      quita(tope);                      // Quita ventana gráfica 252
      quita(tope);                      // Quita ventana gráfica 251
      quita(tope);                      // Quita ventana gráfica 246
      quita(tope);                      // Quita ventana gráfica 150
     }
    quita(tope);                        // Quita ventana gráfica 250
    quita(tope);                        // Quita ventana gráfica 249
   }
  quita(tope);                          // Quita ventana gráfica 100
}

int salir(int i, char semestre[10], struct MATERIA *lista)
{
 int j;
 j = 1;
 cuadro(61,GREEN);
 switch(i)
    {
    case 54: pon_semestre(semestre);    // Modificar el semestre
             break;
    case 55: escribir(lista,semestre,2); // Guarda información en archivo de texto
             break;
    case 56: j = 0;                     // Termina la ejecución del programa
             break;
    }
 cuadro(61,LIGHTBLUE);
 return(j);
}

//    Construye el menú CONSULTA
void ventanas_consulta(void)
{
 cuadro(37,GREEN);
 cuadrito(38,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"MATERIAS");
 cuadrito(39,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"GRUPOS");
 cuadrito(40,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"PROFESORES");
 cuadrito(41,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"SALONES");
 cuadrito(42,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"EXAMENES");
}

//    Construye el menú GRUPOS
void ventanas_grupos(void)
{
 cuadro(43,GREEN);
 cuadrito(44,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CREAR");
 cuadrito(45,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"AGREGAR");
 cuadrito(46,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"ELIMINAR");
 cuadrito(47,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CAMBIAR");
}
```

```
//   Construye el menú EXAMEN
void ventanas_examen(void)
{
 cuadro(49,GREEN);
 cuadrito(49,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CREAR");
 cuadrito(50,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"AGREGAR");
 cuadrito(51,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"ELIMINAR");
 cuadrito(52,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"CAMBIAR");
}

//   Construye el menú SALIR
void ventanas_salir(void)
{
 cuadro(53,GREEN);
 cuadrito(54,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"SEMESTRE");
 cuadrito(55,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"ASCII");
 cuadrito(56,LIGHTBLUE,SANS_SERIF_FONT,1,2,1,2,"SALIR");
}

//   Construye el menú principal
void ventanas_menu(void)
{
 int       c, i;

 cuadro(21,LIGHTBLUE);
 cuadrito(22,GREEN,TRIPLEX_FONT,1,1,1,1,"MATERIAS");
 cuadro(23,LIGHTBLUE);
 cuadrito(25,LIGHTRED,SMALL_FONT,2,1,2,1,"ARCHIVOS");
 cuadrito(26,LIGHTRED,SMALL_FONT,2,1,2,1,"CONSULTA");
 cuadrito(27,LIGHTRED,SMALL_FONT,2,1,2,1,"GRUPOS");
 cuadrito(28,LIGHTRED,SMALL_FONT,2,1,2,1,"EXAMENES");
 cuadrito(29,LIGHTRED,SMALL_FONT,2,1,2,1,"SALIDA");
}

//   Despliega el menú según el valor i
void mover(int i)
{
 switch(i)
   {
   case 25: pon(tope,30,COPY_PUT);          // Pone ventana gráfica 30
            ventanas_archivos();
            break;
   case 26: pon(tope,37,COPY_PUT);          // Pone ventana gráfica 37
            ventanas_consulta();
            break;
   case 27: pon(tope,43,COPY_PUT);          // Pone ventana gráfica 43
            ventanas_grupos();
            break;
   case 28: pon(tope,48,COPY_PUT);          // Pone ventana gráfica 48
            ventanas_examen();
            break;
   case 29: pon(tope,53,COPY_PUT);          // Pone ventana gráfica 53
            ventanas_salir();
            break;
   }
}

/*   Despliega el menú de opciones para la opción ARCHIVOS. Regresa -1 si no hubo elección, de lo contrario regresa
  la ventana de la opción seleccionada.*/
int opcion_archivos(void)
{
 int       i, c;

// Construye los cuadros de las ventanas
 i = 31;
```

```
c = DERECHA;
pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
while (c != ENTER) do
  {
   c = bioskey(0);
   switch(c)
        {
         case F1: llama_ayuda(6);       // Llama al módulo AYUDA
                  break;
                                        // Desplaza el cursor sobre el menú de opciones
         case ABAJO: quita(tope);       // Quita ventana gráfica del cursor
                  i++;
                  if (i == 37) then
                        i = 31;
                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                  break;
         case ARRIBA:  quita(tope);     // Quita ventana gráfica del cursor
                  i--;
                  if (i == 30) then
                        i = 36;
                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                  break;
         case ESC:  quita(tope);        // Quita ventana gráfica del cursor
                  return(-1);           // Sale de la opción ARCHIVOS
        }
  }
quita(tope);          // Quita ventana gráfica del cursor
return(i);
}

/*  Despliega el menú de opciones para la opción CONSULTA. Regresa -1 si no hubo elección, de lo contrario regresa
  la ventana de la opción seleccionada.*/
int opcion_consulta(void)
{
int     i, c;

// Construye los cuadros de las ventanas

i = 36;
c = DERECHA;
pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
while (c != ENTER) do
  {
   c = bioskey(0);
   switch(c)
        {
         case F1: llama_ayuda(57);      // Llama al módulo AYUDA
                  break;
                                        // Desplaza el cursor sobre el menú de opciones
         case ABAJO: quita(tope);       // Quita ventana gráfica del cursor
                  i++;
                  if (i == 43) then
                        i = 36;
                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                  break;
         case ARRIBA:  quita(tope);     // Quita ventana gráfica del cursor
                  i--;
                  if (i == 37) then
                        i = 42;
                  pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                  break;
         case ESC:  quita(tope);        // Quita ventana gráfica del cursor
                  return(-1);           // Sale de la opción CONSULTA
        }
  }
```

```
 quita(tope);          // Quita ventana gráfica del cursor
 return(i);
}

/*  Despliega el menú de opciones para la opción GRUPOS. Regresa -1 si no hubo elección, de lo contrario regresa la
  ventana de la opción seleccionada.*/
int opcion_grupos(void)
{
 int      i, c;

 // Construye los cuadros de las ventanas
 i = 44;
 c = DERECHA;
 pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
 while (c != ENTER) do
   {
    c = bioskey(0);
    switch(c)
         {
          case F1: llama_ayuda(58);      // Llama al módulo AYUDA
                   break;
                                         // Desplaza el cursor sobre el menú de opciones
          case ABAJO: quita(tope);       // Quita ventana gráfica del cursor
                   i++;
                   if (i == 48) then
                        i = 44;
                   pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                   break;
          case ARRIBA:  quita(tope);     // Quita ventana gráfica del cursor
                   i-;
                   if (i == 43) then
                        i = 47;
                   pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
                   break;
          case ESC:  quita(tope);        // Quita ventana gráfica
                   return(-1);           // Sale de la opción GRUPOS.
         }
   }
 quita(tope);                            // Quita ventana gráfica del cursor
 return(i);
}

/*  Despliega el menú de opciones para la opción EXAMEN. Regresa -1 si no hubo elección, de lo contrario regresa la
  ventana de la opción seleccionada.*/
int opcion_examen(void)
{
 int      i, c;

 // Construye los cuadros de las ventanas
 i = 49;
 c = DERECHA;
 pon(tope,i,NOT_PUT);                    // Pone ventana gráfica del cursor
 while (c != ENTER) do
   {
    c = bioskey(0);
    switch(c)
         {
          case F1: llama_ayuda(59);      // Llama al módulo AYUDA
                   break;
                                         // Desplaza el cursor sobre el menú de opciones
          case ABAJO: quita(tope);       // Quita ventana gráfica del cursor
                   i++;
                   if (i == 53) then
                        i = 49;
                   pon(tope,i,NOT_PUT);  // Pone ventana gráfica del cursor
```

```
                    break;
            case ARRIBA:  quita(tope);          // Quita ventana gráfica del cursor
                    i--;
                    if (i == 46) then
                        i = 52;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
            case ESC:   quita(tope);            // Quita ventana gráfica del cursor
                    return(-1);                 // Sale de la opción EXAMEN
          }
    }
 quita(tope);           // Quita ventana gráfica del cursor
 return(i);
}

/*   Despliega el menú de opciones para la opción Salir. Regresa -1 si no hubo elección, de lo contrario regresa la
  ventana de la opción seleccionada.*/
int opcion_salir(void)
{
 int        i, c;

 // construye los cuadros de las ventanas
 i = 54;
 c = DERECHA;
 pon(tope,i,NOT_PUT);                           // Pone ventana gráfica del cursor
 while (c != ENTER) do
   {
    c = bioskey(0);
    switch(c)
          {
          case F1: llama_ayuda(60);             // Llama al módulo AYUDA
                   break;
                                                // Desplaza el cursor sobre el menú de opciones
          case ABAJO: quita(tope);              // Quita ventana gráfica del cursor
                    i++;
                    if (i == 57) then
                        i = 54;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
          case ARRIBA:  quita(tope);            // Quita ventana gráfica del cursor
                    i--;
                    if (i == 53) then
                        i = 56;
                    pon(tope,i,NOT_PUT);        // Pone ventana gráfica del cursor
                    break;
          case ESC:   quita(tope);              // Quita ventana gráfica del cursor
                    return(-1);                 // Sale de la opción SALIR
          }
    }
 quita(tope);                   // Quita ventana gráfica del cursor
 return(i);
}

//   Despliega el menú de opciones principal. Regresa regresa la ventana de la opción seleccionada.
int menu_materias(int i, struct MATERIA *lista, char semestre[10])
{
 int        c, j, k, d;
                                        // Inicializa la lista de materias
 crea_lista_materias(lista);
                                        // Construye los cuadros de las ventanas
 j = 1;                                 // Bandera para salir
 while (j == 1) do
   {
    c = bioskey(0);
    switch(c)
```

```
{
case F1: llama_ayuda(5);                    // Llama al módulo AYUDA
         break;
                                            // Desplaza el cursor sobre el menú de opciones
case DERECHA: quita(tope);                  // Quita ventana gráfica del cursor
              quita(tope);                  // Quita ventana gráfica
              i++;
              if (i == 30) then
                   i = 25;
              pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
              mover(i);
              break;
case IZQUIERDA:  quita(tope);               // Quita ventana gráfica del cursor
              quita(tope);                  // Quita ventana gráfica
              i--;
              if (i == 24) then
                   i = 29;
              pon(tope,i,NOT_PUT);          // Pone ventana gráfica del cursor
              mover(i);
              break;
case ENTER:  k = 0;
              switch(i)
                   {                        // Opción archivos
                    case 25: while (k != (-1))
                                 {
                                  k = opcion_archivos();
                                  quita(tope);          // Quita ventana gráfica del cursor
                                  archivos(k,lista,semestre);
                                  mover(i);
                                 }
                                break;
                    case 26: d = 0;
                                while (k != (-1))
                                {       // Opción consulta
                                  k = opcion_consulta();
                                  quita(tope);          // Quita ventana gráfica
                                  d = consulta(k,lista,&d,semestre);
                                  mover(i);
                                 }
                                break;
                    case 27: while (k != (-1))
                                 {                      // Opción grupos
                                  k = opcion_grupos();
                                  quita(tope);          // Quita ventana gráfica
                                  grupos(k,lista);
                                  mover(i);
                                 }
                                break;
                    case 28: while (k != (-1))
                                 {                      // Opción exámen
                                  k = opcion_examen();
                                  quita(tope);          // Quita ventana gráfica del cursor
                                  examen(k,lista);
                                  mover(i);
                                 }
                                break;
                    case 29: while (k != (-1) && (j == 1))
                                 {                      // Opción salir
                                  k = opcion_salir();
                                  quita(tope);          // Quita ventana gráfica del cursor
                                  j = salir(k,semestre,lista);
                                  mover(i);
                                 }
                                break;
                   }
```

```
                    break;
            }
        }
    return(i);
}

// Construye la pantalla inicial.
void portada(void)
{
 cuadro(21,GREEN);
 cuadro(2,LIGHTBLUE);
 cuadro_texto("UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO",3,GREEN,COMPLEX_FONT,2,3,2,3);
 cuadro_texto("Escuela Nacional de Estudios Profesionales",4,LIGHTBLUE,SIMPLEX_FONT,2,3,2,3);
 cuadro_texto("Plantel ARAGON",5,LIGHTBLUE,SANS_SERIF_FONT,2,3,2,3);
 cuadro_texto("SISTEMA PARA LA ADMINISTRACION DE",6,MAGENTA,COMPLEX_FONT,1,2,1,2);
 cuadro_texto("LOS HORARIOS DE LA CARRERA DE",7,MAGENTA,COMPLEX_FONT,1,2,1,2);
 cuadro_texto("INGENIERIA MECANICA-ELECTRICA",8,MAGENTA,COMPLEX_FONT,1,2,1,2);
 rectangle(45,190,595,289);
 cuadro_texto("POR: YOLANDA CUEVAS SALGADO",9,BLUE,SANS_SERIF_FONT,1,2,1,2);
 cuadro_texto("MEXICO, 1994",10,LIGHTBLUE,SMALL_FONT,2,1,2,1);
 musica();                    // Llama a las funciones que construyen la melodia
 texto("Presione cualquier tecla para continuar . . .",11,DARKGRAY,TRIPLEX_SCR_FONT,2,3,2,3);
 bioskey(0);
}

//    Función principal
void main(void)
{
 struct MATERIA    q, lista;
 struct GRUPO              *grupito, g;
 int       drive, modo;
 int       i, n;
 char semestre[10];

 detectgraph(&drive,&modo); // Determina el manejador de gráficos y el modo
                            // gráfico para ser checado por el hardware.

 registerfarbgidriver(EGAVGA_driver_far);        // Liga el manejador de gráficos.
 registerfarbgifont(complex_font_far);           // Liga los fonts.
 registerfarbgifont(small_font_far);
 registerfarbgifont(sansserif_font_far);
 registerfarbgifont(simplex_font_far);
 registerfarbgifont(triplex_font_far);
 registerfarbgifont(triplex_scr_font_far);
 initgraph(&drive,&modo,"");                     // Inicializa el sistema de gráficos
 strcpy(semestre," ");                           // Inicializa la variable semestre
 tope[0].num = 0;                                // Inicializa la pila
 ventanas(vent);                                 // Funciones de ventanas
 vemidos(vent);
 portada();                                      // Pantalla inicial
 ventanas_menu();                                // Menú principal
 cuadro(61,LIGHTBLUE);
 i = 25;
 pon(tope,i,NOT_PUT);                            // Pone ventana gráfica del cursor
 mover(i);
 while (i != 29)
    {
    i = menu_materias(i,&lista,semestre);        // Función principal
    }
 quita(tope);          // Quita ventana gráfica del cursor
 quita(tope);          // Quita ventana gráfica del cursor
 closegraph();         // Termina el modo gráfico
}
```

B.3.12. OPER.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\archivos.h"
#include "d:\errores.h"

/*    Crea la lista enlazada MATERIA poniendo el primer nodo en la lista. Se inicializa poniendo
   el apuntador al siguiente nodo de la lista enlazada MATERIA a NULL, el apuntador a  la  lista
   enlazada GRUPO en NULL, el apuntador a la lista enlazada EXAMEN en NULL y poniendo los
   tipos de datos entero igual a cero. */
void crea_lista_materias(struct MATERIA *p)
{
  (*p).materias = NULL;                          // Apuntador al siguiente nodo
  (*p).grupos = NULL;                            // Apuntador a la lista GRUPO
  (*p).examen = NULL;                            // Apuntador a la lista EXAMEN
  (*p).lab = 0;                                  // Opción de laboratorio
  (*p).tipomat = 0;                              // Tipo de materia
  (*p).clave = 0;                                // Clave de la materia
  (*p).num = 0;                                  // Número de elementos en la lista
}

/*    Una vez que la lista enlazada MATERIA ha sido creada, ésta  función añade un nodo nuevo
   al final de la lista. */
void pon_lista_materias(struct MATERIA *p, struct MATERIA q)
{
 struct MATERIA  *r;

 if ((*p).num == 0) then
    {
    strcpy((*p).nombre,q.nombre);
    (*p).clave = q.clave;
    (*p).lab = q.lab;
    (*p).tipomat = q.tipomat;
    (*p).num = 1;
    (*p).grupos = q.grupos;
    (*p).examen = q.examen;
    return;
    }
 while ((*p).materias != NULL)
      p = (*p).materias;
                                                 // Reserva espacio en memoria
 r = (struct MATERIA *) farmalloc(sizeof(struct MATERIA));
 strcpy((*r).nombre,q.nombre);
 (*r).clave = q.clave;
 (*r).lab = q.lab;
 (*r).tipomat = q.tipomat;
 (*r).materias = NULL;
 (*r).grupos = q.grupos;
 (*r).examen = q.examen;
 (*r).num = (*p).num + 1;
 (*p).materias = r;
}
```

```
/*    Almacena en disco un archivo en modo texto. La información contenida en el archivo puede
      ser procesada en el Sistema Central de Administración Escolar de la UNAM. */
void guarda_lista_materias_texto(struct MATERIA *p, char *nombre)
{
 struct GRUPO *g;
 FILE     *archivo;
 int plantel,k;
 char numero[100];

 plantel = 411;
 archivo = fopen(nombre,"w");
 while (p != NULL)
  {
   itoa((*p).clave,numero,10);
   grupo_correcto(numero);
   g = (*p).grupos;
   while (g != NULL)
        {
         fprintf(archivo,"%d",plantel);                    // Clave del plantel
         fprintf(archivo,"%s",numero);                     // Clave de la materia
         itoa((*g).numgrupo,numero,10);
         grupo_correcto(numero);
         fprintf(archivo,"%s",numero);                     // Grupo
         itoa((*g).numalumnos,numero,10);
         grupo_correcto(numero);
         fprintf(archivo,"%s",numero);                     // Total de alumnos inscritos
         fprintf(archivo,"%s"," ");
         for (k=0; (*g).nombreprofe[k]; k++);
         while (k < 32)
          {
           strcat((*g).nombreprofe," ");
           k = k + 1;
          }
         fprintf(archivo,"%s",(*g).nombreprofe);           // Nombre del profesor
         for (k=0; (*g).rfc[k]; k++);
         while (k < 12)
          {
           strcat((*g).rfc," ");
           k = k + 1;
         }
         fprintf(archivo,"%s",(*g).rfc);                   // R.F.C. del profesor
         fprintf(archivo,"%s\n"," ");
         g = (*g).apun;
        }
   p = (*p).materias;
  }
 fclose(archivo);
}


/*    Almacena en disco un archivo en modo binario. Usa la trayectoria definida por nombre.
      Se inicializa almacenando en el archivo, la información contenida en la lista enlazada MATE-
      RIA, y al final se almacena la variable semestre.*/
void guarda_lista_materias(struct MATERIA *p, char *nombre, char semestre[10])
{
 struct GRUPO *r;
 struct EXAMEN *e;
 FILE     *archivo;
 int      c;

 archivo = fopen(nombre,"wb");
 while (p != NULL)
  {
   fwrite(p,sizeof(struct MATERIA),1,archivo);           // Escribe la lista MATERIA
   r = (*p).grupos;
```

```
    e = (*p).examen;
    while (r != NULL)
     {
          fwrite(r,sizeof(struct GRUPO),1,archivo);          // Escribe la lista GRUPO
          r = (*r).apun;
     }
    while (e != NULL)
     {
          fwrite(e,sizeof(struct EXAMEN),1,archivo);         // Escribe la lista EXAMEN
          e = (*e).sig;
     }
    p = (*p).materias;
   }
     fwrite(semestre,1,10,archivo);                          // Escribe el semestre
  fclose(archivo);
}


/*    Abre un archivo en modo binario para lectura.  Usa la trayectoria definida por nombre. Pone
  la información contenida en el archivo en las listas enlazadas MATERIA, GRUPO Y EXAMEN.
  Finaliza poniendo la  información correspondiente a la variable semestre. */
void lee_lista_materias(struct MATERIA *p, char *nombre, char semestre[10])
{
 struct MATERIA r, *s;
 struct GRUPO m;
 struct EXAMEN e;
 FILE     *archivo;

 archivo = fopen(nombre,"rb");
 while (!feof(archivo))
  {                                                          // Lee la lista MATERIA
   if (!(fread(&r,sizeof(struct MATERIA),1,archivo))) then
     break;
   pon_lista_materias(p,r);
   s = p;

   while ((*s).materias != NULL)
     s = (*s).materias;

   m.apun = (*s).grupos;
   e.sig = (*s).examen;
   while (m.apun != NULL)
    {                                                        // Lee la lista GRUPO
         if (!fread(&m,sizeof(struct GRUPO),1,archivo)) then
         break;
          if (m.num == 1) then
            {
             (*s).grupos = crea_lista_grupos();
             pon_grupos((*s).grupos,m);
            }
          else
            {
             pon_grupos((*s).grupos,m);
            }
    }
   while (e.sig != NULL)
    {                                                        // Lee la lista EXAMEN
         if (!fread(&e,sizeof(struct EXAMEN),1,archivo)) then
         break;
          if (e.num == 1) then
            {
             (*s).examen = crea_lista_examen();
             pon_examen((*s).examen,e);
            }
          else
```

```
            {
              pon_examen((*s).examen,e);
            }
      }
   if (r.materias == NULL) then
      fread(semestre,1,10,archivo);                    // Lee el semestre
  }
 fclose(archivo);
}


/*    Elimina un nodo de la lista enlazada MATERIA. El nodo puede ser eliminado en tres posicio-
  distintas: 1) cuando el nodo se encuentra al principio de la lista enlazada, 2) cuando el nodo se
  encuentra en medio de la lista enlazada y 3) cuando el nodo se encuentra al final de la lista en-
  lazada. La variable n es la posición del nodo en la lista.*/
int quita_lista_materias(struct MATERIA *p, int n)
{
struct MATERIA *r, *s;

r=p;
while ((*r).num != n)                                  // Mueve el apuntador r hasta la
        r=(*r).materias;                               // posición n.
if (p == r) then                                       // Cuando r es el primer nodo
  {
   if ((*r).materias == NULL) then
    {
        return(0);
    }
   else
    {
        r = (*p).materias;
        strcpy((*p).nombre,(*r).nombre);
        (*p).clave = (*r).clave;
        (*p).lab = (*r).lab;
        (*p).tipomat = (*r).tipomat;
        (*p).num = (*r).num;
        (*p).materias = (*r).materias;
        farfree(r);
        (*p).num = 2;
        while (p != NULL)
            {
             (*p).num = (*p).num - 1;
             p = (*p).materias;
            }
        return(-1);
    }
  }

if ((*r).materias == NULL) then                        // Cuando r es el último nodo
  {
   while ((*p).materias != r)
          p = (*p).materias;
   (*p).materias = NULL;
   farfree(r);
   return(-1);
  }
                                                       // Cuando r esta en medio
while ((*p).materias != r)
    p = (*p).materias;

s = (*r).materias;
(*p).materias = s;

while (s != NULL)
  {
```

```
    (*s).num = (*s).num -1;
    s = (*s).materias;
   }
  farfree(r);
  return(-1);
}


/*    Localiza la posición del nodo n en la lista enlazada, lee la nueva información y la coloca  en
  el nodo apuntado por la posición n.*/
int cambio_lista_materias(struct MATERIA *p, int n)
{
 struct MATERIA *r, s;

 r=p;
 while ((*r).num != n)                                  // Mueve el apuntador r hasta  la  posi-
       r=(*r).materias;                                 // ción n.
 lee_nodo_materia(&s,p);                                // Lee del teclado la nueva información
 strcpy((*r).nombre,s.nombre);                          // y la almacena en la posición apuntada
 (*r).clave=s.clave;                                    // apuntada por r.
 (*r).lab=s.lab;
 (*r).tipomat =s.tipomat;
 return(-1);
}


/*    Realiza una busqueda lineal sobre la lista enlazada MATERIA. El dato buscado (clave) es de
  tipo entero. Si se encuentra regresa la posición que ocupa en la lista, de lo contrario regresa el
  valor 0.*/
int busca_materia_clave(struct MATERIA *p, int clave)
{
 struct MATERIA *r;

 r = p;
 while (r != NULL)                                      // Mientras no sea fin de lista,  compara
  {                                                     // el dato contenido en el nodo con cla-
  if ((*r).clave == clave) then                         // clave. Si es igual  regresa  la  posición
        return((*r).num);                               // del nodo.
  r = (*r).materias;
  }
 if ( r == NULL) then                                   // Si no se encuentra regresa 0.
  return(0);
}


/*    Realiza una busqueda lineal sobre la lista enlazada MATERIA. El dato buscado (cadena)  es
  de tipo char. Si se encuentra regresa la posición que ocupa en la lista, de lo contrario  regresa
  el valor 0.*/
int busca_materia_nombre(struct MATERIA *p, char cadena[100])
{
 struct MATERIA *r;

 r = p;
 while (r != NULL)                                      // Mientras no sea fin de lista,  compara
  {                                                     // el dato contenido en el nodo con ca-
  if (!strcmp((*r).nombre,cadena)) then                 // dena. Si es igual regresa  la  posición
        return((*r).num);                               // del nodo.
  r = (*r).materias;
  }

 if ( r == NULL) then                                   // Si no se encuentra regresa 0.
  return(0);
}
```

```
/*    Crea la lista enlazada GRUPO poniendo el primer nodo en la lista. Se inicializa poniendo el
  apuntador al siguiente nodo de la lista enla zada GRUPO a NULL y poniendo los tipos de da-
  tos entero igual a cero.*/
struct GRUPO * crea_lista_grupos(void)
{
 struct GRUPO *r;
                                                    // Reserva espacio en memoria
 r = (struct GRUPO *) farmalloc(sizeof(struct GRUPO));
 (*r).apun=NULL;                                    // Apuntador al siguiente nodo.
 (*r).num = 0;                                      // Número de elementos en la lista.
 (*r).numgrupo = 0;                                 // Grupo.
 (*r).numsalon = 0;                                 // Salón.
 (*r).tipo = 0;                                     // Tipo de salón.
 (*r).tipoprofe = 0;                                // Nombramiento del profesor.
 (*r).numalumnos = 0;                               // Total de alumnos inscritos.
 (*r).clasegrupo = 0;                               // Tipo de grupo.
 return(r);
}


/*    Una vez que la lista enlazada GRUPO ha sido creada, ésta función añade un nodo nuevo al
  final de la lista. */
void pon_grupos(struct GRUPO *g, struct GRUPO q)
{
 struct GRUPO  *r;
 int k;

 if ((*g).num == 0) then
   {
    (*g).numgrupo = q.numgrupo;
    (*g).clasegrupo = q.clasegrupo;
    (*g).numsalon = q.numsalon;
    (*g).tipo = q.tipo;
    (*g).tipoprofe = q.tipoprofe;
    strcpy((*g).nombreprofe,q.nombreprofe);
    strcpy((*g).rfc,q.rfc);
    strcpy((*g).telofe,q.telofe);
    strcpy((*g).telcasa,q.telcasa);
    strcpy((*g).antunam,q.antunam);
    strcpy((*g).antsnep,q.antsnep);
    strcpy((*g).hora1,q.hora1);
    strcpy((*g).hora2,q.hora2);
    for (k=0; q.dias[k] <= 10; k++)
    (*g).dias[k] = q.dias[k];
    (*g).numalumnos = q.numalumnos;
    (*g).apun = NULL;
    (*g).num = 1;
    return;
   }
 while ((*g).apun != NULL)
        g = (*g).apun;
                                                    // Asigna espacio en memoria
 r = (struct GRUPO *) farmalloc(sizeof(struct GRUPO));

 (*r).numgrupo = q.numgrupo;
 (*r).clasegrupo = q.clasegrupo;
 (*r).numsalon = q.numsalon;
 (*r).tipo = q.tipo;
 (*r).tipoprofe = q.tipoprofe;
 strcpy((*r).nombreprofe,q.nombreprofe);
 strcpy((*r).rfc,q.rfc);
 strcpy((*r).telofe,q.telofe);
 strcpy((*r).telcasa,q.telcasa);
 strcpy((*r).antunam,q.antunam);
 strcpy((*r).antsnep,q.antsnep);
```

```
strcpy((*r).hora1,q.hora1);
strcpy((*r).hora2,q.hora2);
for (k=0; q.dias[k] <= 10; k++)
(*r).dias[k] = q.dias[k];
(*r).numalumnos = q.numalumnos;
(*r).apun = NULL;
(*r).num = (*g).num + 1;
(*g).apun = r;
}


/*    Elimina un nodo de la lista enlazada GRUPO. El nodo puede ser eliminado en tres posicio-
 nes distintas: 1) cuando el nodo se encuentra al principio de la lista enlazada, 2) cuando el nodo
 se encuentra en medio de la lista enlazada y 3) cuando el nodo se encuentra al final de la lista
 enlazada. La variable n es la posición del nodo en la lista.*/
int quita_grupos(struct GRUPO *g, int n)
{
struct GRUPO *r, *s;
int k;

r=g;
while ((*r).num != n)                          // Mueve el apuntador r hasta la posición
        r=(*r).apun;                           // n.

if (g == r) then                               // Cuando r es el primer nodo
  {
   if ((*r).apun == NULL) then
    {
        farfree(r);
        return(0);
    }
    else
       {
        r = (*g).apun;
        (*g).numgrupo = (*r).numgrupo;
        (*g).clasegrupo = (*r).clasegrupo;
        (*g).numsesion = (*r).numsesion;
        (*g).tipo = (*r).tipo;
        (*g).tipoprofe = (*r).tipoprofe;
        strcpy((*g).nombreprofe,(*r).nombreprofe);
        strcpy((*g).rfc,(*r).rfc);
        strcpy((*g).telofc,(*r).telofc);
        strcpy((*g).telcasa,(*r).telcasa);
        strcpy((*g).antunam,(*r).antunam);
        strcpy((*g).antsnep,(*r).antsnep);
        strcpy((*g).hora1,(*r).hora1);
        strcpy((*g).hora2,(*r).hora2);
        for (k=0; (*r).dias[k] <= 10; k++)
        (*g).dias[k] = (*r).dias[k];
        (*g).numalumnos = (*r).numalumnos;
        (*g).apun = (*r).apun;
        farfree(r);
        (*g).num = 2;

        while (g != NULL)
            {
                (*g).num = (*g).num - 1;
                g = (*g).apun;
            }
        return(-1);
       }
  }
if ((*r).apun == NULL) then                    // Cuando r es el último nodo.
  {
   while ((*g).apun != r)
```

```
            g = (*g).apun;
     (*g).apun = NULL;
     farfree(r);
     return(-1);
    }

 while ((*g).apun != r)                          // Cuando r esta en medio
     g = (*g).apun;

 s = (*r).apun;
 (*g).apun = s;
 while (s != NULL)
   {
    (*s).num = (*s).num -1;
    s = (*s).apun;
   }
   farfree(r);
 return(-1);
}


/*    Crea la lista enlazada EXAMEN poniendo el primer nodo en  la  lista. Se inicializa poniendo el apuntador al siguiente nodo de la lista enlazada EXAMEN a NULL y poniendo los  tipos de datos entero igual a cero.*/
struct EXAMEN * crea_lista_examen(void)
{
 struct EXAMEN *r;
                                                 // Reserva espacio en memoria.
 r = (struct EXAMEN *) farmalloc(sizeof(struct EXAMEN));
 (*r).sig=NULL;                                  // Apuntador al siguiente nodo.
 (*r).num = 0;                                   // Número de nodos en la lista.
 (*r).numgrupo = 0;                              // Grupo.
 (*r).numsalon = 0;                              // Salón.
 (*r).tipoesa = 0;                               // Examen final o extraordinario.
 (*r).tipoprofe = 0;                             // Nombre del prof. de exá. fin.
 (*r).tipo = 0;                                  // Nombr. del prof. de exá. extra.
 (*r).tiposalon = 0;                             // Clasificación del salón.
 (*r).dia1 = 0;                                  // Fecha de examen.
 (*r).dia2 = 0;
 (*r).mes1 = 0;
 (*r).mes2 = 0;
 (*r).ano1 = 0;
 (*r).ano2 = 0;
 return(r);
}


/*    Una vez que la lista enlazada EXAMEN ha sido creada, ésta función añade un nodo  nuevo al final de la lista. */
void pon_examen(struct EXAMEN *g, struct EXAMEN q)
{
 struct EXAMEN  *r;
 int k;

 if ((*g).num == 0) then
   {
    (*g).numgrupo = q.numgrupo;
    (*g).numsalon = q.numsalon;
    (*g).tipoesa = q.tipoesa;
    (*g).tipoprofe = q.tipoprofe;
    (*g).tipo = q.tipo;
    (*g).tiposalon = q.tiposalon;
    (*g).dia1 = q.dia1;
    (*g).dia2 = q.dia2;
    (*g).mes1 = q.mes1;
```

```
    (*g).mes2 = q.mes2;
    (*g).ano1 = q.ano1;
    (*g).ano2 = q.ano2;
    strcpy((*g).nombreprofe,q.nombreprofe);
    strcpy((*g).profe,q.profe);
    strcpy((*g).rfc,q.rfc);
    strcpy((*g).rfc2,q.rfc2);
    strcpy((*g).hora1,q.hora1);
    (*g).sig = NULL;
    (*g).num = 1;
    return;
  }
while ((*g).sig != NULL)
        g = (*g).sig;
                                                    // Reserva espacio en memoria.
r = (struct EXAMEN *) farmalloc(sizeof(struct EXAMEN));
(*r).numgrupo = q.numgrupo;
(*r).numsalon = q.numsalon;
(*r).tipoesa = q.tipoesa;
(*r).tipoprofe = q.tipoprofe;
(*r).tipo = q.tipo;
(*r).tipositon = q.tipositon;
(*r).dia1 = q.dia1;
(*r).dia2 = q.dia2;
(*r).mes1 = q.mes1;
(*r).mes2 = q.mes2;
(*r).ano1 = q.ano1;
(*r).ano2 = q.ano2;
strcpy((*r).nombreprofe,q.nombreprofe);
strcpy((*r).profe,q.profe);
strcpy((*r).rfc,q.rfc);
strcpy((*r).rfc2,q.rfc2);
strcpy((*r).hora1,q.hora1);
(*r).sig = NULL;
(*r).num = (*g).num + 1;
(*g).sig = r;
}

/*    Elimina un nodo de la lista enlazada EXAMEN. El nodo puede ser eliminado en  tres  posi-
  ciones distintas: 1) cuando el nodo se encuentra al principio de la lista  enlazada, 2) cuando el
  nodo se encuentra en  medio de la lista enlazada y 3) cuando el nodo se encuentra al final de
  la lista enlazada. La variable n es la posición del nodo en la lista.*/
int quita_examen(struct EXAMEN *g, int n)
{
 struct EXAMEN *r, *s;
 int k;

 r=g;
 while ((*r).num != n)                              // Mueve el apuntador r a la posición n.
        r=(*r).sig;

 if (g == r) then                                   // Cuando r es el primero nodo.
   {
    if ((*r).sig == NULL) then
     {
        farfree(r);
        return(0);
     }
    else
        {
        r = (*g).sig;
        (*g).numgrupo = (*r).numgrupo;
        (*g).numsalon = (*r).numsalon;
        (*g).tipoesa = (*r).tipoesa;
```

```
            (*g).tipoprofe = (*r).tipoprofe;
            (*g).tipo = (*r).tipo;
            (*g).tipopasion = (*r).tipopasion;
            (*g).dia1 = (*r).dia1;
            (*g).dia2 = (*r).dia2;
            (*g).mes1 = (*r).mes1;
            (*g).mes2 = (*r).mes2;
            (*g).ano1 = (*r).ano1;
            (*g).ano2 = (*r).ano2;
            strcpy((*g).nombreprofe,(*r).nombreprofe);
            strcpy((*g).profe,(*r).profe);
            strcpy((*g).rfc,(*r).rfc);
            strcpy((*g).rfc2,(*r).rfc2);
            strcpy((*g).hora1,(*r).hora1);
            (*g).sig = (*r).sig;
            farfree(r);
            (*g).num = 2;
            while (g != NULL)
                {
                    (*g).num = (*g).num - 1;
                    g = (*g).sig;
                }
            return(-1);
           }
    }
 if ((*r).sig == NULL) then                          // Cuando r es el último nodo.
    {
      while ((*g).sig != r)
              g = (*g).sig;

      (*g).sig = NULL;
      farfree(r);
      return(-1);
    }

 while ((*g).sig != r)                               // Cuando r esta en medio.
      g = (*g).sig;

 s = (*r).sig;
 (*g).sig = s;
 while (s != NULL)
    {
      (*s).num = (*s).num -1;
      s = (*s).sig;
    }

    farfree(r);

 return(-1);
}
```

B.3.13. P_TODOS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\oper.h"
#include "d:\cuadros.h"
#include "d:\grafico.h"
#include "d:\archivos.h"
#include "d:\grupos.h"
#include "d:\g_todos.h"
#include "d:\p_todos.h"
#include "d:\consulta.h"
#include "d:\errores.h"
#include "d:\ayuda.h"

#define  then


//   Construye ventanas para consulta de profesores
void cuadros_profesor_todos(int tipo)
{
 texto("RFC",318,LIGHTBLUE,SMALL_FONT,3,2,2,1);

 switch(tipo)
  {
        case 342: texto("PROFESORES TITULARES",319,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                  break;
        case 343: texto("AYUDANTES DE PROFESOR",319,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                  break;
        case 344: texto("PROFESORES SIN NOMBRAMIENTO",319,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                  break;
        case 345: texto("PROFESORES",319,LIGHTBLUE,SMALL_FONT,3,2,2,1);
                  break;
  }
 texto("PAG. 1",340,BLUE,SANS_SERIF_FONT,1,2,1,2);
 cuadro(160,BLUE);
 texto("Seleccione un profesor para obtener más información",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
 texto("<TAB> cambia menú",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
 texto("PgUp/PgDn",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
}

//   Busca profesor en la lista enlazada GRUPO.
int busca_profesor_arreglo(struct MATERIA *p, struct ARREGLO *arreglo, char profesor[100], int j)
{
 struct GRUPO *g;
 struct MATERIA *r, *s;
 int a,k;
 char cadena[100];

 r = p;

 for (k=0, a=0; k <= (j - 1) && a != 1; k++)
 {
   s = busca_materia(r,arreglo[k].materia);
```

```
    g = (*s).grupos;
    while ((*g).num != arreglo[k].grupo) do
            g = (*g).apun;

    if (!strcmp((*g).nombreprofe,profesor)) then
          a = 1;
  }
 return(a);
}

//    Busca la cadena profesor, del tipo de profesor definido por la variable clase en la lista enlazada GRUPO.
void busca_profe_tipo(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).tipoprofe == clase) then
            {
             if (!strcmp((*s).nombreprofe,profesor)) then
              {
               if (j > 0) then
                     a = busca_profesor_arreglo(g,arreglo,(*s).nombreprofe,j);
               if (a == 0) then
                {
                      arreglo[j].grupo = (*s).num;
                      arreglo[j].materia = (*r).num;
                      arreglo[j].numero = (*s).numgrupo;
                      strcpy(arreglo[j].nombre,(*s).nombreprofe);
                      j = j + 1;
                }
              }
            }
     s = (*s).apun;
     }
  r = (*r).materias;
  }
 arreglo[j].grupo = -2;
 arreglo[j].materia = -2;
 arreglo[j].numero = -2;
 strcpy(arreglo[j].nombre,"-2");
}

//    Busca la cadena profesor, del tipo de profesor definido por la variable clase en la lista enlazada GRUPO.
void busca_profe_cadena_tipo(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int k,n,j,dato,a;
 char cadena[100];

 r = p;
 g = p;
 a = 0;
 k = strlen(profesor);
 j = 0;
```

```
while (r != NULL) do
 {
 s = (*r).grupos;
   while (s != NULL) do
    {
    if ((*s).tipoprofe == clase) then
          {
           dato = 1;
           strcpy(cadena,(*s).nombreprofe);
           for (n=0; n < k && dato == 1; n++)
            {
             if (profesor[n] == cadena[n]) then
               dato =1;
             else
               dato = 0;
            }
           if (dato == 1) then
            {
             if (j > 0) then
                   a = busca_profesor_arreglo(g,arreglo,(*s).nombreprofe,j);
             if (a == 0) then
              {
                   arreglo[i].grupo = (*s).num;
                   arreglo[i].materia = (*r).num;
                   arreglo[i].numero = (*s).numgrupo;
                   strcpy(arreglo[i].nombre,(*s).nombreprofe);
                   i = i + 1;
              }
            }
          }
    s = (*s).apun;
    }
 r = (*r).materias;
 }
 arreglo[i].grupo = -2;
 arreglo[i].materia = -2;
 arreglo[i].numero = -2;
 strcpy(arreglo[i].nombre,"-2");
}

//     Busca datos de profesor, del tipo de profesor definido por la variable clase en la lista enlazada GRUPO.
void busca_profe_todos_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int i,a;

 r = p;
 g = p;
 a = 0;
 i = 0;
 while (r != NULL) do
  {
  s = (*r).grupos;
    while (s != NULL) do
     {
     if ((*s).tipoprofe == clase) then
           {
            if (j > 0) then
              a = busca_profesor_arreglo(g,arreglo,(*s).nombreprofe,j);
            if (a == 0) then
              {
                   arreglo[i].grupo = (*s).num;
                   arreglo[i].materia = (*r).num;
                   arreglo[i].numero = (*s).numgrupo;
```

```
                    strcpy(arreglo[].nombre,(*s).nombreprofe);
                    j = j + 1;
                  }
                }
              s = (*s).apun;
             }
            r = (*r).materias;
           }
           arreglo[].grupo = -2;
           arreglo[].materia = -2;
           arreglo[].numero = -2;
           strcpy(arreglo[].nombre,"-2");
}

//     Busca datos de profesor, del tipo de profesor definido por la variable clase en la lista enlazada GRUPO.
void busca_profe_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j,a;

 r = p;
 a = 0;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).tipoprofe == clase) then
           {
            if (j > 0) then
              a = busca_profesor_arreglo(r,arreglo,(*s).nombreprofe,j);
            if (a == 0) then
             {
                    arreglo[].grupo = (*s).num;
                    arreglo[].materia = (*r).num;
                    arreglo[].numero = (*s).numgrupo;
                    strcpy(arreglo[].nombre,(*s).nombreprofe);
                    j = j + 1;
             }
           }
      s = (*s).apun;
     }
 arreglo[].grupo = -2;
 arreglo[].materia = -2;
 arreglo[].numero = -2;
 strcpy(arreglo[].nombre,"-2");
}

//     Busca la cadena profesor en la lista enlazada GRUPO.
void busca_profe(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int j,a;

 r = p;
 g = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
           if (!strcmp((*s).nombreprofe,profesor)) then
```

```
            {
             if (j > 0) then
                    a = busca_profesor_arreglo(g,arreglo,(*s).nombreprofe,j);
             if (a == 0) then
              {
                    arreglo[j].grupo = (*s).num;
                    arreglo[j].materia = (*r).num;
                    arreglo[j].numero = (*s).numgrupo;
                    strcpy(arreglo[j].nombre,(*s).nombreprofe);
                    j = j + 1;
              }
             }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
  strcpy(arreglo[j].nombre,"-2");
}

//     Busca la cadena profesor en la lista enlazada GRUPO.
void busca_profe_cadena(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo)
{
 struct MATERIA *r,*g;
 struct GRUPO *s;
 int k,n,j,dato,a;
 char cadena[100];

 r = p;
 g = p;
 a = 0;
 k = strlen(profesor);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            strcpy(cadena,(*s).nombreprofe);
            for (n=0; n < k && dato == 1; n++)
              {
               if (profesor[n] == cadena[n]) then
                 dato =1;
               else
                 dato = 0;
              }
            if (dato == 1) then
              {
               if (j > 0) then
                    a = busca_profesor_arreglo(g,arreglo,(*s).nombreprofe,j);
               if (a == 0) then
                {
                    arreglo[j].grupo = (*s).num;
                    arreglo[j].materia = (*r).num;
                    arreglo[j].numero = (*s).numgrupo;
                    strcpy(arreglo[j].nombre,(*s).nombreprofe);
                    j = j + 1;
                }
              }
      s = (*s).apun;
     }
  r = (*r).materias;
```

```
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
  strcpy(arreglo[j].nombre,"-2");
}

//   Pone toda la información de profesor en el ARREGLO
void busca_profe_todos(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r,*q;
 struct GRUPO *s;
 int j,a;

 r = p;
 q = p;
 a = 0;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
          if (j > 0) then
             a = busca_profesor_arreglo(q,arreglo,(*s).nombreprofe,j);
          if (a == 0) then
            {
              arreglo[j].grupo = (*s).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*s).numgrupo;
              strcpy(arreglo[j].nombre,(*s).nombreprofe);
              j = j + 1;
            }
      s = (*s).apun;
     }
  r = (*r).materias;
  }
  arreglo[j].grupo = -2;
  arreglo[j].materia = -2;
  arreglo[j].numero = -2;
  strcpy(arreglo[j].nombre,"-2");
}

//   Pone la información de profesor de una materia
void busca_profe_todos_materia(struct MATERIA *p, struct ARREGLO *arreglo)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j,a;

 r = p;
 a = 0;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
          if (j > 0) then
             a = busca_profesor_arreglo(r,arreglo,(*s).nombreprofe,j);
          if (a == 0) then
            {
              arreglo[j].grupo = (*s).num;
              arreglo[j].materia = (*r).num;
              arreglo[j].numero = (*s).tipoprofe;
              strcpy(arreglo[j].nombre,(*s).nombreprofe);
              j = j + 1;
```

```
            }
        s = (*s).apun;
       }
  arreglo[].grupo = -2;
  arreglo[].materia = -2;
  arreglo[].numero = -2;
  strcpy(arreglo[].nombre,"-2");
}

//  Cuenta el número de profesores para el tamaño del arreglo ARREGLO
void busca_profe_tipo_cuenta(struct MATERIA *p, char profesor[100], int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipoprofe == clase) then
             {
              if (!strcmp((*s).nombreprofe,profesor)) then
                     j = j + 1;
             }
       s = (*s).apun;
      }
   r = (*r).materias;
  }
 *cuenta = j;
}

/*    Busca la cadena profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable clase.
  Cuenta el número de profesores para el tamaño del arreglo ARREGLO.*/
void busca_profe_cadena_tipo_cuenta(struct MATERIA *p, char profesor[100], int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int k,n,j,dato;
 char cadena[100];

 r = p;
 k = strlen(profesor);
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
     {
      if  ((*s).tipoprofe == clase) then
            {
             dato = 1;
             strcpy(cadena,(*s).nombreprofe);
             for (n=0; n < k && dato == 1; n++)
              {
               if (profesor[n] == cadena[n]) then
                 dato =1;
               else
                 dato = 0;
              }
```

```
            if (dato == t) then
                        j = j + 1;
                }
        s = (*s).apun;
       }
    r = (*r).materias;
   }
   *cuenta = j;
}

/*    Busca la información de profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable
  clase. Cuenta el número de profesores para el tamaño del arreglo ARREGLO. */
void busca_profe_todos_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipoprofe == clase) then
                        j = j + 1;
       s = (*s).apun;
      }
   r = (*r).materias;
  }
  *cuenta = j;
}

/*    Busca la información de profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable
  clase. Cuenta el número de profesores para el tamaño del arreglo ARREGLO.*/
void busca_profe_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
     while (s != NULL) do
      {
       if  ((*s).tipoprofe == clase) then
                        j = j + 1;

       s = (*s).apun;
      }
    *cuenta = j;
}

//    Cuenta el número de profesores para el tamaño del arreglo ARREGLO.
void busca_profe_cuenta(struct MATERIA *p, char profesor[100], int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;
```

```
 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            if (!strcmp((*s).nombreprofe,profesor)) then
                    j = j + 1;
     s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

/*    Busca la cadena profesor en la lista enlazada GRUPO. Cuenta el número de profesores para el tamaño del
  arreglo ARREGLO.*/
void busca_profe_cadena_cuenta(struct MATERIA *p, char profesor[100], int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int k,n,j,dato;
 char cadena[100];

 r = p;
 k = strlen(profesor);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            strcpy(cadena,(*s).nombreprofe);
            for (n=0; n < k && dato == 1; n++)
             {
              if (profesor[n] == cadena[n]) then
                dato =1;
              else
                dato = 0;
             }
            if (dato == 1) then
                    j = j + 1;
     s = (*s).apun;
     }
   r = (*r).materias;
  }
  *cuenta = j;
}

//  Cuenta el número de profesores para el tamaño del arreglo ARREGLO.
void busca_profe_todos_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
```

```
    while (s != NULL) do
    {
            j = j + 1;
     s = (*s).apun;
    }
  r = (*r).materias;
 }
 *cuenta = j;
}

// Cuenta el número de profesores para el tamaño del arreglo ARREGLO.
void busca_profe_todos_materia_cuenta(struct MATERIA *p, int *cuenta)
{
 struct MATERIA *r;
 struct GRUPO *s;
 struct ARREGLO *arreglo;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
    {
     j = j + 1;
     s = (*s).apun;
    }
   *cuenta = j;
}

//    Busca la cadena profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable clase.
void busca_profe_tipo_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
 {
  s = (*r).grupos;
    while (s != NULL) do
    {
     if  ((*s).tipoprofe == clase) then
          {
           if (!strcmp((*s).nombreprofe,profesor)) then
           {
                  salida[j].grupo = (*s).num;
                  salida[j].materia = (*r).num;
                  salida[j].numero = (*s).numgrupo;
                  strcpy(salida[j].nombre,(*s).nombreprofe);
                  j = j + 1;
           }
          }
     s = (*s).apun;
    }
  r = (*r).materias;
 }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
 strcpy(salida[j].nombre,"-2");
}
```

```
//      Busca la cadena profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable clase.
void busca_profe_cadena_tipo_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char cadena[100];

 r = p;
 k = strlen(profesor);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).tipoprofe == clase) then
            {
             dato = 1;
             strcpy(cadena,(*s).nombreprofe);
             for (n=0; n < k && dato == 1; n++)
              {
               if (profesor[n] == cadena[n]) then
                 dato =1;
               else
                 dato = 0;
              }
             if (dato == 1) then
              {
                       salida[j].grupo = (*s).num;
                       salida[j].materia = (*r).num;
                       salida[j].numero = (*s).numgrupo;
                       strcpy(salida[j].nombre,(*s).nombreprofe);
                       j = j + 1;
              }
            }
     s = (*s).apun;
     }
  r = (*r).materias;
  }

  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
  strcpy(salida[j].nombre,"-2");
}

/*    Busca la información de profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable
   clase. */
void busca_profe_todos_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).tipoprofe == clase) then
            {
                     salida[j].grupo = (*s).num;
```

```
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    strcpy(salida[j].nombre,(*s).nombreprofe);
                    j = j + 1;
                }
            s = (*s).apun;
            }
        r = (*r).materias;
        }
        salida[j].grupo = -2;
        salida[j].materia = -2;
        salida[j].numero = -2;
        strcpy(salida[j].nombre,"-2");
}

/*    Busca la información de profesor en la lista enlazada GRUPO, según el tipo de profesor definido por la variable
  clase.*/
void busca_profe_materia_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
      if ((*s).tipoprofe == clase) then
            {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    strcpy(salida[j].nombre,(*s).nombreprofe);
                    j = j + 1;
            }
      s = (*s).apun;
     }
  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
  strcpy(salida[j].nombre,"-2");
}

//    Busca la información de profesor en la lista enlazada GRUPO.
void busca_profe_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
          if (!strcmp((*s).nombreprofe,profesor)) then
           {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    strcpy(salida[j].nombre,(*s).nombreprofe);
                    j = j + 1;
```

```
            }
      s = (*s).apun;
     }
   r = (*r).materias;
  }
  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
  strcpy(salida[j].nombre,"-2");
}

//    Busca la información de profesor en la lista enlazada GRUPO.
void busca_profe_cadena_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int k,n,j,dato;
 char cadena[100];

 r = p;
 k = strlen(profesor);
 j = 0;
 while (r != NULL) do
  {
   s = (*r).grupos;
    while (s != NULL) do
     {
            dato = 1;
            strcpy(cadena,(*s).nombreprofe);
            for (n=0; n < k && dato == 1; n++)
             {
              if (profesor[n] == cadena[n]) then
                dato =1;
              else
                dato = 0;
             }
            if (dato == 1) then
              {
                    salida[j].grupo = (*s).num;
                    salida[j].materia = (*r).num;
                    salida[j].numero = (*s).numgrupo;
                    strcpy(salida[j].nombre,(*s).nombreprofe);
                    j = j + 1;
              }
     s = (*s).apun;
     }
   r = (*r).materias;
  }
  salida[j].grupo = -2;
  salida[j].materia = -2;
  salida[j].numero = -2;
  strcpy(salida[j].nombre,"-2");
}

//    Busca la información de profesor en la lista enlazada GRUPO.
void busca_profe_todos_imprime(struct MATERIA *p, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 while (r != NULL) do
  {
```

```
  s = (*r).grupos;
   while (s != NULL) do
    {
              salida[j].grupo = (*s).num;
              salida[j].materia = (*r).num;
              salida[j].numero = (*s).numgrupo;
              strcpy(salida[j].nombre,(*s).nombreprofe);
              j = j + 1;
     s = (*s).apun;
    }
  r = (*r).materias;
 }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
 strcpy(salida[j].nombre,"-2");
}

//    Busca la información de profesor en la lista enlazada GRUPO.
void busca_profe_todos_materia_imprime(struct MATERIA *p, struct SALIDA *salida)
{
 struct MATERIA *r;
 struct GRUPO *s;
 int j;

 r = p;
 j = 0;
 s = (*r).grupos;
    while (s != NULL) do
     {
              salida[j].grupo = (*s).num;
              salida[j].materia = (*r).num;
              salida[j].numero = (*s).numgrupo;
              strcpy(salida[j].nombre,(*s).nombreprofe);
              j = j + 1;
     s = (*s).apun;
     }
 salida[j].grupo = -2;
 salida[j].materia = -2;
 salida[j].numero = -2;
 strcpy(salida[j].nombre,"-2");
}

/*    Despliega en pantalla la información de profesor de la lista enlazada GRUPO. Regresa la posición de la última
  ventana colocada.*/
int ventanas_profesor_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo, int *max, int tipo)
{
 struct MATERIA *r,*s;
 struct GRUPO *g;
 int i,j,k,c;
 char cadena[100];

 i = 250;
 j = *max;
 r = p;
 for (k=inicio; k <= fin; k++)
  {
   s = busca_materia(r,arreglo[k].materia);
   g = (*s).grupos;
   while ((*g).num != arreglo[k].grupo) do
              g = (*g).apun;
   if (g != NULL) then
    {
     cuadro(j,CYAN);
     pon(tapa,i,COPY_PUT);          // Pone ventana gráfica para i
```

```
        texto((*g).rfc,l,RED,SMALL_FONT,3,2,2,1);
        l = l+1;
        pon(tope,l,COPY_PUT);          // Pone ventana gráfica para l
        if (tipo == 346) then
              {
               strcpy(cadena,(*g).nombreprofe);
               c = (*g).tipoprofe;
               switch(c)
                {
                      case 1: strcat(cadena,"   Titular ");
                              break;
                      case 2: strcat(cadena,"   Ayudante");
                              break;
                      case 3: strcat(cadena,"   Sin norm.");
                              break;
                }
               texto(cadena,l,RED,SMALL_FONT,2,2,2,1);
              }
        else
              texto((*g).nombreprofe,l,RED,SMALL_FONT,3,2,2,1);
        l = l+1;
        *max = l;
        j = j+1;
       }
    }
  return(i);
}

/*    Despliega los nodos para profesor de la lista enlazada GRUPO. Regresa la posición del nodo en la lista enlazada
  GRUPO.*/
int profesor_todos(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int b, int tipo, int cuenta)
{
 struct MATERIA *r;

 struct POS
   {
    int inicio;
    int fin;
   };
 struct POS *pos;
 int k,l,regresa,numc,c,ultimo,grupo,cuot,rem,j,a,inicio,fin,dato,max;
 char sig[20],cad[20];

 // Reserva espacio en memoria para pos
 pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
 if (!pos)
   {
    printf("Error.");
    exit(1);
   }
 arreglo[0].grupo = 0;
 arreglo[0].materia = 0;
 arreglo[0].numero = 0;
 r = p;
 switch(tipo)
  {
        case 342: if (b == 0) then
                    busca_profe_tipo(r,profesor,arreglo,1);
                  if (b == 1) then
                    busca_profe_cadena_tipo(r,profesor,arreglo,1);
                  if (b == 2) then
                    busca_profe_todos_tipo(r,arreglo,1);
                  if (b == 3) then
                    busca_profe_materia_tipo(r,arreglo,1);
                  break;
```

```
        case 343: if (b == 0) then
                    busca_profe_tipo(r,profesor,arreglo,2);
                  if (b == 1) then
                    busca_profe_cadena_tipo(r,profesor,arreglo,2);
                  if (b == 2) then
                    busca_profe_todos_tipo(r,arreglo,2);
                  if (b == 3) then
                    busca_profe_materia_tipo(r,arreglo,2);
                  break;
        case 344: if (b == 0) then
                    busca_profe_tipo(r,profesor,arreglo,3);
                  if (b == 1) then
                    busca_profe_cadena_tipo(r,profesor,arreglo,3);
                  if (b == 2) then
                    busca_profe_todos_tipo(r,arreglo,3);
                  if (b == 3) then
                    busca_profe_materia_tipo(r,arreglo,3);
                  break;
        case 345: if (b == 0) then
                    busca_profe(r,profesor,arreglo);
                  if (b == 1) then
                    busca_profe_cadena(r,profesor,arreglo);
                  if (b == 2) then
                    busca_profe_todos(r,arreglo);
                  if (b == 3) then
                    busca_profe_todos_materia(r,arreglo);
                  break;
  }
ultimo = 0;
for (k=0; arreglo[k].grupo != -2; k++)
    ultimo = ultimo + 1;
quot = (ultimo/3);
rem = (ultimo%3);
ultimo = ultimo - 1;
if (ultimo == -1) then
  ultimo = 0;
ordena_nombre_arreglo(arreglo,ultimo);                    // Ordena la información antes de desplegarla en pantalla
if (rem != 0) then
  grupo = quot + 1;
else
  grupo = quot;
if (grupo == 0) then
 {
  llama_error(17);
  return(-2);
 }
k = 0;
for (j=1; j <= grupo; j++)
  {
   for (a=0; arreglo[k].grupo != -2 && a < 3; k++, a++)
         {
          fin = k;
          if (a == 0) then
            inicio = k;
         }
   pos[j].inicio = inicio;
   pos[j].fin = fin;
  }
cuadros_profesor_todos(tipo);                    // Construye menú inferior
j = 1;
max = 122;
inicio = pos[j].inicio;
fin = pos[j].fin;
numc = ventanas_profesor_todos(p,inicio,fin,arreglo,&max,tipo);
i = 122;
```

```
c=END;
regresa=0;
pon(tope,l,NOT_PUT);                    // Pone ventana gráfica del cursor
while (c != ENTER) do
{
 c=bioskey(0);
 switch(c)
 {
   case F1: llama_ayuda(53);            // Llama al módulo AYUDA
            break;
                                        // Desplaza el cursor sobre las ventanas de profesor
       case ABAJO:  quita(tope);        // Quita ventana gráfica del cursor
                    l++;
                    if (l == (max + 1)) then
                            l = 122;
                    pon(tope,l,NOT_PUT);    // Pone ventana gráfica del cursor
                    break;

       case ARRIBA: quita(tope);        // Quita ventana gráfica del cursor
                    l--;
                    if (l == (122 - 1)) then
                            l = max;
                    pon(tope,l,NOT_PUT);    // Pone ventana gráfica del cursor
                    break;

       case TAB:    if ((l >= 122) && (l <= max)) then          // Cambia al menú inferior
                    {
                     regresa=l;
                     l=237;
                     l=mover_ventana_menu(l,l,237,regresa);
                     if (l == 237) then
                            {
                             c = ENTER;
                             dato = -2;
                            }
                    }
                    break;
                                        // Cambia a la siguiente página
       case PGDN:   quita(tope);        // Quita ventana gráfica  del cursor
                    numo--;
                    for (k=259; k <= numc; k++)
                            quita(tope);        // Quita ventanas gráficas restantes

                    for (k = 122; k <= max; k++)
                           {
                            setfillstyle(SOLID_FILL,GREEN);
                            bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                           }
                    j++;
                    if (j == (grupo+1)) then
                     j=1;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    max=122;
                    numc = ventanas_profesor_todos(p,inicio,fin,arreglo,&max,tipo);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                    if (l > max) then
                    {
                     l = max;
                     pon(tope,l,NOT_PUT);               // Pone ventana gráfica del cursor
                    }
                    else
```

```
                    pon(tope,l,NOT_PUT);              // Pone ventana gráfica del cursor
                    break;
                                                      // Cambia a la página anterior
          case PGUP:  quita(tope);                    // Quita ventana gráfica  del cursor
                    numc-;
                    for (k=259; k <= numc; k++)
                          quita(tope);                // Quita ventanas gráficas restantes

                    for (k = 122; k <= max; k++)
                          {
                           setfillstyle(SOLID_FILL,GREEN);
                           bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
                          }
                    l-;
                    if (l == 0) then
                      l=grupo;
                    inicio=pos[l].inicio;
                    fin=pos[l].fin;
                    max=122;
                    numc = ventanas_profesor_todos(p,inicio,fin,arreglo,&max,tipo);
                    strcpy(sig,"PAG. ");
                    itoa(l,cad,10);
                    strcat(sig,cad);
                    texto(sig,240,BLUE,SANS_SERIF_FONT,1,2,1,2);
                    if (l > max) then
                     {
                      l = max;
                      pon(tope,l,NOT_PUT);        // Pone ventana gráfica del cursor
                     }
                    else
                     pon(tope,l,NOT_PUT);         // Pone ventana gráfica del cursor
                    break;
  }
 }
 quita(tope);        // Quita ventana gráfica  del cursor
 numc-;
 for (k=259; k <= numc; k++)
   quita(tope);      // Quita ventanas gráficas restantes
 for (k = 316; k <= 319; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 for (k = 122; k <= max; k++)
    {
     setfillstyle(SOLID_FILL,GREEN);
     bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
    }
 texto("MATERIA SELECCIONADA",240,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 if (dato != -2) then
  {
   dato = l - 122;
   a = pos[l].inicio;
   a = a + dato;
  }
 else
  a = dato;
 farfree(pos);                                    // Libera el espacio reservado para pos
 return(a);
}

//  Despliega por páginas los datos de profesor.
int mover_datos_profesor_todos(struct GRUPO *g, char profesor[100], int cuenta)
{
 struct GRUPO *r,*s;
```

```
struct POS{
            int inicio;
            int fin;
           };
struct POS *pos;
int c,inicio,fin,grupo,ultimo,j,quot,rem,numc,k,a,data;
char sig[20], ced[20];
                        // Reserva espacio en memoria para pos
pos = (struct POS *) farcalloc(cuenta+1,sizeof(struct POS));
if (!pos)
 {
  printf("Error.");
  exit(1);
 }
r=g;
s=g;
ultimo = 0;
while (r != NULL) do
{
  if (!strcmp(profesor,(*r).nombreprofe)) then
        ultimo=ultimo + 1;
  r=(*r).apun;
}
quot=(ultimo/3);
rem=(ultimo%3);
if (rem != 0) then
 grupo=quot + 1;
else
 grupo=quot;
fin=0;
for (j=1; j <= grupo; j++)
{
 inicio=fin+1;
 a = 0;
 while (( s != NULL) && (a < 3)) do
 {
  if (!strcmp(profesor,(*s).nombreprofe)) then
   {
        fin=(*s).num;
        a = a + 1;
   }
  s=(*s).apun;
 }
 pos[j].inicio=inicio;
 pos[j].fin=fin;
}
j=1;
inicio=pos[j].inicio;
fin=pos[j].fin;
numc=ventanas_ponen_datos(g,inicio,fin,profesor);      // Despliega los datos de profesor en pantalla
c=END;
while (c != ESC) do
{
 c=bioskey(0);
 switch(c)
 {
       case F1: llama_ayuda(56);                       // Llama al módulo AYUDA
                break;
                                                       // Cambia a la siguiente página
       case PGDN:  numc--;
                   for (k=274; k <= numc; k++)
                        quita(tope);                   // Quita ventana gráfica
                   j++;
                   if (j == (grupo+1)) then
                    j=1;
```

```
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_ponen_datos(g,inicio,fin,profesor);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,269,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;
                                                    // Cambia a la página anterior
        case PGUP:  numc--;
                    for(k=274; k <= numc; k++)
                        quita(tope);                // Quita ventanas gráficas restantes
                    j--;
                    if (j == 0) then
                      j=grupo;
                    inicio=pos[j].inicio;
                    fin=pos[j].fin;
                    numc=ventanas_ponen_datos(g,inicio,fin,profesor);
                    strcpy(sig,"PAG. ");
                    itoa(j,cad,10);
                    strcat(sig,cad);
                    texto(sig,269,RED,SANS_SERIF_FONT,1,2,1,2);
                    break;
                                        // Asigna -3 para mostrar en pantalla los datos
        case DERECHA:  dato = -3;       // de profesor de la siguiente materia.
                       c = ESC;
                       break;
                                        // Asigna -4 para mostrar en pantalla los datos
        case IZQUIERDA: dato = -4;      // de profesor de la materia anterior.
                       c = ESC;
                       break;

        case ESC :     dato = -2;       // Salir.
                       break;
   }
  }
  numc--;
  for (k=274; k <= numc; k++)
    quita(tope);        // Quita ventanas gráficas restantes
  farfree(pos);         // Libera el espacio asignado para pos
  return(dato);
}

//   Busca la cadena profesor en la lista enlazada GRUPO.
int busca_profesor_resultado(struct MATERIA *p, char profesor[100], int materia[50])
{
 struct MATERIA *r;
 struct GRUPO *g;
 int j,a;

r = p;
j = 0;
 while (r != NULL) do
  {
   g = (*r).grupos;
   a = 0;
   while ((g != NULL) && (a == 0)) do
    {
        if (!strcmp((*g).nombreprofe,profesor)) then
            a = 1;
        g = (*g).apun;
    }
  if (a == 1) then
   {
    materia[j] = (*r).num;
```

```
      j = j + 1;
     }
    r = (*r).materias;
  }
 materia[j] = -2;
 return(j);
}

//   Despliega en pantalla los datos de profesor para cada materia.
void consulta_datos_profesor_todos(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10])
{
 struct MATERIA *r,*m,*s;
 struct GRUPO *g,*t;
 int k,c,b,materia[50],ultimo,n,e,num;
 char numero[100],cadena[100];

 r = p;
 m = p;
 if (y == 3) then
 quita(tope);          // Quita ventana gráfica  240
 for (k=239; k <= 240; k++)
  {
   setfillstyle(SOLID_FILL,GREEN);
   bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  }
 while ((*r).num != arreglo[a].materia) do
          r = (*r).materias;
 g = (*r).grupos;
 while ((*g).num != arreglo[a].grupo) do
          g = (*g).apun;
 if (y == 3) then
   texto("Presione < ESC > para ir al menú . . . . ",248,LIGHTBLUE,SMALL_FONT,3,2,2,1);
 else
  {
   texto("Las flechas de posición (IZQ y DER) cambian materia",329,LIGHTBLUE,SMALL_FONT,2,2,2,1);
   texto("<ESC> para salir",330,LIGHTBLUE,SMALL_FONT,2,2,2,1);
   texto("PgUp/PgDn",331,LIGHTBLUE,SMALL_FONT,2,2,2,1);
  }
 // R.F.C., profesor y tipo
 texto((*g).rfc,346,LIGHTRED,SMALL_FONT,3,2,2,1);
 texto((*g).nombreprofe,347,LIGHTRED,SMALL_FONT,3,2,2,1);
 c=(*g).tipoprofe;
 switch(c)
  {
          case 1: texto("TITULAR",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                  break;
          case 2: texto("AYUDANTE",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                  break;
          case 3: texto("SIN NOMBRAMIENTO",348,LIGHTRED,SMALL_FONT,2,2,2,1);
                  break;
  }
 // Teléfono
 strcpy(cadena,"Teléfono de la Oficina:  ");
 strcat(cadena,(*g).telofic);
 strcat(cadena,"      Teléfono de la Casa:  ");
 strcat(cadena,(*g).telcasa);
 texto(cadena,372,RED,SMALL_FONT,1,1,2,1);
 cuadro(373,CYAN);
 // Materia
 texto("MAT. 1°,267,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
 strcpy(cadena,(*r).nombre);
 strcat(cadena,"   Sem.  ");
 strcat(cadena,semestre);
 texto(cadena,268,LIGHTBLUE,SMALL_FONT,1,1,2,1);
 texto("Pág. 1°,269,RED,SANS_SERIF_FONT,1,2,1,2);
```

```
// Etiquetas
texto("GRUPO",270,BLUE,SMALL_FONT,1,1,2,1);
texto("SALON",271,BLUE,SMALL_FONT,1,1,2,1);
texto("DIAS",272,BLUE,SMALL_FONT,1,1,2,1);
texto("HORARIO",273,BLUE,SMALL_FONT,1,1,2,1);
f = (*r).grupos;
k = 0;
while (f != NULL) do
 {
  if (!strcmp((*f).nombreprofe,(*g).nombreprofe)) then
        k = k + 1;
  f = (*f).apun;
 }
if (y == 3) then                    // Despliega en pantalla los datos de profesor de la siguiente materia
 {
  e = 1;
  while (e != -2) do
   e = mover_datos_profesor_todos((*r).grupos,(*g).nombreprofe,k);
 }
else
{
 b = busca_profesor_resultado(m,(*g).nombreprofe,materia);
 for (n=0; materia[n] != -2; n++)
       ultimo = n;
 e = 1;
 s = p;
 b = 0;
 while ((*s).num != materia[b]) do
       s = (*s).materias;
 while (e != -2) do
  {
      e = mover_datos_profesor_todos((*s).grupos,(*g).nombreprofe,k);
      if (e == -3) then
       {
        b++;
        if (b == (ultimo + 1)) then
              b = 0;
        num = b + 1;
        s = busca_materia(m,materia[b]);
        strcpy(cadena,"MAT.  ");
        itoa(num,numero,10);
        strcat(cadena,numero);
        texto(cadena,267,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
        strcpy(cadena,(*s).nombre);
        strcat(cadena,"   Sem.  ");
        strcat(cadena,semestre);
        texto(cadena,286,LIGHTBLUE,SMALL_FONT,1,1,2,1);
        f = (*s).grupos;
        while (strcmp((*f).nombreprofe,(*g).nombreprofe)) do
              f = (*f).apun;
        c = (*f).tipoprofe;
        switch(c)
         {
          case 1: texto("TITULAR",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                  break;
          case 2: texto("AYUDANTE",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                  break;
          case 3: texto("SIN NOMBRAMIENTO",348,LIGHTRED,SMALL_FONT,2,2,2,1);
                  break;
         }
        e = 1;
       }
      if (e == -4) then                 // Despliega en pantalla los datos de profesor de la materia anterior
       {
        b--;
```

```
            if (b == - 1) then
                  b = ultimo;
            num = b + 1;
            s = busca_materia(m,materia[b]);
            strcpy(cadena,"MAT.  ");
            itoa(num,numero,10);
            strcat(cadena,numero);
            texto(cadena,287,LIGHTRED,SANS_SERIF_FONT,1,2,1,2);
            strcpy(cadena,(*s).nombre);
            strcat(cadena,"   Sem.  ");
            strcat(cadena,semestre);
            texto(cadena,288,LIGHTBLUE,SMALL_FONT,1,1,2,1);
            f = (*s).grupos;
            while (strcmp((*f).nombreprofe,(*g).nombreprofe)) do
                  f = (*f).apun;
            c = (*f).tipoprofe;
            switch(c)
             {
              case 1: texto("TITULAR",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                      break;
              case 2: texto("AYUDANTE",348,LIGHTRED,SMALL_FONT,3,2,2,1);
                      break;
              case 3: texto("SIN NOMBRAMIENTO",348,LIGHTRED,SMALL_FONT,2,2,2,1);
                      break;
             }
            e = 1;
           }
      }
   }
  for (k=267; k <= 273; k++)
   {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
  k = 373;
  setfillstyle(SOLID_FILL,GREEN);
  bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  if (y == 3) then
   {
    k = 346;
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
    k = 372;
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
  for (k=346; k <= 346; k++)
   {
    setfillstyle(SOLID_FILL,GREEN);
    bar(vent[k].x1,vent[k].y1,vent[k].x2,vent[k].y2);
   }
  ventana_menu_consulta();                    // Despliega el menú de consulta de profesores
  if (y == 3) then
   {
    pon(tope,240,COPY_PUT);          // Pone ventana gráfica
    texto((*r).nombre,240,BLUE,SMALL_FONT,1,1,2,1);
   }
}
```

B.3.14. VENTANAS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\grafico.h"
#include "d:\cuadros.h"
struct VENTANA vent[400];
struct PILA tope[400];

// Guarda una ventana en la pila
void pon(struct PILA *p, int i, int clase)
{
 long      tam;
 int       tope, flag;
 void far  *ventana;

 // Asigna espacio para un nodo más en la pila
 tope = p[0].num;
 tope = tope + 1;
 tam = imagesize(vent[i].x1,vent[i].y1,vent[i].x2,vent[i].y2);
 if (tam > 0x8000) then
   {
    outtextxy(100,100,"Espacio insuficiente");
    exit(1);
   }
 p[tope].num = i;
 p[tope].tam = tam;
 ventana = farmalloc(tam);
 p[0].num = tope;
 if (!ventana) then
    {
     outtextxy(100,100,"No hay espacio disponible");
     exit(1);
    }
 getimage(vent[i].x1,vent[i].y1,vent[i].x2,vent[i].y2,ventana);
 flag = graphresult();
 if (flag != grOk)
  {
   outtextxy(10,400,grapherrormsg(flag));
   exit(1);
  }
 putimage(vent[i].x1,vent[i].y1,ventana,clase);
 flag = graphresult();
 if (flag != grOk)
  {
   outtextxy(10,400,grapherrormsg(flag));
   exit(1);
  }
          // PUTXM copia datos de memoria convencional a memoria extendida
 flag = putxm(ventana,p[0].ventana,p[i].tam);
 if (flag != 0) then
    exit(1);
 p[tope].ventana = p[0].ventana;
 p[0].ventana = p[0].ventana + p[tope].tam;
```

```
  farfree(ventana);              // Libera el espacio reservado para ventana
}

// Quita la ventana tope de la pila
void quita(struct PILA *p)
{
 int        i, tope, flag;
 void far  *ventana;

 // Abre archivo y saca información
 tope = p[0].num;
 i = p[tope].num;
 ventana = farmalloc(p[tope].tam);
 if (!ventana)
   {
    outtextxy(100,100,"No hay espacio disponible");
    exit(1);
   }
          // GETXM copia datos de memoria expandida a memoria convencional
 flag = getxm(p[tope].ventana,ventana,p[tope].tam);
 if (flag != 0) then
   exit(1);
 putimage(vent[i].x1,vent[i].y1,ventana,COPY_PUT);
 flag = graphresult();
 if (flag != grOk)
   {
    outtextxy(10,400,grapherrormsg(flag));
    exit(1);
   }
 if (tope == 1) then
   p[0].ventana = 0x700000;
 else
   p[0].ventana = p[tope-1].ventana;
 p[0].num = tope - 1;
 farfree(ventana);              // Libera el espacio reservado para ventana
}

//  Guarda las dimensiones de todas las ventanas utilizadas en el programa
void ventanas(struct VENTANA v[])
{
 // Ventanas de la portada
 v[2].x1 = 30;
 v[2].y1 = 10;
 v[2].x2 = 610;
 v[2].y2 = 470;

 v[3].x1 = 45;
 v[3].y1 = 25;
 v[3].x2 = 595;
 v[3].y2 = 65;

 v[4].x1 = 45;
 v[4].y1 = 80;
 v[4].x2 = 595;
 v[4].y2 = 120;

 v[5].x1 = 45;
 v[5].y1 = 120;
 v[5].x2 = 595;
 v[5].y2 = 160;

 v[6].x1 = 45;
 v[6].y1 = 190;
 v[6].x2 = 595;
 v[6].y2 = 223;
```

```
v[7].x1 = 45;
v[7].y1 = 225;
v[7].x2 = 595;
v[7].y2 = 256;

v[8].x1 = 45;
v[8].y1 = 256;
v[8].x2 = 595;
v[8].y2 = 289;

v[9].x1 = 45;
v[9].y1 = 320;
v[9].x2 = 595;
v[9].y2 = 350;

v[10].x1 = 45;
v[10].y1 = 375;
v[10].x2 = 595;
v[10].y2 = 405;

v[11].x1 = 45;
v[11].y1 = 420;
v[11].x2 = 595;
v[11].y2 = 450;
// Ventana del menú de Materias
v[21].x1 = 0;
v[21].y1 = 0;
v[21].x2 = 639;
v[21].y2 = 479;
// ventana del nombre 'MATERIA'
v[22].x1 = 10;
v[22].y1 = 10;
v[22].x2 = 629;
v[22].y2 = 40;
// Ventanas del menú de 'Materias'
// Cuadro de la ventana del menú de materias
v[23].x1 = 10;
v[23].y1 = 50;
v[23].x2 = 629;
v[23].y2 = 469;
// ARCHIVOS
v[25].x1 = 20;
v[25].y1 = 60;
v[25].x2 = 120;
v[25].y2 = 110;
// CONSULTA
v[26].x1 = 145;
v[26].y1 = 60;
v[26].x2 = 245;
v[26].y2 = 110;
// GRUPOS
v[27].x1 = 270;
v[27].y1 = 60;
v[27].x2 = 370;
v[27].y2 = 110;
// MOUSE
v[28].x1 = 395;
v[28].y1 = 60;
v[28].x2 = 495;
v[28].y2 = 110;
// SALIR
v[29].x1 = 520;
v[29].y1 = 60;
v[29].x2 = 620;
v[29].y2 = 110;
```

```
// Ventanas del menú de archivos
// cuadro de las ventanas del menú
v[30].x1 = 20;
v[30].y1 = 120;
v[30].x2 = 120;
v[30].y2 = 430;
// LEER
v[31].x1 = 30;
v[31].y1 = 130;
v[31].x2 = 110;
v[31].y2 = 170;
// ESCRIBIR
v[32].x1 = 30;
v[32].y1 = 180;
v[32].x2 = 110;
v[32].y2 = 220;
// CREAR
v[33].x1 = 30;
v[33].y1 = 230;
v[33].x2 = 110;
v[33].y2 = 270;
// AGREGAR
v[34].x1 = 30;
v[34].y1 = 280;
v[34].x2 = 110;
v[34].y2 = 320;
// BORRAR
v[35].x1 = 30;
v[35].y1 = 330;
v[35].x2 = 110;
v[35].y2 = 370;
// CAMBIAR
v[36].x1 = 30;
v[36].y1 = 380;
v[36].x2 = 110;
v[36].y2 = 420;
// Ventanas del menú de CONSULTA
// cuadro de las ventanas de la opción
v[37].x1 = 145;
v[37].y1 = 120;
v[37].x2 = 245;
v[37].y2 = 380;

// MATERIAS
v[38].x1 = 155;
v[38].y1 = 130;
v[38].x2 = 235;
v[38].y2 = 170;
// GRUPOS
v[39].x1 = 155;
v[39].y1 = 180;
v[39].x2 = 235;
v[39].y2 = 220;
// PROFESORES
v[40].x1 = 155;
v[40].y1 = 230;
v[40].x2 = 235;
v[40].y2 = 270;
// SALONES
v[41].x1 = 155;
v[41].y1 = 280;
v[41].x2 = 235;
v[41].y2 = 320;
// EXAMENES
v[42].x1 = 155;
```

```
v[42].y1 = 330;
v[42].x2 = 295;
v[42].y2 = 370;
// Ventanas del menú GRUPOS
// cuadro del menú
v[43].x1 = 270;
v[43].y1 = 120;
v[43].x2 = 370;
v[43].y2 = 330;
// CREAR
v[44].x1 = 280;
v[44].y1 = 130;
v[44].x2 = 360;
v[44].y2 = 170;
// AGREGAR
v[45].x1 = 280;
v[45].y1 = 180;
v[45].x2 = 360;
v[45].y2 = 220;
// BORRAR
v[46].x1 = 280;
v[46].y1 = 230;
v[46].x2 = 360;
v[46].y2 = 270;
// CAMBIAR
v[47].x1 = 280;
v[47].y1 = 280;
v[47].x2 = 360;
v[47].y2 = 320;
// Menú de EXAMEN
// Ventana del menú EXAMEN
v[48].x1 = 395;
v[48].y1 = 120;
v[48].x2 = 495;
v[48].y2 = 330;
// CREAR
v[49].x1 = 405;
v[49].y1 = 130;
v[49].x2 = 485;
v[49].y2 = 170;
// AGREGAR
v[50].x1 = 405;
v[50].y1 = 180;
v[50].x2 = 485;
v[50].y2 = 220;
// ELIMINAR
v[51].x1 = 405;
v[51].y1 = 230;
v[51].x2 = 485;
v[51].y2 = 270;
// CAMBIAR
v[52].x1 = 405;
v[52].y1 = 280;
v[52].x2 = 485;
v[52].y2 = 320;
// SALIR
// Ventana del menú de SALIR
v[53].x1 = 520;
v[53].y1 = 120;
v[53].x2 = 620;
v[53].y2 = 280;
// Semestre
v[54].x1 = 550;
v[54].y1 = 130;
v[54].x2 = 610;
```

```
v[54].y2 = 170;
// ASCII
v[55].x1 = 530;
v[55].y1 = 180;
v[55].x2 = 610;
v[55].y2 = 220;
// Salir
v[56].x1 = 530;
v[56].y1 = 250;
v[56].x2 = 610;
v[56].y2 = 270;
// Cuadro de las distintas opciones, cuadro de trabajo
v[61].x1 = 20;
v[61].y1 = 120;
v[61].x2 = 620;
v[61].y2 = 459;
// Ventana que dice que directorio se esta leyendo
v[62].x1 = 90;
v[62].y1 = 425;
v[62].x2 = 400;
v[62].y2 = 455;
// Ventanas de las opciones de LEER
// LECTURA un archivo
v[63].x1 = 30;
v[63].y1 = 130;
v[63].x2 = 150;
v[63].y2 = 170;
// Cambio de SUBDIRECTORIO
v[64].x1 = 150;
v[64].y1 = 130;
v[64].x2 = 250;
v[64].y2 = 170;
// Elección de la cadena para buscar todos los archivos similares
v[65].x1 = 270;
v[65].y1 = 130;
v[65].x2 = 370;
v[65].y2 = 170;
// SALIR
v[66].x1 = 390;
v[66].y1 = 130;
v[66].x2 = 490;
v[66].y2 = 170;
// Subdirectorio actual
v[67].x1 = 510;
v[67].y1 = 130;
v[67].x2 = 610;
v[67].y2 = 170;
// Ventana de pregunta con el usuario (el programa le pregunta una cadena)
v[68].x1 = 30;
v[68].y1 = 200;
v[68].x2 = 300;
v[68].y2 = 250;
// Ventana donde contesta el usuario a la pregunta de la ventana anterior
v[69].x1 = 320;
v[69].y1 = 200;
v[69].x2 = 610;
v[69].y2 = 250;
// Ventanas para los nombre de los archivos en disco
v[71].x1 = 30;
v[71].y1 = 190;
v[71].x2 = 150;
v[71].y2 = 290;

v[72].x1 = 160;
v[72].y1 = 190;
```

```
v[72].x2 = 250;
v[72].y2 = 230;

v[73].x1 = 270;
v[73].y1 = 190;
v[73].x2 = 370;
v[73].y2 = 230;

v[74].x1 = 390;
v[74].y1 = 190;
v[74].x2 = 490;
v[74].y2 = 230;

v[75].x1 = 510;
v[75].y1 = 190;
v[75].x2 = 610;
v[75].y2 = 230;

v[76].x1 = 30;
v[76].y1 = 250;
v[76].x2 = 130;
v[76].y2 = 290;

v[77].x1 = 150;
v[77].y1 = 250;
v[77].x2 = 250;
v[77].y2 = 290;

v[78].x1 = 270;
v[78].y1 = 250;
v[78].x2 = 370;
v[78].y2 = 290;

v[79].x1 = 390;
v[79].y1 = 250;
v[79].x2 = 490;
v[79].y2 = 290;

v[80].x1 = 510;
v[80].y1 = 250;
v[80].x2 = 610;
v[80].y2 = 290;

v[81].x1 = 30;
v[81].y1 = 310;
v[81].x2 = 130;
v[81].y2 = 350;

v[82].x1 = 150;
v[82].y1 = 310;
v[82].x2 = 250;
v[82].y2 = 350;

v[83].x1 = 270;
v[83].y1 = 310;
v[83].x2 = 370;
v[83].y2 = 350;

v[84].x1 = 390;
v[84].y1 = 310;
v[84].x2 = 490;
v[84].y2 = 350;

v[85].x1 = 510;
v[85].y1 = 310;
```

```
v[85].x2 = 610;
v[85].y2 = 350;

v[86].x1 = 90;
v[86].y1 = 370;
v[86].x2 = 130;
v[86].y2 = 410;

v[87].x1 = 150;
v[87].y1 = 370;
v[87].x2 = 250;
v[87].y2 = 410;

v[88].x1 = 270;
v[88].y1 = 370;
v[88].x2 = 370;
v[88].y2 = 410;

v[89].x1 = 390;
v[89].y1 = 370;
v[89].x2 = 490;
v[89].y2 = 410;

v[90].x1 = 510;
v[90].y1 = 370;
v[90].x2 = 610;
v[90].y2 = 410;

v[100].x1 = 20;
v[100].y1 = 60;
v[100].x2 = 620;
v[100].y2 = 110;

v[115].x1 = 40;
v[115].y1 = 390;
v[115].x2 = 600;
v[115].y2 = 440;
// Datos de materia
v[101].x1 = 40;
v[101].y1 = 134;
v[101].x2 = 310;
v[101].y2 = 184;

v[102].x1 = 330;
v[102].y1 = 134;
v[102].x2 = 600;
v[102].y2 = 184;

v[103].x1 = 40;
v[103].y1 = 198;
v[103].x2 = 310;
v[103].y2 = 248;

v[104].x1 = 330;
v[104].y1 = 198;
v[104].x2 = 600;
v[104].y2 = 248;

v[105].x1 = 40;
v[105].y1 = 262;
v[105].x2 = 310;
v[105].y2 = 312;

v[106].x1 = 330;
v[106].y1 = 262;
```

```
v[106].x2 = 600;
v[106].y2 = 312;

v[107].x1 = 330;
v[107].y1 = 282;
v[107].x2 = 400;
v[107].y2 = 312;

v[108].x1 = 430;
v[108].y1 = 282;
v[108].x2 = 500;
v[108].y2 = 312;

v[109].x1 = 530;
v[109].y1 = 282;
v[109].x2 = 600;
v[109].y2 = 312;

v[110].x1 = 40;
v[110].y1 = 328;
v[110].x2 = 310;
v[110].y2 = 378;

v[111].x1 = 330;
v[111].y1 = 328;
v[111].x2 = 455;
v[111].y2 = 378;

v[112].x1 = 475;
v[112].y1 = 328;
v[112].x2 = 600;
v[112].y2 = 378;

v[113].x1 = 330;
v[113].y1 = 328;
v[113].x2 = 600;
v[113].y2 = 378;
// Ventanas de ayuda, agregar y salir de la opción agregar
v[117].x1 = 41;
v[117].y1 = 282;
v[117].x2 = 213;
v[117].y2 = 317;

v[118].x1 = 284;
v[118].y1 = 282;
v[118].x2 = 406;
v[118].y2 = 317;

v[119].x1 = 427;
v[119].y1 = 282;
v[119].x2 = 599;
v[119].y2 = 317;
// Ventanas de la opción de cambiar
v[120].x1 = 30;
v[120].y1 = 130;
v[120].x2 = 610;
v[120].y2 = 170;

v[121].x1 = 30;
v[121].y1 = 190;
v[121].x2 = 610;
v[121].y2 = 260;

v[122].x1 = 30;
v[122].y1 = 260;
```

```
v[122].x2 = 610;
v[122].y2 = 290;

v[123].x1 = 30;
v[123].y1 = 310;
v[123].x2 = 610;
v[123].y2 = 350;

v[124].x1 = 30;
v[124].y1 = 370;
v[124].x2 = 610;
v[124].y2 = 410;
// Ventanas para cada dato de la lista de materias
// Num
v[130].x1 = 60;
v[130].y1 = 137;
v[130].x2 = 90;
v[130].y2 = 163;
// Clave
v[131].x1 = 120;
v[131].y1 = 137;
v[131].x2 = 170;
v[131].y2 = 163;
// Materia
v[132].x1 = 200;
v[132].y1 = 137;
v[132].x2 = 500;
v[132].y2 = 163;
// Laboratorio
v[133].x1 = 530;
v[133].y1 = 137;
v[133].x2 = 560;
v[133].y2 = 163;
// Num
v[134].x1 = 60;
v[134].y1 = 197;
v[134].x2 = 90;
v[134].y2 = 223;
// Clave
v[135].x1 = 120;
v[135].y1 = 197;
v[135].x2 = 170;
v[135].y2 = 223;
// Materia
v[136].x1 = 200;
v[136].y1 = 197;
v[136].x2 = 500;
v[136].y2 = 223;
// Laboratorio
v[137].x1 = 530;
v[137].y1 = 197;
v[137].x2 = 560;
v[137].y2 = 223;
 // Num
v[138].x1 = 60;
v[138].y1 = 257;
v[138].x2 = 90;
v[138].y2 = 283;
// Clave
v[139].x1 = 120;
v[139].y1 = 257;
v[139].x2 = 170;
v[139].y2 = 283;
// Materia
v[140].x1 = 200;
```

```
v[140].y1 = 257;
v[140].x2 = 500;
v[140].y2 = 283;
// Laboratorio
v[141].x1 = 530;
v[141].y1 = 257;
v[141].x2 = 580;
v[141].y2 = 283;
// Num
v[142].x1 = 60;
v[142].y1 = 317;
v[142].x2 = 90;
v[142].y2 = 343;
// Clave
v[143].x1 = 120;
v[143].y1 = 317;
v[143].x2 = 170;
v[143].y2 = 343;
// Materia
v[144].x1 = 200;
v[144].y1 = 317;
v[144].x2 = 500;
v[144].y2 = 343;
// Laboratorio
v[145].x1 = 530;
v[145].y1 = 317;
v[145].x2 = 580;
v[145].y2 = 343;
 // Num
v[146].x1 = 60;
v[146].y1 = 377;
v[146].x2 = 90;
v[146].y2 = 403;
// Clave
v[147].x1 = 120;
v[147].y1 = 377;
v[147].x2 = 170;
v[147].y2 = 403;
// Materia
v[148].x1 = 200;
v[148].y1 = 377;
v[148].x2 = 500;
v[148].y2 = 403;
// Laboratorio
v[149].x1 = 530;
v[149].y1 = 377;
v[149].x2 = 580;
v[149].y2 = 403;
// Ventana del menú de ayuda
v[150].x1 = 20;
v[150].y1 = 420;
v[150].x2 = 620;
v[150].y2 = 459;
// Ventana de la opción ayuda
v[152].x1 = 30;
v[152].y1 = 428;
v[152].x2 = 180;
v[152].y2 = 453;
// Ventana de la opción salida
v[153].x1 = 245;
v[153].y1 = 428;
v[153].x2 = 396;
v[153].y2 = 453;
// Ventana de la opción PgUp/PgDn
v[154].x1 = 460;
```

```
v[154].y1 = 425;
v[154].x2 = 610;
v[154].y2 = 450;
// Ventanas de error de lectura o creación de lista
v[156].x1 = 35;
v[156].y1 = 257;
v[156].x2 = 605;
v[156].y2 = 322;

v[157].x1 = 55;
v[157].y1 = 287;
v[157].x2 = 585;
v[157].y2 = 312;
//Ventanas de lectura para el menú de la opción de GRUPO
//número de grupo
v[160].x1 = 30;
v[160].y1 = 135;
v[160].x2 = 210;
v[160].y2 = 180;

v[161].x1 = 240;
v[161].y1 = 135;
v[161].x2 = 320;
v[161].y2 = 180;

v[162].x1 = 350;
v[162].y1 = 135;
v[162].x2 = 465;
v[162].y2 = 180;

v[163].x1 = 495;
v[163].y1 = 135;
v[163].x2 = 610;
v[163].y2 = 180;

v[164].x1 = 340;
v[164].y1 = 135;
v[164].x2 = 610;
v[164].y2 = 180;
//Ventana de mensaje
v[165].x1 = 30;
v[165].y1 = 195;
v[165].x2 = 610;
v[165].y2 = 225;
//Ventanas de datos de profesor, salón, horarios, días y número de alumnos
v[166].x1 = 30;
v[166].y1 = 237;
v[166].x2 = 130;
v[166].y2 = 282;

v[167].x1 = 150;
v[167].y1 = 237;
v[167].x2 = 250;
v[167].y2 = 282;

v[168].x1 = 270;
v[168].y1 = 237;
v[168].x2 = 370;
v[168].y2 = 282;

v[169].x1 = 390;
v[169].y1 = 237;
v[169].x2 = 490;
v[169].y2 = 282;
```

```
v[170].x1 = 510;
v[170].y1 = 297;
v[170].x2 = 610;
v[170].y2 = 888;

v[171].x1 = 30;
v[171].y1 = 294;
v[171].x2 = 306;
v[171].y2 = 324;

v[172].x1 = 336;
v[172].y1 = 294;
v[172].x2 = 610;
v[172].y2 = 324;

v[173].x1 = 30;
v[173].y1 = 336;
v[173].x2 = 306;
v[173].y2 = 366;

v[174].x1 = 336;
v[174].y1 = 336;
v[174].x2 = 610;
v[174].y2 = 366;

v[175].x1 = 336;
v[175].y1 = 336;
v[175].x2 = 396;
v[175].y2 = 366;

v[176].x1 = 408;
v[176].y1 = 336;
v[176].x2 = 467;
v[176].y2 = 366;

v[177].x1 = 477;
v[177].y1 = 336;
v[177].x2 = 536;
v[177].y2 = 366;

v[178].x1 = 548;
v[178].y1 = 336;
v[178].x2 = 610;
v[178].y2 = 366;

v[179].x1 = 170;
v[179].y1 = 378;
v[179].x2 = 470;
v[179].y2 = 408;

v[180].x1 = 30;
v[180].y1 = 294;
v[180].x2 = 110;
v[180].y2 = 324;

v[181].x1 = 130;
v[181].y1 = 294;
v[181].x2 = 210;
v[181].y2 = 324;

v[182].x1 = 230;
v[182].y1 = 294;
v[182].x2 = 310;
v[182].y2 = 324;
```

```
v[183].x1 = 350;
v[183].y1 = 294;
v[183].x2 = 410;
v[183].y2 = 324;

v[184].x1 = 430;
v[184].y1 = 294;
v[184].x2 = 510;
v[184].y2 = 324;

v[185].x1 = 530;
v[185].y1 = 294;
v[185].x2 = 610;
v[185].y2 = 324;

v[186].x1 = 30;
v[186].y1 = 336;
v[186].x2 = 180;
v[186].y2 = 366;

v[187].x1 = 186;
v[187].y1 = 336;
v[187].x2 = 250;
v[187].y2 = 366;

v[188].x1 = 260;
v[188].y1 = 336;
v[188].x2 = 322;
v[188].y2 = 366;

v[189].x1 = 332;
v[189].y1 = 336;
v[189].x2 = 394;
v[189].y2 = 366;


v[190].x1 = 404;
v[190].y1 = 336;
v[190].x2 = 466;
v[190].y2 = 366;

v[191].x1 = 476;
v[191].y1 = 336;
v[191].x2 = 536;
v[191].y2 = 366;

v[192].x1 = 546;
v[192].y1 = 336;
v[192].x2 = 610;
v[192].y2 = 366;

v[193].x1 = 336;
v[193].y1 = 336;
v[193].x2 = 467;
v[193].y2 = 366;

v[194].x1 = 477;
v[194].y1 = 336;
v[194].x2 = 610;
v[194].y2 = 366;
// Ventanas de la opción cambiar datos de grupo
// Grupo
v[195].x1 = 90;
v[195].y1 = 136;
v[195].x2 = 240;
```

```
v[195].y2 = 165;
// Classegrupo
v[196].x1 = 300;
v[196].y1 = 135;
v[196].x2 = 550;
v[196].y2 = 165;
// Grupo
v[197].x1 = 90;
v[197].y1 = 195;
v[197].x2 = 240;
v[197].y2 = 225;
// Classegrupo
v[198].x1 = 300;
v[198].y1 = 195;
v[198].x2 = 550;
v[198].y2 = 225;
// Grupo
v[199].x1 = 90;
v[199].y1 = 255;
v[199].x2 = 240;
v[199].y2 = 285;
// Classegrupo
v[200].x1 = 300;
v[200].y1 = 255;
v[200].x2 = 550;
v[200].y2 = 285;
// Grupo
v[201].x1 = 90;
v[201].y1 = 315;
v[201].x2 = 240;
v[201].y2 = 345;
// Classegrupo
v[202].x1 = 300;
v[202].y1 = 315;
v[202].x2 = 550;
v[202].y2 = 345;
// Grupo
v[203].x1 = 90;
v[203].y1 = 375;
v[203].x2 = 240;
v[203].y2 = 405;
// Classegrupo
v[204].x1 = 300;
v[204].y1 = 375;
v[204].x2 = 550;
v[204].y2 = 405;
// Numgrupo
v[205].x1 = 200;
v[205].y1 = 317;
v[205].x2 = 320;
v[205].y2 = 343;
// Classegrupo
v[206].x1 = 370;
v[206].y1 = 317;
v[206].x2 = 610;
v[206].y2 = 343;
// Num
v[207].x1 = 30;
v[207].y1 = 377;
v[207].x2 = 150;
v[207].y2 = 409;
// Numgrupo
v[208].x1 = 200;
v[208].y1 = 377;
v[208].x2 = 320;
```

```
v[208].y2 = 405;
// Casegrupo
v[209].x1 = 570;
v[209].y1 = 377;
v[209].x2 = 610;
v[209].y2 = 403;

// Mensaje de selección
v[210].x1 = 30;
v[210].y1 = 150;
v[210].x2 = 610;
v[210].y2 = 181;

v[211].x1 = 30;
v[211].y1 = 171;
v[211].x2 = 160;
v[211].y2 = 202;

v[212].x1 = 160;
v[212].y1 = 171;
v[212].x2 = 310;
v[212].y2 = 202;

v[213].x1 = 330;
v[213].y1 = 171;
v[213].x2 = 460;
v[213].y2 = 202;

v[214].x1 = 460;
v[214].y1 = 171;
v[214].x2 = 610;
v[214].y2 = 202;

v[215].x1 = 30;
v[215].y1 = 212;
v[215].x2 = 160;
v[215].y2 = 243;

v[216].x1 = 160;
v[216].y1 = 212;
v[216].x2 = 310;
v[216].y2 = 243;

v[217].x1 = 330;
v[217].y1 = 212;
v[217].x2 = 460;
v[217].y2 = 243;

v[218].x1 = 460;
v[218].y1 = 212;
v[218].x2 = 610;
v[218].y2 = 243;

v[219].x1 = 30;
v[219].y1 = 253;
v[219].x2 = 160;
v[219].y2 = 284;

v[220].x1 = 160;
v[220].y1 = 253;
v[220].x2 = 610;
v[220].y2 = 284;

v[221].x1 = 330;
v[221].y1 = 253;
```

```
v[221].x2 = 490;
v[221].y2 = 284;

v[222].x1 = 480;
v[222].y1 = 253;
v[222].x2 = 610;
v[222].y2 = 284;
// Ventanas de la opción de profesor titular o ayudante
v[223].x1 = 30;
v[223].y1 = 294;
v[223].x2 = 190;
v[223].y2 = 324;

v[224].x1 = 210;
v[224].y1 = 294;
v[224].x2 = 450;
v[224].y2 = 324;

v[225].x1 = 470;
v[225].y1 = 294;
v[225].x2 = 530;
v[225].y2 = 324;

v[226].x1 = 550;
v[226].y1 = 294;
v[226].x2 = 610;
v[226].y2 = 324;

v[227].x1 = 470;
v[227].y1 = 294;
v[227].x2 = 610;
v[227].y2 = 324;
// Ventanas de la opción CONSULTA
v[230].x1 = 30;
v[230].y1 = 428;
v[230].x2 = 130;
v[230].y2 = 453;

v[231].x1 = 30;
v[231].y1 = 428;
v[231].x2 = 160;
v[231].y2 = 453;

v[232].x1 = 160;
v[232].y1 = 428;
v[232].x2 = 310;
v[232].y2 = 453;

v[233].x1 = 330;
v[233].y1 = 428;
v[233].x2 = 460;
v[233].y2 = 453;

v[234].x1 = 480;
v[234].y1 = 428;
v[234].x2 = 610;
v[234].y2 = 453;
// Ventanas de la opción consulta de grupos
v[235].x1 = 30;
v[235].y1 = 130;
v[235].x2 = 100;
v[235].y2 = 170;

v[236].x1 = 30;
v[236].y1 = 130;
```

```
v[235].x2 = 150;
v[235].y2 = 170;

v[237].x1 = 145;
v[237].y1 = 130;
v[237].x2 = 215;
v[237].y2 = 170;

v[238].x1 = 230;
v[238].y1 = 130;
v[238].x2 = 320;
v[238].y2 = 170;

v[239].x1 = 335;
v[239].y1 = 130;
v[239].x2 = 405;
v[239].y2 = 170;


v[240].x1 = 420;
v[240].y1 = 130;
v[240].x2 = 610;
v[240].y2 = 170;

v[241].x1 = 30;
v[241].y1 = 175;
v[241].x2 = 610;
v[241].y2 = 205;

v[242].x1 = 30;
v[242].y1 = 216;
v[242].x2 = 610;
v[242].y2 = 256;

v[243].x1 = 30;
v[243].y1 = 267;
v[243].x2 = 610;
v[243].y2 = 307;

v[244].x1 = 30;
v[244].y1 = 318;
v[244].x2 = 610;
v[244].y2 = 358;

v[245].x1 = 30;
v[245].y1 = 369;
v[245].x2 = 610;
v[245].y2 = 409;

v[246].x1 = 30;
v[246].y1 = 428;
v[246].x2 = 610;
v[246].y2 = 453;
// Ventanas de la opción consulta de grupos
// Grupo
v[247].x1 = 30;
v[247].y1 = 130;
v[247].x2 = 300;
v[247].y2 = 170;
// Clasegrupo
v[248].x1 = 340;
v[248].y1 = 190;
v[248].x2 = 610;
v[248].y2 = 170;
// Materia
```

```
v[249].x1 = 45;
v[249].y1 = 195;
v[249].x2 = 300;
v[249].y2 = 225;
// Clave
v[250].x1 = 340;
v[250].y1 = 195;
v[250].x2 = 595;
v[250].y2 = 225;
// Profesor
v[251].x1 = 45;
v[251].y1 = 255;
v[251].x2 = 300;
v[251].y2 = 285;
// RFC
v[252].x1 = 340;
v[252].y1 = 255;
v[252].x2 = 595;
v[252].y2 = 285;
// Salón
v[253].x1 = 45;
v[253].y1 = 315;
v[253].x2 = 300;
v[253].y2 = 345;
// Cupo
v[254].x1 = 340;
v[254].y1 = 315;
v[254].x2 = 595;
v[254].y2 = 345;
// Días
v[255].x1 = 45;
v[255].y1 = 375;
v[255].x2 = 300;
v[255].y2 = 405;
// Horas
v[256].x1 = 340;
v[256].y1 = 375;
v[256].x2 = 595;
v[256].y2 = 405;
// Ventanas de la opción de profesores
v[257].x1 = 55;
v[257].y1 = 195;
v[257].x2 = 205;
v[257].y2 = 225;

v[258].x1 = 230;
v[258].y1 = 195;
v[258].x2 = 585;
v[258].y2 = 225;

v[259].x1 = 55;
v[259].y1 = 255;
v[259].x2 = 205;
v[259].y2 = 285;

v[260].x1 = 230;
v[260].y1 = 255;
v[260].x2 = 585;
v[260].y2 = 285;

v[261].x1 = 55;
v[261].y1 = 315;
v[261].x2 = 205;
v[261].y2 = 345;
```

```
v[262].x1 = 290;
v[262].y1 = 315;
v[262].x2 = 585;
v[262].y2 = 345;

v[263].x1 = 55;
v[263].y1 = 375;
v[263].x2 = 205;
v[263].y2 = 405;

v[264].x1 = 290;
v[264].y1 = 375;
v[264].x2 = 585;
v[264].y2 = 405;
// Datos de profesor
// RFC
v[265].x1 = 30;
v[265].y1 = 130;
v[265].x2 = 165;
v[265].y2 = 170;
// Nombre del profesor
v[266].x1 = 215;
v[266].y1 = 130;
v[266].x2 = 610;
v[266].y2 = 170;
// Clave
v[267].x1 = 45;
v[267].y1 = 220;
v[267].x2 = 130;
v[267].y2 = 250;
// Materia
v[268].x1 = 170;
v[268].y1 = 220;
v[268].x2 = 470;
v[268].y2 = 250;
// Página
v[269].x1 = 510;
v[269].y1 = 220;
v[269].x2 = 595;
v[269].y2 = 250;
// Etiquetas de grupo
v[270].x1 = 45;
v[270].y1 = 280;
v[270].x2 = 130;
v[270].y2 = 290;

v[271].x1 = 150;
v[271].y1 = 280;
v[271].x2 = 250;
v[271].y2 = 290;

v[272].x1 = 280;
v[272].y1 = 280;
v[272].x2 = 450;
v[272].y2 = 290;

v[273].x1 = 450;
v[273].y1 = 280;
v[273].x2 = 595;
v[273].y2 = 290;
// Grupo
v[274].x1 = 45;
v[274].y1 = 300;
v[274].x2 = 130;
v[274].y2 = 330;
```

```
// Salón
v[275].x1 = 150;
v[275].y1 = 300;
v[275].x2 = 250;
v[275].y2 = 330;
// Días
v[276].x1 = 270;
v[276].y1 = 300;
v[276].x2 = 430;
v[276].y2 = 330;
// Horas
v[277].x1 = 450;
v[277].y1 = 300;
v[277].x2 = 595;
v[277].y2 = 330;
// Grupo
v[278].x1 = 45;
v[278].y1 = 340;
v[278].x2 = 130;
v[278].y2 = 370;
// Salón
v[279].x1 = 150;
v[279].y1 = 340;
v[279].x2 = 250;
v[279].y2 = 370;
// Días
v[280].x1 = 270;
v[280].y1 = 340;
v[280].x2 = 430;
v[280].y2 = 370;
// Horas
v[281].x1 = 450;
v[281].y1 = 340;
v[281].x2 = 595;
v[281].y2 = 370;
// Grupo
v[282].x1 = 45;
v[282].y1 = 380;
v[282].x2 = 130;
v[282].y2 = 410;
// Salón
v[283].x1 = 150;
v[283].y1 = 380;
v[283].x2 = 250;
v[283].y2 = 410;
// Días
v[284].x1 = 270;
v[284].y1 = 380;
v[284].x2 = 430;
v[284].y2 = 410;
// Horas
v[285].x1 = 450;
v[285].y1 = 380;
v[285].x2 = 595;
v[285].y2 = 410;
}
```

B.3.15. VENTDOS.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <alloc.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <bios.h>
#include <dir.h>
#include <dos.h>
#include "d:\grafico.h"
#include "d:\cuadros.h"


// Esta función es la continuación de la función ventanas
void ventdos(struct VENTANA v[])
{

 // Ventanas de la opcion consulta de datos de asiones
 // Cuadro de datos
 v[286].x1 = 30;
 v[286].y1 = 182;
 v[286].x2 = 610;
 v[286].y2 = 412;

 v[287].x1 = 45;
 v[287].y1 = 195;
 v[287].x2 = 145;
 v[287].y2 = 225;

 v[288].x1 = 170;
 v[288].y1 = 195;
 v[288].x2 = 470;
 v[288].y2 = 225;

 v[289].x1 = 490;
 v[289].y1 = 195;
 v[289].x2 = 595;
 v[289].y2 = 225;

 v[290].x1 = 45;
 v[290].y1 = 256;
 v[290].x2 = 125;
 v[290].y2 = 286;

 v[291].x1 = 140;
 v[291].y1 = 236;
 v[291].x2 = 335;
 v[291].y2 = 286;

 v[292].x1 = 350;
 v[292].y1 = 236;
 v[292].x2 = 480;
 v[292].y2 = 286;

 v[293].x1 = 496;
 v[293].y1 = 236;
 v[293].x2 = 596;
 v[293].y2 = 286;
```

```
v[294].x1 = 45;
v[294].y1 = 281;
v[294].x2 = 125;
v[294].y2 = 311;

v[295].x1 = 140;
v[295].y1 = 281;
v[295].x2 = 335;
v[295].y2 = 311;

v[296].x1 = 350;
v[296].y1 = 281;
v[296].x2 = 480;
v[296].y2 = 311;

v[297].x1 = 495;
v[297].y1 = 281;
v[297].x2 = 595;
v[297].y2 = 311;

v[298].x1 = 45;
v[298].y1 = 324;
v[298].x2 = 125;
v[298].y2 = 354;

v[299].x1 = 140;
v[299].y1 = 324;
v[299].x2 = 335;
v[299].y2 = 354;
// Opción "." de consulta de grupos
v[300].x1 = 350;
v[300].y1 = 324;
v[300].x2 = 480;
v[300].y2 = 354;

v[301].x1 = 495;
v[301].y1 = 324;
v[301].x2 = 595;
v[301].y2 = 354;

v[302].x1 = 45;
v[302].y1 = 367;
v[302].x2 = 125;
v[302].y2 = 397;

v[303].x1 = 140;
v[303].y1 = 367;
v[303].x2 = 335;
v[303].y2 = 397;

v[304].x1 = 350;
v[304].y1 = 367;
v[304].x2 = 480;
v[304].y2 = 397;

v[305].x1 = 495;
v[305].y1 = 367;
v[305].x2 = 595;
v[305].y2 = 397;

v[310].x1 = 90;
v[310].y1 = 208;
v[310].x2 = 210;
v[310].y2 = 278;
```

```
v[311].x1 = 250;
v[311].y1 = 238;
v[311].x2 = 410;
v[311].y2 = 278;

v[312].x1 = 430;
v[312].y1 = 238;
v[312].x2 = 610;
v[312].y2 = 278;

v[313].x1 = 30;
v[313].y1 = 238;
v[313].x2 = 610;
v[313].y2 = 278;

v[314].x1 = 30;
v[314].y1 = 282;
v[314].x2 = 610;
v[314].y2 = 322;

v[315].x1 = 30;
v[315].y1 = 324;
v[315].x2 = 610;
v[315].y2 = 354;

v[316].x1 = 30;
v[316].y1 = 367;
v[316].x2 = 300;
v[316].y2 = 407;

v[317].x1 = 340;
v[317].y1 = 367;
v[317].x2 = 610;
v[317].y2 = 407;

v[318].x1 = 55;
v[318].y1 = 190;
v[318].x2 = 205;
v[318].y2 = 230;

v[319].x1 = 230;
v[319].y1 = 190;
v[319].x2 = 585;
v[319].y2 = 230;

v[320].x1 = 70;
v[320].y1 = 244;
v[320].x2 = 190;
v[320].y2 = 274;

v[321].x1 = 70;
v[321].y1 = 286;
v[321].x2 = 190;
v[321].y2 = 316;

v[322].x1 = 70;
v[322].y1 = 332;
v[322].x2 = 190;
v[322].y2 = 362;

v[323].x1 = 70;
v[323].y1 = 376;
v[323].x2 = 190;
v[323].y2 = 406;
```

```
v[324].x1 = 290;
v[324].y1 = 244;
v[324].x2 = 570;
v[324].y2 = 274;

v[325].x1 = 230;
v[325].y1 = 288;
v[325].x2 = 570;
v[325].y2 = 318;

v[326].x1 = 290;
v[326].y1 = 332;
v[326].x2 = 570;
v[326].y2 = 362;

v[327].x1 = 230;
v[327].y1 = 376;
v[327].x2 = 570;
v[327].y2 = 406;

v[328].x1 = 30;
v[328].y1 = 237;
v[328].x2 = 610;
v[328].y2 = 413;

v[329].x1 = 30;
v[329].y1 = 428;
v[329].x2 = 340;
v[329].y2 = 453;

v[330].x1 = 350;
v[330].y1 = 428;
v[330].x2 = 500;
v[330].y2 = 453;

v[331].x1 = 510;
v[331].y1 = 428;
v[331].x2 = 610;
v[331].y2 = 453;
// Salones
v[332].x1 = 30;
v[332].y1 = 243;
v[332].x2 = 138;
v[332].y2 = 273;

v[333].x1 = 148;
v[333].y1 = 243;
v[333].x2 = 256;
v[333].y2 = 273;

v[334].x1 = 266;
v[334].y1 = 243;
v[334].x2 = 374;
v[334].y2 = 273;

v[335].x1 = 384;
v[335].y1 = 243;
v[335].x2 = 492;
v[335].y2 = 273;

v[336].x1 = 502;
v[336].y1 = 243;
v[336].x2 = 610;
v[336].y2 = 273;
```

```
v[337].x1 = 30;
v[337].y1 = 245;
v[337].x2 = 610;
v[337].y2 = 275;

v[338].x1 = 145;
v[338].y1 = 190;
v[338].x2 = 495;
v[338].y2 = 250;

v[339].x1 = 145;
v[339].y1 = 255;
v[339].x2 = 495;
v[339].y2 = 285;

v[340].x1 = 145;
v[340].y1 = 315;
v[340].x2 = 495;
v[340].y2 = 345;

v[341].x1 = 145;
v[341].y1 = 375;
v[341].x2 = 495;
v[341].y2 = 405;
// Ventanas de profesor
v[342].x1 = 30;
v[342].y1 = 250;
v[342].x2 = 160;
v[342].y2 = 290;

v[343].x1 = 180;
v[343].y1 = 250;
v[343].x2 = 310;
v[343].y2 = 290;

v[344].x1 = 330;
v[344].y1 = 250;
v[344].x2 = 460;
v[344].y2 = 290;

v[345].x1 = 480;
v[345].y1 = 250;
v[345].x2 = 610;
v[345].y2 = 290;
//  Ventanas de datos de profesor opcion *.*
v[346].x1 = 30;
v[346].y1 = 130;
v[346].x2 = 170;
v[346].y2 = 170;

v[347].x1 = 180;
v[347].y1 = 130;
v[347].x2 = 505;
v[347].y2 = 170;

v[348].x1 = 515;
v[348].y1 = 130;
v[348].x2 = 610;
v[348].y2 = 170;
// Ventanas para consulta de MATERIAS
v[349].x1 = 30;
v[349].y1 = 286;
v[349].x2 = 210;
v[349].y2 = 303;
```

```
v[350].x1 = 290;
v[350].y1 = 265;
v[350].x2 = 410;
v[350].y2 = 303;

v[351].x1 = 430;
v[351].y1 = 263;
v[351].x2 = 610;
v[351].y2 = 303;

v[352].x1 = 30;
v[352].y1 = 263;
v[352].x2 = 610;
v[352].y2 = 303;

v[353].x1 = 30;
v[353].y1 = 341;
v[353].x2 = 210;
v[353].y2 = 381;

v[354].x1 = 230;
v[354].y1 = 341;
v[354].x2 = 610;
v[354].y2 = 381;
// Datos de materias
v[355].x1 = 30;
v[355].y1 = 130;
v[355].x2 = 130;
v[355].y2 = 170;

v[356].x1 = 140;
v[356].y1 = 130;
v[356].x2 = 460;
v[356].y2 = 170;

v[357].x1 = 470;
v[357].y1 = 130;
v[357].x2 = 610;
v[357].y2 = 170;

v[358].x1 = 45;
v[358].y1 = 195;
v[358].x2 = 440;
v[358].y2 = 225;

v[359].x1 = 455;
v[359].y1 = 195;
v[359].x2 = 595;
v[359].y2 = 225;

v[360].x1 = 45;
v[360].y1 = 236;
v[360].x2 = 205;
v[360].y2 = 266;

v[361].x1 = 220;
v[361].y1 = 236;
v[361].x2 = 595;
v[361].y2 = 266;

v[362].x1 = 45;
v[362].y1 = 281;
v[362].x2 = 205;
v[362].y2 = 311;
```

```
v[363].x1 = 220;
v[363].y1 = 281;
v[363].x2 = 595;
v[363].y2 = 311;

v[364].x1 = 45;
v[364].y1 = 324;
v[364].x2 = 205;
v[364].y2 = 354;

v[365].x1 = 220;
v[365].y1 = 324;
v[365].x2 = 595;
v[365].y2 = 354;

v[366].x1 = 45;
v[366].y1 = 367;
v[366].x2 = 205;
v[366].y2 = 397;

v[367].x1 = 220;
v[367].y1 = 367;
v[367].x2 = 595;
v[367].y2 = 397;
// Datos del módulo de cambiar datos de exámenes
v[368].x1 = 55;
v[368].y1 = 135;
v[368].x2 = 205;
v[368].y2 = 166;

v[369].x1 = 230;
v[369].y1 = 135;
v[369].x2 = 585;
v[369].y2 = 166;
// Opción consulta de datos de exámenes
v[370].x1 = 45;
v[370].y1 = 135;
v[370].x2 = 300;
v[370].y2 = 166;

v[371].x1 = 340;
v[371].y1 = 135;
v[371].x2 = 595;
v[371].y2 = 165;
// Ventana del teléfono
v[372].x1 = 30;
v[372].y1 = 182;
v[372].x2 = 610;
v[372].y2 = 202;
// Cuadro de la nueva ventana de profesor
v[373].x1 = 90;
v[373].y1 = 215;
v[373].x2 = 610;
v[373].y2 = 415;
// Ventanas de la opción AYUDA
v[376].x1 = 120;
v[376].y1 = 185;
v[376].x2 = 550;
v[376].y2 = 395;

v[376].x1 = 90;
v[376].y1 = 170;
v[376].x2 = 550;
v[376].y2 = 390;
```

```
v[377].x1 = 270;
v[377].y1 = 170;
v[377].x2 = 370;
v[377].y2 = 196;

v[378].x1 = 115;
v[378].y1 = 346;
v[378].x2 = 525;
v[378].y2 = 366;

v[379].x1 = 115;
v[379].y1 = 196;
v[379].x2 = 525;
v[379].y2 = 345;

v[380].x1 = 466;
v[380].y1 = 170;
v[380].x2 = 525;
v[380].y2 = 194;

v[381].x1 = 455;
v[381].y1 = 346;
v[381].x2 = 525;
v[381].y2 = 366;
// Ventanas de la opción errores
// Ventana magenta
v[382].x1 = 90;
v[382].y1 = 207;
v[382].x2 = 550;
v[382].y2 = 353;
// Ventana de mensaje
v[383].x1 = 270;
v[383].y1 = 207;
v[383].x2 = 370;
v[383].y2 = 232;

v[384].x1 = 115;
v[384].y1 = 232;
v[384].x2 = 525;
v[384].y2 = 308;

v[385].x1 = 115;
v[385].y1 = 309;
v[385].x2 = 525;
v[385].y2 = 328;

v[386].x1 = 466;
v[386].y1 = 207;
v[386].x2 = 525;
v[386].y2 = 231;

v[387].x1 = 455;
v[387].y1 = 309;
v[387].x2 = 525;
v[387].y2 = 328;
}
```

B.3.16. EXTMEM.ASM

```
;       EXTMEM.ASM, es un módulo que contiene dos funciones que transfieren datos entre la memoria
; convencional y la memoria extendida.
; Está creado para trabajar en programas compilados en el modelo grande (large).
;

        .MODEL LARGE                                    ; modelo grande

; Segmento de datos
        .DATA
GDT     db      30h dup(0)                              ; Tabla descriptora global

; Segmento de código
        .CODE
; Transfiere un bloque de datos de la memoria extendida y lo pone en la memoria convencional
        PUBLIC  _gebm
_gebm   proc    far
        ARG     source: dword, dest: dword, len: word   ; Argumentos de transferencia
        push    bp                                      ; Guarda la base del stack
        mov     bp, sp
        push    si
        push    di
        push    ds

        mov     si, offset DGROUP: gdt                  ; Pone el offset (desplazamiento) de GDT en si
        mov     ax, ds                                  ; Pone el segmento de GDT en es
        mov     es, ax
                                                        ; Almacena los bytes de acceso correcto
        mov     byte ptr [si + 15h], 93h
        mov     byte ptr [si + 1dh], 93h

        mov     ax, word ptr [source]                   ; Almacena las direcciones fuente en el
        mov     [si + 12h], ax                          ; descriptor
        mov     ax, word ptr [source + 2]
        mov     [si + 14h], al

        mov     ax, word ptr [dest + 2]                 ; Segmento destino * 16
        mov     dx, 16
        mul     dx
        add     ax, word ptr [dest]                     ; + desplazamiento = dirección lineal
        adc     dx, 0
        mov     [si + 1ah], ax                          ; Almacena las direcciones destino en el
        mov     [si + 1ch], dl                          ; descriptor

        mov     cx, word ptr [len]                      ; Almacena el tamaño del bloque de memori
        mov     [si + 10h], cx                          ; en los descriptores fuente y destino
        mov     [si + 18h], cx

        shr     cx, 1                                   ; Convierte el tamaño en palabras
        mov     ah, 87h                                 ; Int 15H fun 87H = Bloque de memoria transferido
        int     15h
        mov     al, ah
        cbw

        pop     ds
        pop     di                                      ; Almacena registros
        pop     si
        pop     bp
        ret
_gebm   endp
```

```
; Transfiere datos de la memoria convencional a la memoria extendida
        PUBLIC  _pubm
_pubm   proc    far
        ARG     source: dword, dest: dword, len: word           ; Argumentos de transferencia
        push    bp                                              ; Guarda la base del stack
        mov     bp, sp
        push    si
        push    di
        push    ds

        mov     si, offset DGROUP: gdt                          ; Pone el offset (desplazamiento) de GDT en si
        mov     ax, ds                                          ; Pone el segmento de datos de GDT en es
        mov     es, ax

        mov     byte ptr [si + 15h], 93h                        ; Almacena bytes de acceso correcto
        mov     byte ptr [si + 1dh], 93h

        mov     ax, word ptr [dest]                             ; Almacena las direcciones destino en el
        mov     [si + 18h], ax                                  ; descriptor
        mov     ax, word ptr [dest + 2]
        mov     [si + 1ch], al

        mov     ax, word ptr [source + 2]                       ; Segmento fuente * 16
        mov     dx, 16
        mul     dx
        add     ax, word ptr [source]                           ; + desplazamiento = dirección lineal
        adc     dx, 0
        mov     [si + 12h], ax                                  ; Almacena las direcciones fuentes en el
        mov     [si + 14h], dl                                  ; descriptor

        mov     cx, word ptr [len]                              ; Almacena el tamaño del bloque de memoria en los
        mov     [si + 10h], cx                                  ; descriptores fuente y destino
        mov     [si + 16h], cx

        shr     cx, 1                                           ; Convierte el tamaño en palabras
        mov     ah, 87h                                         ; Int 15H fun 87H = Bloque de memoria transferido
        int     15h

        mov     al, ah
        cbw

        pop     ds
        pop     di                                              ; Almacena registros
        pop     si
        pop     bp
        ret
_pubm   endp
        end
```

B.3.17. ARCHIVOS.H

```
extern void cambia_elementos_nombre(struct MATERIA *p, int k, struct ARREGLO *arreglo);
extern void ordena_materias_clave(struct MATERIA *p, struct ARREGLO *arreglo);
extern int error_clave_igual(struct MATERIA *p, char cadena[100]);
extern int error_nombre_igual(struct MATERIA *p, char cadena[100]);
extern int error_clave_cambiar(struct MATERIA *p, char cadena[100], int dato);
extern int error_nombre_cambiar(struct MATERIA *p, char cadena[100], int dato);
extern int confirmar(int n);
extern void ventanas_archivos(void);
extern int selecciona_archivo(char archivos[20][15], char *cadena, int *n);
extern void leer(struct MATERIA *p, char semestre[10]);
extern void escribir(struct MATERIA *p, char semestre[10], int tipo);
extern void lee_nodo_materia(struct MATERIA *p, struct MATERIA *s);
extern int menu_agregar(void);
extern void ventana_nodo_lista_materia(void);
extern void crear(struct MATERIA *p);
extern void agregar(struct MATERIA *p);
extern void eliminar(struct MATERIA *p);
extern void cambiar(struct MATERIA *p);
extern void archivos(int i, struct MATERIA *lista, char semestre[10]);
extern int ventanas_ponen_materias(struct MATERIA *p, int inicio, int fin, int*j);
extern int mover_pantallas(int i, int min, int ayuda, int salir, struct MATERIA *p);
extern int mover_ventana_menu(int i, int min, int max, int regresa);
extern void ventana_cambia_datos_materia(void);
extern void cambia_datos_materia(struct MATERIA *p, int dato);
```

B.3.18. AYUDA.H

```
extern void cuadro_texto(char *p, int n, int color, int tipo, int x1, int x2, int y1, int y2);
extern void ventana_ayuda(char cadena[20][100], int inicio, int fin);
extern void llama_ayuda(int num);
```

B.3.19. CONSULTA.H

```
extern void pon_stack(struct STACK *ps, struct ELEMENTO el, int N);
extern void quita_stack(struct STACK *ps, struct ELEMENTO *el);
extern void particion_arreglo(struct ARREGLO *arreglo, int inferior, int superior, int *j);
extern void ordena_arreglo(struct ARREGLO *arreglo, int n);
extern void particion_salida(struct SALIDA *salida, int inferior, int superior, int *j);
extern void ordena_salida(struct SALIDA *salida, int n);
extern void particion_nombre_arreglo(struct ARREGLO *arreglo, int inferior, int superior, int *j);
extern void ordena_nombre_arreglo(struct ARREGLO *arreglo, int n);
extern void particion_nombre_salida(struct SALIDA *salida, int inferior, int superior, int *j);
extern void ordena_nombre_salida(struct SALIDA *salida, int n);
extern void ventana_nodo_lista_consulta(void);
extern int mover_pantallas_consulta(int i, int min, int salir, struct MATERIA *p, int cuenta);
extern void ventana_menu_consulta(void);
extern int ventanas_datos_mat(struct GRUPO *g, int inicio, int fin);
extern void mover_materias_datos(struct GRUPO *g, int cuenta);
extern void consulta_datos_materia(struct MATERIA *p, int dato, char semestre[10]);
extern int escoje_consulta_materia(struct MATERIA *p, int *i);
```

```
extern int consulta_materias(struct MATERIA *p, char semestre[10]);
extern void consulta_grupos(struct MATERIA *p, int *d, char semestre[10]);
extern void consulta_profesores(struct MATERIA *p, int *d, char semestre[10]);
extern void consulta_salones(struct MATERIA *p, int *d, char semestre[10]);
extern void consulta_examenes(struct MATERIA *p, int *d, char semestre[10]);
extern int consulta(int l, struct MATERIA *lista, int *d, char semestre[10]);
```

B.3.20. CUADROS.H

```
extern void cuadro(int l, int color);
extern void cuadrito(int n,int color,int tipo, int x1,int x2,int y1,int y2,char *p);
extern int opciones1(int l, int min, int max, int num);
extern int opciones2(int i, int min, int max);
extern void letra_especial(int x, int y, int largo, char *p);
extern void texto(char *p, int n, int color, int tipo, int x1, int x2, int y1, int y2);
extern void lee_cadena(char *p, int max, int v, int color, int tipo, int x1, int x2, int y1, int y2);
extern void pon(struct PILA *p, int i, int clase);
extern void quita(struct PILA *p);
```

B.3.21. ERRORES.H

```
extern void grupo_correcto(char cadena[100]);
extern void ventana_error(char cadena[20][100], int inicio, int fin);
extern void llama_error(int num);
extern int error_numero_entero(char cadena[100]);
extern int error_dia(char cadena[100]);
extern int error_mes(char cadena[100]);
extern int error_ano(char cadena[100]);
extern int hora_correcta(char cadena1[100], char cadena2[100], int num);
```

B.3.22. EXAMEN.H

```
extern int error_grupoexa_cambiar(struct EXAMEN *p, char cadena[100], int dato);
extern void crear_examen(struct MATERIA *p);
extern void agregar_examen(struct MATERIA *p);
extern void eliminar_examen(struct MATERIA *p);
extern void cambiar_examen(struct MATERIA *p);
extern void examen(int l, struct MATERIA *lista);
```

B.3.23. E_TODOS.H

```
extern void cuadros_consulta_examen(int tipo, int b);
extern void cambio_extraord(struct EXAMEN *e, int dato);
extern void cambio_final(struct EXAMEN *e, int dato);
extern void cambio_lista_examen(struct EXAMEN *e, int dato);
extern void busca_examen_final(struct MATERIA *p, int numero, struct ARREGLO *arreglo);
extern void busca_examen_todo(struct MATERIA *p, struct ARREGLO *arreglo, int tipo);
extern void busca_examen_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_examen_final_cuenta(struct MATERIA *p, int numero, int *cuenta);
extern void busca_examen_todo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_examen_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern int ventanas_examen_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo, int *max, int z, int b);
extern int examen_todos(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int b, int tipo, int cuenta);
extern void consulta_datos_examen_extraord(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10]);
extern void consulta_datos_examen_final(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10]);
```

B.3.24. GRAFICO.H

```
#define         ENTER           0X1C0D
#define         BACKSPACE       0X0E08
#define         DELETE          0X5300
#define         HOME            0X4700
#define         END             0X4F00
#define         TAB             0X0F09
#define         ESC             0X011B
#define         ARRIBA          0X4800
#define         ABAJO           0X5000
#define         DERECHA         0X4D00
#define         IZQUIERDA       0X4B00
#define         PGUP            0X4900
#define         PGDN            0X5100
#define         F1              0X3B00
#define         bajo(f)         ((f) & 0xff)
#define         alto(f)         (bajo(f >> 8))
#define  then
#define         do

typedef struct VENTANA {
                int     x1, x2, y1, y2;
              };

typedef struct PILA {
                unsigned long   ventana;
                int             num;
                unsigned long   tam;
              };
extern struct VENTANA vent[400];
extern struct PILA tope[400];
extern "C"
{
 int getxm(unsigned long source, void far *dest, unsigned len);
 int putxm(void far *source, unsigned long dest, unsigned len);
}
extern void pon(struct PILA p[], int i, int clase);
extern void quita(struct PILA p[]);
extern void ventanas(struct VENTANA v[]);
```

B.3.25. GRUPOS.H

```
extern void lee_nodo_grupos(struct GRUPO *g, struct GRUPO *s);
extern int salir_lee_grupo(void);
extern void ventana_nodo_lista_grupo(void);
extern int ventanas_ponen_grupos(struct GRUPO *g, int inicio, int fin, int *max);
extern int mover_pantallas_grupos(int i, int min, int salir, struct GRUPO *g, int cuenta);
extern void cambio_lista_grupos(struct GRUPO *g, int dato);
extern void crear_grupos(struct MATERIA *p);
extern void agregar_grupos(struct MATERIA *p);
extern void eliminar_grupos(struct MATERIA *p);
extern void cambiar_grupos(struct MATERIA *p);
extern void grupos(int i, struct MATERIA *lista);
```

B.3.26. G_TODOS.H

```
extern struct MATERIA * busca_materia(struct MATERIA *p, int materia);
extern int ventanas_ponen_datos(struct GRUPO *g, int inicio, int fin, char nombreprofe[100]);
extern int ventanas_ponen_datos_salones(struct GRUPO *g, int inicio, int fin, long int numesalon);
extern void busca_grupo_tipo(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int clase);
extern void busca_grupo_cadena_tipo(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int clase);
extern void busca_grupo_todo_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_grupo_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_grupo_teolab(struct MATERIA *p, int numero, struct ARREGLO *arreglo);
extern void busca_grupo_cadena_teolab(struct MATERIA *p, int numero, struct ARREGLO *arreglo);
extern void busca_gteolab_todo(struct MATERIA *p, struct ARREGLO *arreglo);
extern void busca_gteolab_materia(struct MATERIA *p, struct ARREGLO *arreglo);
extern void busca_grupo_tipo_cuenta(struct MATERIA *p, int numero, int clase, int *cuenta);
extern void busca_grupo_cadena_tipo_cuenta(struct MATERIA *p, int numero, int clase, int *cuenta);
extern void busca_grupo_todo_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_grupo_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_grupo_teolab_cuenta(struct MATERIA *p, int numero, int *cuenta);
extern void busca_grupo_cadena_teolab_cuenta(struct MATERIA *p, int numero, int *cuenta);
extern void busca_gteolab_todo_cuenta(struct MATERIA *p, int *cuenta);
extern void busca_gteolab_materia_cuenta(struct MATERIA *p, int *cuenta);
extern void cuadros_grupos_todos(int tipo);
extern int ventanas_grupos_todos(struct MATERIA *p,int inicio, int fin, struct ARREGLO *arreglo,int *max,int tipo);
extern int grupos_teoria(struct MATERIA *p, int numero, struct ARREGLO *arreglo, int b, int tipo, int cuenta);
extern void consulta_datos_grupos_todos(struct MATERIA *p, struct ARREGLO *arreglo,int a, int b, char semestre[10]);
extern int busca_salon_arreglo(struct MATERIA *p, struct ARREGLO *arreglo, long int numero, int j);
extern void cuadros_salones_todos(int tipo);
extern void busca_salon_tipo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo, int clase);
extern void busca_salon_cadena_tipo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo, int clase);
extern void busca_salon_todo_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_salon_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_salon(struct MATERIA *p, long int numero, struct ARREGLO *arreglo);
extern void busca_salon_cadena_todo(struct MATERIA *p, long int numero, struct ARREGLO *arreglo);
extern void busca_salon_todo(struct MATERIA *p, struct ARREGLO *arreglo);
extern void busca_salon_materia_todo(struct MATERIA *p, struct ARREGLO *arreglo);
extern int ventanas_salones_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo, int *max, int tipo);
extern int salones_todos(struct MATERIA *p, long int numero, struct ARREGLO * arreglo, int b, int tipo, int cuenta);
extern int mover_datos_salones_todos(struct GRUPO *g, long int numesalon, int cuenta);
extern int busca_salon_resultado(struct MATERIA *p, long int salon, int materia[50]);
extern void consulta_datos_salones_todos(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10]);
extern void busca_salon_tipo_cuenta(struct MATERIA *p, long int numero, int clase, int *cuenta);
extern void busca_salon_cadena_tipo_cuenta(struct MATERIA *p, long int numero, int clase, int *cuenta);
extern void busca_salon_todo_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
```

```
extern void busca_salon_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_salon_cuenta(struct MATERIA *p, long int numero, int *cuenta);
extern void busca_salon_cadena_todo_cuenta(struct MATERIA *p, long int numero, int *cuenta);
extern void busca_salon_todo_cuenta(struct MATERIA *p, int *cuenta);
extern void busca_salon_materia_todo_cuenta(struct MATERIA *p, int *cuenta);
extern void busca_salon_tipo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida, int clase);
extern void busca_salon_cadena_tipo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida, int clase);
extern void busca_salon_todo_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase);
extern void busca_salon_materia_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase);
extern void busca_salon_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida);
extern void busca_salon_cadena_todo_imprime(struct MATERIA *p, long int numero, struct SALIDA *salida);
extern void busca_salon_todo_imprime(struct MATERIA *p, struct SALIDA *salida);
extern void busca_salon_materia_todo_imprime(struct MATERIA *p, struct SALIDA *salida);
```

B.3.27. IMPRIME.H

```
extern void imprimir(struct MATERIA *p, struct ARREGLO *arreglo, int tipo, char semestre[10]);
extern void imprime_todo(struct MATERIA *p, struct ARREGLO *arreglo, int tipo, int condicion, char semestre[10]);
extern void imprimir_profe_salida(struct MATERIA *p, char semestre[10],int b, int tip, char profesor[100], int cuenta);
extern void imprimir_salon_salida(struct MATERIA *p, char semestre[10],int b, int tip, long int numesalon, int cuenta);
```

B.3.28. OPER.H

```
#define  tran
// Estructura de la lista enlazada EXAMEN.
typedef struct EXAMEN
        {
        int     tipoesa;                // Tipo de exámen.
        int     tipoprofe;              // Nombre del prof. de exá. final.
        int     tipo;                   // Nombre del prof. de exá. extra.
        int     tiposalon;              // Clasificación del salón.
        long int numesalon;             // Salón.
        int     numgrupo;               // Grupo.
        char    hora1[10];              // hora de inicio de exámen.
        int     dia1;                   // Fecha de exámen.
        int     dia2;
        int     mes1;
        int     mes2;
        int     ano1;
        int     ano2;
        char    nombreprofe[40];        // Nombre del profesor titular.
        char    profe[40];              // Nombre del profesor suplente.
        char    rfc[15];                // R.F.C. del profesor titular.
        char    rfc2[15];               // R.F.C. del profesor suplente.
        int     num;                    // Número de elementos en la lista.
        struct EXAMEN *sig;
        };

// Estructura de la lista enlazada GRUPO.
typedef struct GRUPO
        {
        int     numgrupo;               // Grupo.
        long int numesalon;             // Salón.
```

```
    int     tipo;                   // Clasificación del salón.
    char    nombreprofe[40];        // Nombre del profesor.
    char    rfc[15];                // R.F.C. del profesor.
    int     tipoprofe;              // Nombramiento del profesor.
    char    telofc[20];             // Tel. de la oficina del prof.
    char    telcasa[20];            // Tel. de la casa del profesor.
    char    antunam[10];            // Antigüedad del prof. en la UNAM.
    char    antenep[10];            // Antigüedad del prof. en la ENEP.
    int     dias[10];               // Horario de clase.
    char    hora1[10];
    char    hora2[10];
    int     numalumnos;             // Total de alumnos inscritos.
    int     clasegrupo;             // Grupo de lab. ó de teoría.
    int     num;                    // Número de elemtos en la lista.
    struct GRUPO  apun;
    };

// Estructura de la lista enlazada MATERIA
typedef  struct MATERIA
        {
            char            nombre[30];     // Nombre de la materia.
            int             clave;          // Clave de la materia.
            int             lab;            // Opción de laboratorio.
            int             tipomat;        // Tipo de materia.
            int             num;            // Número de elementos en la lista.
            struct MATERIA  *materias;      // Apuntador al siguiente nodo de la lista
            struct GRUPO    *grupos;        // Apuntador a la estructura GRUPO
            struct EXAMEN   *examen;        // Apuntador a la estructura EXAMEN
        };

typedef struct ARREGLO
            {
            int grupo;
            int materia;
            long int numero;
            char nombre[100];
            };

typedef struct SALIDA
            {
            int grupo;
            int materia;
            long int numero;
            char nombre[100];
            };

extern char semestre[10];
extern void crea_lista_materias(struct MATERIA *p);
extern void pon_lista_materias(struct MATERIA *p,struct MATERIA q);
extern void lee_lista_materias(struct MATERIA *p, char *nombre, char semestre[10]);
extern int quita_lista_materias(struct MATERIA *p, int n);
extern int cambio_lista_materias(struct MATERIA *p, int n);
extern void guarda_lista_materias(struct MATERIA *p, char *nombre, char semestre[10]);
extern void guarda_lista_materias_todo(struct MATERIA *p, char *nombre);
extern struct GRUPO * crea_lista_grupos(void);
extern void pon_grupos(struct GRUPO *p, struct GRUPO g);
extern int quita_grupos(struct GRUPO *g, int n);
extern struct EXAMEN * crea_lista_examen(void);
extern void pon_examen(struct EXAMEN *g, struct EXAMEN q);
extern int quita_examen(struct EXAMEN *g, int n);
extern int busca_materia_clave(struct MATERIA *p, int clave);
extern int busca_materia_nombre(struct MATERIA *p, char cadena[100]);
```

B.3.29. P_TODOS.H

```
extern void cuadros_profesor_todos(int tipo);
extern int busca_profesor_arreglo(struct MATERIA *p, struct ARREGLO *arreglo, char profesor[100], int );
extern void busca_profe_tipo(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int clase);
extern void busca_profe_cadena_tipo(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int clase);
extern void busca_profe_todos_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_profe_materia_tipo(struct MATERIA *p, struct ARREGLO *arreglo, int clase);
extern void busca_profe(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo);
extern void busca_profe_cadena(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo);
extern void busca_profe_todos(struct MATERIA *p, struct ARREGLO *arreglo);
extern void busca_profe_todos_materia(struct MATERIA *p, struct ARREGLO *arreglo);
extern int ventanas_profesor_todos(struct MATERIA *p, int inicio, int fin, struct ARREGLO *arreglo,int *max, int tipo);
extern int profesor_todos(struct MATERIA *p, char profesor[100], struct ARREGLO *arreglo, int b, int tipo, int cuenta);
extern int mover_datos_profesor_todos(struct GRUPO *g, char profesor[100], int cuenta);
extern int busca_profesor_resultado(struct MATERIA *p, char profesor[100], int materia[50]);
extern void consulta_datos_profesor_todos(struct MATERIA *p, struct ARREGLO *arreglo, int a, int y, char semestre[10]);
extern void busca_profe_tipo_cuenta(struct MATERIA *p, char profesor[100], int clase, int *cuenta);
extern void busca_profe_cadena_tipo_cuenta(struct MATERIA *p, char profesor[100], int clase, int *cuenta);
extern void busca_profe_todos_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_profe_materia_tipo_cuenta(struct MATERIA *p, int clase, int *cuenta);
extern void busca_profe_cuenta(struct MATERIA *p, char profesor[100], int *cuenta);
extern void busca_profe_cadena_cuenta(struct MATERIA *p, char profesor[100], int *cuenta);
extern void busca_profe_todos_cuenta(struct MATERIA *p, int *cuenta);
extern void busca_profe_todos_materia_cuenta(struct MATERIA *p, int *cuenta);
extern void busca_profe_tipo_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida, int clase);
extern void busca_profe_cadena_tipo_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida, int clase);
extern void busca_profe_todos_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase);
extern void busca_profe_materia_tipo_imprime(struct MATERIA *p, struct SALIDA *salida, int clase);
extern void busca_profe_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida);
extern void busca_profe_cadena_imprime(struct MATERIA *p, char profesor[100], struct SALIDA *salida);
extern void busca_profe_todos_imprime(struct MATERIA *p, struct SALIDA *salida);
extern void busca_profe_todos_materia_imprime(struct MATERIA *p, struct SALIDA *salida);
```

B.3.30. VENTANAS.H

```
extern void cuadro(int l, int color);
extern void cuadrito(int n,int color,int tipo, int x1,int x2,int y1,int y2,char *p);
extern int opciones1(int l, int min, int max);
extern int opciones2(int l, int min, int max);
extern void letra_especial(int x, int y, int largo, char *p);
extern void texto(char *p, int n, int color, int tipo, int x1, int x2, int y1, int y2);
extern void lee_cadena(char *p, int max, int v, int color, int tipo, int x1, int x2, int y1, int y2);
```

B.3.31. VENTDOS.H

```
extern ventdos(struct VENTANA v[]);
```

GLOSARIO

APUNTADOR. Es una variable que contiene una dirección de memoria. Normalmente, esa dirección es la posición de otra variable de memoria. Si una variable contiene la dirección de otra variable, entonces se dice que la primera apunta a la segunda.

ARBOL. Es una estructura de datos no lineal. Esta estructura se usa principalmente para representar datos con una relación jerárquica entre sus elementos.

ASCII (American Standard Code for Information Interchange). Es un código de 7 bits que permite hasta 128 caracteres.

BIBLIOTECA. Es un archivo que contiene las funciones estándar que se pueden usar en los programas. Esas funciones incluyen todas las operaciones de E/S, así como otras rutinas útiles.

CODIGO FUENTE. Es el texto de un programa que el usuario puede leer.

CODIGO OBJETO. Es la traducción del código fuente de un programa a código máquina, que es el que la computadora puede leer y ejecutar directamente. El código objeto es la entrada al ligador.

COLA. Es una colección ordenada de elementos a partir de la cual se pueden eliminar elementos de un extremo y en la cual también se pueden agregar elementos en el otro extremo.

DESCRIPTOR. Controlan todos los aspectos de bloques de memoria o segmentos en las operaciones del modo protegido. Cada descriptor contiene una dirección de memoria física base para el segmento, la longitud del segmento, y algunos bits que definen los atributos del segmento.

GDT (Global descriptor table). Contiene descriptores empleados para definir el ambiente de todo el computador.

HARDWARE. Es el nombre que se le da al conjunto de circuitos electrónicos, junto con la memoria y los dispositivos de Entrada/Salida que componen un computador.

IDT (Interrupt descriptor table). Esta tabla es usada para el manejo de las interrupciones.

INTERRUPCIONES. Son cambios en el flujo de control, no ocasionados por el programa que se ejecuta, sino por alguna otra cosa normalmente relacionada con la E/S.

LDT (lLocal descriptor table). Contiene descriptores usados para localizar el ambiente de los programas.

LIGADOR. Es un programa que enlaza funciones compiladas por separado para producir un solo programa. La salida del ligador es un programa ejecutable.

LISTA ENLAZADA. Es una colección lineal de elementos, llamados nodos, donde el orden de los mismos se establece mediante apuntadores. Cada nodo se divide en dos partes, una contiene la información asociada al elemento, y la otra contiene la dirección del siguiente nodo de la lista.

PILA. Es una colección ordenada de elementos en la cual en un extremo se pueden insertar nuevos elementos y de la cual se pueden retirar otros.

PROJECT. Se refiere a los programas de archivos múltiples. Los programas demasiado largos se parten en pequeños archivos, se compila cada archivo y se enlazan juntos.

REGISTRO. Es un conjunto de elementos relacionados entre sí, cada uno de los cuales recibe el nombre de campo o atributo.

SEGMENTO. Son espacios de direcciones completamente independientes. Cada segmento consta de una secuencia lineal de direcciones que van desde 0 a un determinado máximo. La longitud de cada segmento puede ser cualquiera situada entre 0 y el máximo permitido. En la arquitectura del 80386 la longitud de un segmento varía de 1 byte hasta 4 gigabytes.

SOFTWARE. Se refiere a los algoritmos (instrucciones detalladas que dicen cómo hacer algo) y sus representaciones en la computadora.

TABLA DESCRIPTORA. Contiene información acerca de los segmentos, incluyendo las direcciones de inicio. Existen tres tipos de tablas descriptoras, GDT, LDT, IDT.

V86. Se refiere al modo virtual del microprocesador 8086.

BIBLIOGRAFIA

[1]. DUNCAN, RAY: "Extending DOS. A Programmers's Guide to Protected-Mode DOS". Segunda edición. Addison Wesley, 1992.

[2]. GILES, WILLIAM B.: "Assembly Language Programming for the Intel 80xxx Family". Primera edición. Maxwell Macmillan International, N.Y., 1991.

[3]. KORTH, HENRY F. y SILBERSCHATZ ABRAHAM.: "Fundamentos de Bases de Datos". Segunda edición. McGraw Hill, España 1993.

[4]. LIPSCHUTZ, SEYMOUR: "Estructura de Datos". Primera edición. McGraw Hill, 1988.

[5]. MOSICH, DONNA, SHAMMAS, NAMIR y FLAMIG, BRYAN: "Advanced Turdo C Programmer's Guide". Primera edición. John Wiley, 1988.

[6]. MURRAY, WILLIAM H. y PAPPAS, CHRIS: "80386/80286 Programación en Lenguaje Ensamblador". Primera edición. McGraw Hill, México, 1989.

[7]. MURRAY, WILLIAM H. y PAPPAS, CHRIS: "Turbo C++ Professional". McGraw Hill, México, 1990.

[8]. SCHILDT, HERBERT: "Turbo C/C++ Manual de refencia". Primera edición. McGraw Hill, 1992.

[9]. SCHILDT, HERBERT: "Turbo C". Segunda edición. McGraw Hill, 1989.

[10]. SWAN, TOM: "Mastering Turbo Assembler". Primera edición. Hayden Books, 1989.

[11].TANENBAUM, ANDREW: "Organización de Computadoras: un enfoque estructurado". Segunda edición. Prentice Hall, México, 1988.

[12]. TENENBAUM, AARON M. y LANGSAM, YEDIDYAH: "Data Structures Using C". Primera edición. Prentice Hall, Englewood Cliffs, N.J., 1990.

[13]. WYATT, ALLEN L.: "Advanced Assembly Language". Primera edición. Que, 1992.

[14]. WYATT, ALLEN L.: "Using Assemby Language". Segunda edición. Que, 1990.

[15]. "BORLAND C++ Programmer's Guide. V. 3.1". Borland International, U.S.A., 1992.

[16]. "BORLAND C++ User's Guide. V. 3.1". Borland International, U.S.A., 1992.

[17]. "BORLAND C++ Library Reference. V. 3.1". Borland International, U.S.A., 1992.

[18]. "Turbo Assembler. V. 3.0". Borland International, U.S.A., 1992.