UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

ELABORACION Y APLICACION DE FUNCIONES DESARROLLADAS EN AMBIENTE WINDOWS ATRAVES DE C++

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

P r e s e n t a :

MARCO ANTONIO AVILA MORALES

San Juan de Aragón Edo. de Méx. 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CON CARIÑO Y GRATITUD
A MIS PADRES.

A LA MEMORIA DE MI
ABUELITA AMELIA E.
Y ABUELITOS.

A MIS HERMANOS.

POR EL APOYO Y
CARIÑO QUE SIEMPRE
RECIBI DE TODOS ELLOS.

AL PROFESOR
ING.ERNESTO PEÑALOZA.
DIRECTOR DE TESIS.

A LOS REVISORES.

A LA ESCUELA Y
PROFESORES.

CON GRATITUD Y
AGRADECIMIENTO.

AL H. JURADO
RESPETUOSAMENTE.

COMPAÑEROS
Y AMIGOS.

TEMARIO.

Titulo I.-

ELABORACIÓN Y APLICACIÓN DE FUNCIONES DESARROLLADAS EN AMBIENTE WINDOWS A TRAVÉS DE C++.

Objetivo.
Introducción.

Capítulo I.- AMBIENTE GRÁFICO DE WINDOWS.

I.1.- Ambiente gráfico de Windows.
I.2-- Funciones gráficas de Windows utilizadas en el desarrollo del sistema.

Capítulo II.- DESARROLLO DE FUNCIONES.

II-1-- Estructura , objetivo y Programación de las funciones a través de "C".

Capítulo III.- DESARROLLO DE UNA APLICACIÓN UTILIZANDO LAS FUNCIONES DESARROLLADAS EN EL CAPITULO II.

III.1.- Estructura de la aplicación.
III.2.- Programación de los módulos.

Capítulo IV.- FUNCIONAMIENTO DEL TUTORIAL DESARROLLADO.

IV.1.- Función de los módulos que componen al tutorial.

Capítulo V.- FUNCIONAMIENTO DEL SOFTWARE DE APOYO.

V.1.- Función del programa de apoyo.

- Listado de la aplicación.
- Apendice.
- Glosario.
- Bibliografía.

OBJETIVO:

La finalidad es presentar la manera como se pueden aprovechar las funciones que incorpora el ambiente Windows junto con la programación en C, para desarrollar otras funciones a partir de éstas que sirvan de apoyo para el desarrollo de aplicaciones.

JUSTIFICACION:

Actualmente la computadora esta presente en la solución de toda clase de actividades y problemas, y dependiendo del tipo de actividad a realizar, ó problema a resolver, será la selección de la herramienta más apropiada para efectuarla. Hojas de cálculo, procesadores de palabra, bases de datos, lenguajes de programación, etc, son solo algunas herramientas entre las que podemos elegir la más apropiada para nuestra necesidad.

Hoy en día existen gran cantidad de sistemas los cuales nos enseñan a utilizar algún programa ó paquete en especial. A estos sistemas se les denominan "Tutoriales". Estos tutoriales son muy prácticos, ya que a través de ellos aprendemos cosas nuevas de una manera muy diferente a como lo sería si lo leyéramos de un libro. Esto se debe a que en un tutorial podemos explicar los conceptos apoyándonos en las ventajas que nos brinda la computadora, como lo son la animación, los sonidos, etc.

Partiendo de esta opción, desarrolle algunas funciones básicas para el desarrollo de tutoriales, esto con el fin de facilitar el desarrollo de más programas tutoriales de temas de interés para el usuario. Los tutoriales que pueden generarse con dichas funciones deben de basar su técnica para enseñar los temas, principalmente en la animación.

Dichas funciones están desarrolladas bajo ambiente Windows a través del lenguaje de programación C, por lo cual estas también servirán como apoyo para el aprendizaje de la programación de las funciones que incorpora Windows para la elaboración de otra funciones a través de un lenguaje como lo es C.

Para mostrar la utilidad de las funciones desarrolladas y de las funciones propias de Windows, elabore un tutorial, el cual enseña al usuario de una forma amena los conceptos y funcionamiento básicos de una computadora, y en el cual el usuario interactúa continuamente con la máquina.

INTRODUCCION:

Al desarrollar una nueva aplicación, nos encontramos con una serie de factores muy importantes como son el ambiente de trabajo, así como el lenguaje o software a utilizar en la elaboración del sistema. La adecuada selección de estos factores, se vera reflejada en la calidad del software.

Actualmente la plataforma de trabajo Windows es de las mas utilizadas debido a sus características. Windows es una ambiente gráfico, en el cual el usuario interactúa con las aplicaciones de una manera muy sencilla. Windows nos permite explotar todo su potencial a través de funciones que trae incorporadas, pero para lo cual requerimos de algún lenguaje de programación por medio del cual podamos programarlas.

En el software que aquí presento, desarrollo una serie de funciones a partir de las que nos ofrece Windows, apoyándome en el lenguaje de programación "C". Estas funciones muestran la manera como podemos aprovechar las funciones de Windows, para desarrollar aplicaciones que se adapten a nuestras necesidades.

Este libro se presenta junto con dos aplicaciones, la primera llamada "MANUAL", la cual contiene a todas las funciones que he desarrollado, presentando el código de cada una, explicando cada línea que la compone, así como su ejecución. La segunda aplicación es un ejemplo de las funciones aquí desarrolladas llamada "TUTORIAL", el cual no solo pretende ser un ejemplo práctico del uso y manejo de funciones, su objetivo va un poco más haya, intentando ser realmente un software de apoyo para el aprendizaje del tema que se aborda.

El "TUTORIAL", aborda cada tema de una manera general y sin entrar en muchos detalles, y si desea conocer mas de algún tema, tendrá que consultar material mas especializado.

ACERCA DEL TUTORIAL....

En muchas ocasiones al estudiar algún tema en algún libro, sobre todo temas que involucran gráficas o procesos internos en alguna máquina ,nos enfrentamos al problema de no entender claramente algunos puntos, debido a que no logramos formarnos una imagen de lo que estamos leyendo. Una solución a este problema es introducir figuras y gráficas en el texto, y una mejor solución sería el poder observar de manera animada lo que se nos esta explicando.

Partiendo de este problema se desarrollo este tutorial, el cual es un apoyo para entender de una manera mas fácil, rápida y divertida los temas que nos interesan. En nuestro caso el tutorial está enfocado al estudio de la computadora, como es su organización interna, es decir , la manera como esta dividida , la forma como están interrelacionados sus componentes , así como el manejo de los datos, información y señales

Como se mencionó anteriormente, los temas serán tratados sin muchos detalles, tratando de mostrar solamente los puntos más representativos. A lo largo del tutorial se encontrará con términos técnicos propios del tema, para los cuales existe la opcion de "Conceptos básicos" y la de "Ayuda ", en las cuales encontrará una breve explicación de su significado.

El sistema se encuentra dividido en tres módulos, el primer módulo llamado "Micro computadora", nos habla de la computadora como una unidad , en esta parte observamos el funcionamiento de la máquina manejándose como una unidad,. El segundo módulo llamado "Módulos", nos presenta a la computadora separada en sus principales componentes, y a su vez cada componentes nos muestra información especifica. El tercer módulo se llama "Función" , es semejante al módulo dos, pero en este el objetivo es mostrar el flujo de los datos, de la información y de las señales de control.

A lo largo del tutorial se encontrará con una serie de evaluaciones al final de cada tema, cuyo fin es reafirmar los conceptos correspondientes a esa lección.

Capítulo 1.- AMBIENTE GRÁFICO DE WINDOWS.

1.1.- Ambiente gráfico de Windows.

Windows es un entorno de ventanas multitarea basado en gráficos. El entorno gráfico de Windows, debido a sus características como son la transferencia de datos entra programas, la ejecución de más de un programa a la vez, la posibilidad de establecer enlaces dinámicos entre programas en ejecución , la mezcla de texto y gráficos en la pantalla, el manejo de una interfaz que permite seleccionar una opción a través del mouse y activándola a con el mismo, etc. facilita enormemente el trabajo del usuario.

Windows ofrece una gran riqueza de recursos al programador, que incluyen un potente mecanismo de gráficos predefinidos como una serie de funciones que pueden utilizar las aplicaciones.

Para el desarrollo de aplicactones, Windows ofrece abundantes subrutinas incorporadas que permiten la fácil implementación de menús instantáneos, barras de desplazamiento, cuadros de diálogo, iconos y otras muchas características de un interfaz gráfico de fácil manejo para el usuario. Haciendo uso del extenso lenguaje de programación gráfica ofrecido con Windows, una aplicación puede fácilmente dar formato y mostrar texto usando diversas fuentes y tamaños.

Windows también ofrece al programador de aplicaciones el tratamiento de la salida de vídeo, el teclado, el ratón, la impresora, el puerto serie y los temporizadores del sistema de forma independiente de los dispositivos. Esto hace posible que un mismo programa se ejecute en forma idéntica en diversas configuraciones de hardware.

Respecto al software . . .

El lenguaje de alto nivel C/C++, ha sido descrito como lo mejor ; presenta una combinación de lenguaje ensamblador y lenguaje de alto nivel. C/C++ ofrece a los programadores el fácil acceso al hardware proporcionado por el ensamblador, y las funciones ofrecidas por un lenguaje de alto nivel. Con la aparición de Windows, el usuario tiene la capacidad de la multitarea, pero lo que resulta de mayor importancia son las interfaces de dispositivos gráficos (GDI) ofrecidos por Windows.

Con las funciones GDI de Windows, el programador puede escribir una aplicación que tenga apariencia familiar al usuario (con menús, barras de desplazamiento, cuadros de mensaje y demás). Además, se puede modificar el tamaño de la aplicación, mover su ventana, convertirla en un icono, ponerla en segundo plano, y más cosas, mientras que otros programas están trabajando simultáneamente. Aún más importante son las funciones GDI que hacen posible que una aplicación se comunique con otra.

Interfaz gráfico de Windows.

En comparación con el entorno MS-DOS convencional, el entorno operativo de Windows ofrece ventajas considerables tanto a los usuarios como a los programadores. Las tres principales características interfaz gráfico de usuario , la multitarea y la independencia del hardware, no son nuevas, lo que resulta innovador es el intento de combinarlas en un único entorno operativo de microordenador.

De las tres características principales que ofrece Windows, el interfaz de usuario orientado a gráficos y estandarizado es la que presenta mayor interés, y, desde luego, es la más importante para el usuario.

El interfaz de usuario consistente utiliza dibujos , o iconos para representar unidades de disco, archivos, subdirectorios y la mayoría de las órdenes y acciones del sistema operativo. Los programas se identifican mediante barras de títulos, y se accede a muchas de las funciones básicas de manipulación de archivos mediante menús sin necesidad de hacer otra cosa más que apuntar y pulsar un botón del ratón. La mayoría de los programadores de Windows poseen un interfaz de teclado y otros de ratón.

Todas las aplicaciones Windows tienen la misma apariencia, esto facilita el aprendizaje de nuevas aplicaciones al usuario.

El sistema multitarea...

Windows nos ofrece un entorno multitarea, permite al usuario tener varias aplicaciones, o varias instancias de una misma aplicación, ejecutándose de forma concurrente, aunque realmente sólo una de ellas puede utilizar al procesador en un instante dado. La diferencia entre una tarea que se está procesando y una tarea que simplemente se está ejecutando es importante. Existen también un tercer estado en que se puede encontrar una aplicación, denominado estado activo. Una aplicación activa es la que recibe la atención del usuario. De la misma forma que sólo se puede estar procesando una aplicación en un determinado instante, sólo puede haber una aplicación activa en un instante dado.

Sin embargo puede existir cualquier número de tareas ejecutándose de forma concurrente. Es responsabilidad de Windows el reparto del tiempo de uso del microprocesador. Windows controla el uso compartido del microprocesador utilizando colas de entrada y mensajes. Con Windows los recursos del ordenador, incluyendo los dispositivos de entrada y salida, la memoria, el sistema de video y el CPU tienen que ser compartidos.

El control de la memoria.

La memoria es uno de los recursos importantes compartidos con Windows. Con más de una aplicación ejecutándose al mismo tiempo, las aplicaciones deben cooperar para compartir la memoria a fin de no agotar este recurso. Además, conforme empiezan a ejecutarse unos programas a la vez que otros finalizan su ejecución, la memoria puede quedar fragmentada. Windows es capaz de consolidar el espacio de memoria libre moviendo bloques de código y datos en memoria, de esta manera garantiza que cada aplicación disponga de la memoria que necesite.

Windows permite a las aplicaciones "sobre utilizar" la memoria. Esto es , una aplicación puede contener más código del que realmente cabe en memoria en un determinado momento. Windows puede retirar bloques de código de la memoria y posteriormente volver a cargar dicho código del archivo .exe del programa en cuestión.

Bajo otras circunstancias, un usuario puede tener varias instancias, o copias, de un mismo programa ejecutándose concurrentemente. Para ahorrar espacio de memoria, Windows comparte el mismo código. Los programas que se ejecutan con Windows pueden incluso compartir rutinas situadas en otros archivos .EXE. Los archivos que contienen esas rutinas compartidas se denominan "*bibliotecas de enlace dinámico*". Windows dispone del mecanismo para enlazar el programa con las rutinas de la biblioteca de enlace dinámico en tiempo de ejecución (Windows en sí mismo es un conjunto de bibliotecas de enlace dinámico). Para facilitar todo esto, los programas de Windows utilizan un nuevo formato para los archivos .EXE, denominado "nuevo formato ejecutable". Estos archivos incluyen la información que necesita Windows para gestionar los segmentos de código y datos para realizar el enlazado dinámico.

La ejecución de los programas.

En la ejecución de un programa, Windows realiza tres tareas diferentes para proporcionar a la aplicación un entorno de ejecución. Primero carga y ejecuta el programa, segundo, gestiona la memoria de la computadora de manera que tanto la aplicación particular como el resto de aplicaciones Windows, que en ese momento se estén ejecutando, tengan acceso a la memoria y a los demás recursos del sistema que puedan necesitar. Tercero, gestiona la mayoría de las entradas de bajo nivel que la aplicación realice.

Activación de una aplicación.

Cuando el usuario selecciona el icono de una aplicación, Windows asigna una cierta cantidad de memoria al programa. Después carga el código ejecutable y determina una pila local. A continuación, se

ocupa de cargar los datos de la aplicación y de proporcionar cualquier recurso particular que sea requerido en tiempo de ejecución. Estos recursos especiales son propios del entorno Windows e incluyen objetos tales como un menú del sistema, abreviaturas de teclado, cuadros de diálogo, cuadros de mensaje, iconos y mapas de bits. No se deben confundir los recursos Windows (iconos, cursores, cuadros de mensaje, cuadros de dialogo, las fuentes, matrices de puntos, lápices, pinceles, etc.) con los recursos del sistema, como la memoria, el acceso a disco, las entradas de ratón y de teclado, y el acceso a la pantalla, entre otros.

Entrada.

Windows Proporciona un extenso conjunto de servicios de entrada y salida que permiten aislar la aplicación de la máquina física. Esto facilita el desarrollo de aplicaciones, puesto que no es necesario tomar en consideración las diferencias entre computadoras, monitores, impresoras y demás dispositivos. Windows está constantemente sondeando el reloj, el teclado el ratón. Cuando se produce un suceso en alguno de estos dispositivos, Windows genera un mensaje que coloca en la cola del sistema. La cola del sistema no es más que una lista "primero en entrar primero en salir" (FIFO) que contiene los mensajes correspondientes a los sucesos del teclado y el ratón. Los sucesos del reloj se almacenan de manera aparte.

Windows también decide que mensaje debe ir a que aplicación. No se debe olvidar que puede haber más de una aplicación qué se esté ejecutando con Windows al mismo tiempo. Un programa puede , durante su ejecución, recibir mensajes de otras aplicaciones, o incluso del propio Windows. Existe una cola de mensajes asociada explícitamente con cada aplicación, que Windows se encarga de gestionar.

El paso de mensajes.

La clave para entender la programación en Windows es el paso de mensajes. Las entradas de la aplicación son los mensajes que toma de su cola de aplicación, gestionada por Windows para el programa correspondiente. Estos mensajes Windows envía estos mensajes y la aplicación los recibe una vez que estos son registrados cosa que no ocurre bajo DOS puesto que es inverso el proceso. La aplicación también puede enviar mensajes a otras aplicaciones o al propio Windows. Además, puede recibir mensajes de otras aplicaciones y del propio Windows.

Algunos mensajes típicos enviados por Windows a las aplicaciones en tiempo de ejecución son los siguientes:

WM_COMMAND	Indica una selección de alguna opción del menú.
WM_LBUTTONDOWN	Pulsación del botón izquierdo del ratón.
WM_LBUTTONUP	Liberación del ratón izquierdo del ratón.
WM_MOUSEMOVE	Desplazamiento del ratón.
WM_KEYDOWN	Pulsación en el teclado.
WM_SYSCOMMAND	Una selección en el menú del sistema.
WM_INITMENUPOPUP	Se va a visualizar un menú.
WM_DESTROY	Se va a destruir la ventana principal.
WM_TIMER	Un mensaje del reloj.

Nota: Todos los identificadores de mensajes comienzan con los caracteres WM_.

El bucle de mensajes.

Las aplicaciones toman los mensajes de la correspondiente cola. Los programas contienen un bucle que se encarga de leer los mensajes, denominado bucle de mensajes. El bucle de mensajes está contenido en la función WinMain () .Todas las aplicaciones Windows contienen una función WinMain, esté es el punto de partida de la aplicación.

Cuando la aplicación detecta un mensaje, abandona temporalmente el bucle para realizar la acción que juzgue oportuna y vuelva después al bucle de mensajes a la espera de nuevos mensajes. El bucle de mensajes llama a la función predefinida de Windows GetMessage() para comprobar si existe algún mensaje pendiente. Cuando se llama a la función GetMessage(), Windows comprueba primero si existe algún mensaje en la cola de la aplicación, a continuación busca en la cola del sistema si existe algún mensaje del teclado o el ratón y, por último, comprueba si hay algún mensaje del reloj.

El gestor de mensajes.

Al detectar la aplicación la llegada de un mensaje, abandona el bucle de mensajes para ejecutar las funciones correspondientes incluidas en el programa. Esto se realiza mediante el envío de un mensaje a través de Windows, a una parte especial del programa denominada "*gestor de mensajes*". Todas las aplicaciones Windows deben tener un gestor de mensajes. Windows asocia toda aplicación a una ventana y a su vez cada ventana con un gestor de mensajes, que se corresponde con un procedimiento de ventana incluido en la aplicación.

Bibliotecas de enlace dinámico.

Las funciones ejecutables del API (Interfaz de programas de aplicaciones) de Windows están contenidas en las bibliotecas de enlace dinámico (Dynamic - Link Libraries, DLL). Existen tres DLL que son : KERNELL.EXE, USER.EXE y GDI.EXE.

KERNELL.EXE proporciona las funciones para la gestión de ventanas, que constituyen el soporte del entorno Windows,. USER:EXE proporciona los servicios del sistema como la gestión de memoria o la gestión de la multitarea. GDI:EXE es el interfaz de dispositivos gráficos, proporciona una serie de rutinas gráficas.

Las tres bibliotecas API de Windows se denominan de enlace dinámico debido a que se enlazan a la aplicación en el momento de la ejecución, el enlace se realiza dinámicamente. El enlazador inserta un bloque de código objeto denominado biblioteca de importación.

La biblioteca de importación contiene información acerca de la ubicación de las funciones ejecutables incluidas en KERNELL.EXE , USER.EXE Y GDI.EXE. Esta información permite a las aplicaciones localizar y llamar a las funciones del API de Windows en tiempo de ejecución. La biblioteca de importación para Borland C++ se llama IMPORT.LIB.

El modo que tiene Windows de utilizar las bibliotecas de enlace dinámico para proporcionar funciones ejecutables hace que tan sólo exista una copia de cada biblioteca en la memoria, y a pesar de ello, cualquier aplicación que se esté ejecutando puede llamar a las funciones API. Esta técnica obtiene un ahorro de memoria al disminuir el tamaño de las aplicaciones.

I.2-- Funciones gráficas de Windows utilizadas en el desarrollo del sistema.

En este capítulo se presentan algunas de las funciones del API de Windows utilizadas en el desarrollo de la aplicación. Para cada caso se explica el objetivo de la función , así como los parámetros que esta requiere. De las siguientes funciones no todas ellas dan como resultado directo algún efecto visual, por lo tanto solo se pondrán ejemplos de aquellas en las cuales su ejecución sea visual. En el apéndice de este libro podrá encontrar una lista completa de todas las funciones gráficas del API de Windows, junto con una breve descripción de estas

API: HDC BeginPaint(hWnd,lpDibujo)

Función
Esta función prepara la ventana para que se dibuje y utilice el parámetro lpDibujo, que apunta a una estructura de dibujo y rellena la estructura con información sobre el dibujo que se va a realizar.

Parámetros
hWnd indica la ventana que se va a volver a pintar.
lpDibujo (LPPAINSTRUCT) apunta a una estructura PAINTSTRUCT que va a recibir los parámetros de dibujo.

Valor devuelto
devuelve el contexto de dispositivo de la ventana especificada.

Notas
Se debe llamar a BeginPaint en respuesta a un mensaje WM_PAINT.

ejemplo:

```
case WM_PAINT:

        hdc=BeginPaint(hWnd,&ps);
        if(ayuda==TRUE){
        LECTOR_AYUDA(hWnd);
        }
        else {BitBlt(hdc,0,0,limite.right,limite.bottom,hdcUltima_Imagen,0,0,SRCCOPY);}
        EndPaint(hWnd,&ps);
        break;
```

API: HFONT CreateFont(nAltura, nAnchura, nEscape, nOrientación, nPeso, cItálica, cSubrayado,
cTachado, cConjCarac, cPrecisSalida, cPcsisRs, cCalidad, cPasoyFamilia,
lpNombreTipo)

Función
Esta función crea una fuente lógica que se puede seleccionar como fuente para cualquier dispositivo. La fuente lógica creada dependerá de las siguientes opciones.

Parámetros
nAaltura (int), la altura de la fuente lógica se puede definir, de alguna de las siguientes formas:

- nAltura > 0 trasforma la altura en unidades de dispositivo y la hace corresponder con la altura de la celdilla de las fuentes disponibles.
- nAltura = 0 selecciona una razonable tamaño por defecto.
- nAltura < 0 trasforma la altura en unidad de dispositivo y hace corresponde el valor absoluto con la altura del carácter de las fuentes disponibles.

En las correspondencias de fuentes, todas las comparaciones de altura se realizan siguiendo las reglas que se citan a continuación: En primer lugar se busca la fuente más grande que no excede el tamaño de fuente solicitado. Si no está disponible tal fuente, se busca la fuente más pequeña disponible.
nAnchura (int), Usando unidades lógicas, define la anchura media de los caracteres de la fuente. Cuando nAnchura contiene el valor cero (0), la relación de aspecto del dispositivo se hace corresponder con la relación de aspecto de la representación digital de las fuentes disponibles, a fin de encontrar la mejor selección.

nEscape (int), usando incrementos de décimas de grado, define el ángulo de impresión de cada línea de texto de una determinada fuente, relativo a la parte inferior de la página.

nOrientación (int), usando incrementos de décimas de grado, especifica el ángulo de la línea base de cada carácter, relativo a la parte inferior de la página.

nPeso (int) indica el peso deseado de la fuente, dentro del rango de 0 a 1000. El valor cero selecciona el peso 0 por defecto; un valor próximo a 450 define un peso normal; y 750 representa el peso asociado a la negrita.

cItálica (BYTE) indica si la fuente es cursiva o no.
cSubrayado (BYTE) indica si la fuente está subrayada o no.
cTachado (BYTE) indica si los caracteres de la fuente están tachados o no.
cConjCarac (BYTE) indica cuál es el conjunto de caracteres deseados. Hay dos valores predefinidos: ANSI_CHARSET y OEM_CHARSET. Usted puede adquirir otros conjuntos de caracteres desarrollados por creadores de fuentes.
cPrecisSalida (BYTE) especifica la precisión de salida deseada. Con esto se especifica el grado de similitud entre la salida generada y las características de la fuente solicitada. existen 4 opciones :

OUT_CHARACTER_PRECIS
OUT_DEFAULT_PRECIS
OUT_STRING_PRECIS
OUT_STROKE_PRECIS

cPrecisRec (BYTE) indica la precisión de recorte deseada. La precisión de recorte se refiere a la forma en que se van a recortar los caracteres que sobrepasen la región de recorte. Existen tres opciones:

CLIP_CHARACTER_PRECIS
CLIP_DEFAULT_PRECIS
CLIP_STROKE_PRECIS

cCalidad (BYTE) selecciona la calidad de salida deseada. Este parámetro selecciona el grado de precisión
con el que el Interfaz de dispositivo gráfico (GDI) debe intentar realizar la correspondencia entre fuentes físicas reales suministradas para el dispositivo de salida solicitado. Existen tres posibles condiciones de correspondencia:

DEFAULT_QUALITY
DRAFT_QUALITY
PROOF_QUALITY

cPasoyFamilia (BYTE) selecciona el paso y la familia de la fuente. Existen tres opciones de paso:

DEFAULT_PITCH
FIXED_PITCH
VARIABLE_PITCH

Hay seis posibles familias de fuentes:

FF_DECORATIVE
FF_DONTCARE
FF_MODERN
FF_ROMAN
FF_SCRIPT
FF_SWISS

lpNombreTipo (LPSTR) es una cadena finalizada con el carácter nulo (y de un máximo de 30 caracteres) que identifica el nombre del tipo de letra de la fuente. Si desea conocer los tipos de letra disponibles, puede llamar a la función EnumFonts para mostrar la lista de fuentes.

Valor devuelto.

Si la operación tiene éxito, se devuelve un valor que identifica la fuente lógica recién creada. El valor NULL indica que no se ha podido crear la fuente.

Notas.

Es importante entender que la función CreateFont no crea una nueva fuente. Todo lo que hace es hacer corresponder lo mejor que se pueda, en función de los parámetros de correspondencia especificados, sus preferencias con las fuentes físicas reales disponibles.

Ejemplo:

```
Font1= CreateFont(   // para cambiar el texto de tamaño
   10,                          /* altura, en pixeles */
   0,     /* usar la razón de aspecto para determinar la anchura */
   0,          /* ángulo de la línea de texto, en .1 grados */
   0,          /* nulo de cada cara etc., en .1 grados */
   FW_BOLD,                     /* ó FW_LIGHT ó FW_NORMAL */
   FALSE,                    /* TRUE para que sea cursiva */
   FALSE,                  /* TRUE para que este subrayada */
   FALSE,                   /* TRUE para que este tachada */
   ANSI_CHARSET,              /* juego de caracteres ANSI */
   OUT_DEFAULT_PRECIS,          /* m todo de correspondencia */
   CLIP_DEFAULT_PRECIS,            /* m todo de recorte */
   DRAFT_QUALITY,                /* escalado activado */
   VARIABLE_PITCH | FF_SWISS,       /* inclinación y familia */
   "Helv");                  /* nombre de la tipografía */
```

API : HPEN CreatePen(nEstiloLápiz,nAnchura,rgbColor)

Función.

Esta función crea un lápiz lógico en función del estilo, la anchura, y el color seleccionados.

Parámetros.

nEstiloLápiz (int) selecciona el estilo del lápiz, de las seis posibles opciones.

Constante	Estilo lápiz
0	Sólido
1	Guión
2	Punto
3	Guión-punto
4	Guión-punto-punto
5	NULL (ninguno)

nAnchura (int) define el grosor del lápiz en unidades lógicas.
rgbColor (DWORD) y crColor (DWORD) selecciona un color rgb para el lápiz.

Valor devuelto.

Este valor identifica al lápiz lógico recién creado, o es NULL si no tiene éxito la operación.

Notas.

Los lápices creados con grosor físico mayor que un pixel siempre tendrán estilo sólido o bien NULL.

Ejemplo:

```
HPEN Pluma, Pluma1;
Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,40,4,430,57);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
```

API : HBRUSH CreateSolidBrush(rgbColor)

Función.

Esta función crea un pixel lógico, usando el color seleccionado.

Parámetros.

rgbColor (DWORD) Y crColor (DWORD) seleccionan un color RGB para el pincel.

Valor devuelto.

Si tiene éxito, el valor devuelto identifica al pincel recién creado; en caso contrario, se devuelve NULL.

Ejemplo:

```
HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;

Brocha= CreateSolidBrush(RGB(0,0,255));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,440,2,620,82);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
```

API : HWND CreateWindow(lpNomClase,lpNomVentana,dwEstilo, X,Y, nAnchura, nAltura, hWndPadre, hMenu, hInstancia, lpParam)

Función.

Esta función, frecuentemente usada, crea una ventana superpuesta, instantánea o hija. Esta función especifica la clase de la ventana, el título y el estilo, e incluso se puede indicar el tamaño y la posición iniciales de la ventana. Si la ventana que se va a crear pertenece a una ventana o menú padre, también se deben especificar. Esta función también envía a la ventana los necesarios mensajes WM_CREATE, WM_GETMINMAXINFO y WM_NCCREATE. Si se ha seleccionado la opción de estilo WS_VISIBLE, se envían todos los mensajes necesarios para activar y mostrar la ventana.

Parámetros.

lpNomClase (LPSTR) apunta a una cadena de caracteres finalizada con el carácter nulo, y que especifica el nombre de la clase de la ventana.

lpNomVentana (LPSTR) apunta a una cadena de caracteres finalizada con el carácter nulo, y que especifica el nombre de la ventana.

dwEstilo especifica el estilo de la ventana.

X (int) especifica la coordenada x de la posición inicial de la ventana. Para las ventanas superpuestas / instantáneas, ésta es la coordenada x de la esquina superior izquierda de la ventana (en coordenadas de

pantalla). Si el valor es CW_USEDEFAULT, Windows selecciona la coordenada x por defecto Para las ventanas hijas, ésta es la coordenada x de la esquina superior izquierda de la ventana dentro del área de trabajo de su ventana padre.

Y (int) especifica la coordenada y de la posición inicial de la ventana. Para las ventanas superpuestas / / / instantáneas ,ésta es la coordenada y de la esquina superior izquierda de la ventana (en coordenadas de pantalla). Si el valor es CW_USEDEFAULT , Windows selecciona la coordenada y por defecto. Para las ventanas hijas, ésta es la coordenada y de las superior izquierda de la ventana dentro del área de trabajo de su ventana padre.

nAnchura (int) define la anchura de la ventana, usando unidades del dispositivo Para las ventanas superpuestas , ésta es la anchura de la ventana en coordenadas de pantalla, o bien CW_USEDEFAULT, en cuyo caso Windows selecciona la anchura y altura de la ventana

nAltura (int) define la altura de la ventana, utilizando unidades del dispositivo. Para las ventanas superpuestas, ésta es la altura de la ventana, en coordenadas de pantalla, o bien CW_USEDEFAULT, en cuyo caso Windows selecciona la anchura y altura de la ventana

hWndPadre (HWND) especifica la ventana padre de la ventana que se va a crear. Las ventanas superpuestas no deben tener una ventana padre (hWndPadre debe ser NULL). Cuando se esté creando una ventana hija, se debe pasar el manipulador válido de una ventana padre.

hMenu (HMENU), en función del estilo de la ventana, especifica un identificador de una ventana hija o de un menú. Para los menús superpuestos / instantáneos, este parámetro identifica al menú que se va a usar en la ventana. Un valor NULL especifica la utilización del menú de la clase. Para un menú hijo , contiene el identificador de una ventana hija. Este identificador lo determina la aplicación, y va a ser único para todas las ventanas que pertenecen a una misma ventana padre.

hInstancia (HANDLE) especifica la instancia del módulo que se va a identificar con la ventana.

lpParam (LPSTR) apunta al valor pasado a la ventana a través del parámetro lParam del mensaje WM_CREATE.

Valor devuelto.

Si tiene éxito, se devuelve un identificador válido de la nueva ventana; en caso contrario se devuelve un NULL.

Scrollbar.

El control scrollbar muestra un cuadro rectangular alargado que contiene una referencia de posición de página, que se conoce como cuadro de desplazamiento, junto con flechas de sentido de desplazamiento en ambos extremos. El usuario puede seleccionar una posición en la lista desplazando hacia abajo o hacia arriba el cuadro de desplazamiento. Los controles scrollbar tienen idéntica apariencia que las barras de desplazamiento que aparecen en las ventanas normales. El control SIZEBOX esta automáticamente asociado a los controles scrollbar. El control SIZEBOX es un pequeño rectángulo que permite al usuario modificar el tamaño de la ventana. Los estilos del control scrollbar son los siguientes:

SBS_BOTTOMALIGN
SBS_HORZ
SBS_BOTTOMALIGN
SBS_LEFTALIGN
SBS_RIGHTALIGN
SBS_SIZEBOX
SBS_SIZEBOXBOTTOMRIGHTALIGN
SBS_SIZEBOXTOPLEFTALIGN
SBS_SIZEBOXBOTTOMRIGHTALIGN
SBS_TOPALIGN
SBS_VERT

Static.

La clase de control STATIC define un simple campo , cuadro o rectángulo de texto, y que se usa con frecuencia para , identificar, enmarcar o separar otros controle. Estos controles muestran o piden información . Los estilos de control STATIC son :

pantalla). Si el valor es CW_USEDEFAULT, Windows selecciona la coordenada x por defecto Para las ventanas hijas, ésta es la coordenada x de la esquina superior izquierda de la ventana dentro del área de trabajo de su ventana padre.

Y (int) especifica la coordenada y de la posición inicial de la ventana Para las ventanas superpuestas / / / instantáneas ,ésta es la coordenada y de la esquina superior izquierda de la ventana (en coordenadas de pantalla). Si el valor es CW_USEDEFAULT , Windows selecciona la coordenada y por defecto. Para las ventanas hijas, ésta es la coordenada y de las superior izquierda de la ventana dentro del área de trabajo de su ventana padre.

nAnchura (int) define la anchura de la ventana, usando unidades del dispositivo Para las ventanas superpuestas , ésta es la anchura de la ventana en coordenadas de pantalla, o bien CW_USEDEFAULT, en cuyo caso Windows selecciona la anchura y altura de la ventana

nAltura (int) define la altura de la ventana, utilizando unidades del dispositivo. Para las ventanas superpuestas, ésta es la altura de la ventana, en coordenadas de pantalla, o bien CW_USEDEFAULT, en cuyo caso Windows selecciona la anchura y altura de la ventana.

hWndPadre (HWND) especifica la ventana padre de la ventana que se va a crear. Las ventanas superpuestas no deben tener una ventana padre (hWndPadre debe ser NULL). Cuando se esté creando una ventana hija, se debe pasar el manipulador válido de una ventana padre.

hMenu (HMENU), en función del estilo de la ventana, especifica un identificador de una ventana hija o de un menú. Para los menús superpuestos / instantáneos, este parámetro identifica al menú que se va a usar en la ventana. Un valor NULL especifica la utilización del menú de la clase. Para un menú hijo , contiene el identificador de una ventana hija. Este identificador lo determina la aplicación, y va a ser único para todas las ventanas que pertenecen a una misma ventana padre.

hInstancia (HANDLE) especifica la instancia del módulo que se va a identificar con la ventana.

lpParam (LPSTR) apunta al valor pasado a la ventana a través del parámetro lParam del mensaje WM_CREATE.

Valor devuelto.

Si tiene éxito, se devuelve un identificador válido de la nueva ventana; en caso contrario se devuelve un NULL.

Scrollbar.

El control scrollbar muestra un cuadro rectangular alargado que contiene una referencia de posición de página, que se conoce como cuadro de desplazamiento, junto con flechas de sentido de desplazamiento en ambos extremos. El usuario puede seleccionar una posición en la lista desplazando hacia abajo o hacia arriba el cuadro de desplazamiento. Los controles scrollbar tienen idéntica apariencia que las barras de desplazamiento que aparecen en las ventanas normales. El control SIZEBOX esta automáticamente asociado a los controles scrollbar. El control SIZEBOX es un pequeño rectángulo que permite al usuario modificar el tamaño de la ventana. Los estilos del control scrollbar son los siguientes:

SBS_BOTTOMALIGN
SBS_HORZ
SBS_BOTTOMALIGN
SBS_LEFTALIGN
SBS_RIGHTALIGN
SBS_SIZEBOX
SBS_SIZEBOXBOTTOMRIGHTALIGN
SBS_SIZEBOXTOPLEFTALIGN
SBS_SIZEBOXBOTTOMRIGHTALIGN
SBS_TOPALIGN
SBS_VERT

Static.

La clase de control STATIC define un simple campo , cuadro o rectángulo de texto, y que se usa con frecuencia para , identificar, enmarcar o separar otros controle. Estos controles muestran o piden información . Los estilos de control STATIC son :

SS_BLACKFRAME
SS_BLACKRECT
SS_CENTER
SS_GRAYFRAME
SS_GRAYRECT
SS_ICON
SS_LEFT.
SS_LEFTNOWORDWRAP
SS_NOPREFIX
SS_RIGHT
SS_SIMPLE
SS_WHITEFRAME
SS_WHITERECT
SS_USERITEM

Estilos de ventanas.

A continuación se muestra una lista de estilos de ventana:

WS_VSCROLL
WS_BORDER
WS_CAPTION
WS_DLGFRAME
WS_CHILD
WS_POPUP
WS_CHILDWINDOW
WS_CLIPCHILDREN
WS_CLIPSIBLINGS
WS_CHILD
WS_DISABLED.
WS_DLGFRAME
WS_GROUP
WS_HSCROLL
WS_MAXIMIZE
WS_MINIMIZE
WS_OVERLAPPED
WS_OVERLAPPEDWINDOW
WS_THICKFRAME
WS_MINIMIZEBOX
WS_MAXIMIZEBOX
WS_POPUP
WS_POPUPWINDOW
WS_SYSMENU
WS_CAPTION
WS_TABSTOP
WS_THICKFRAME
WS_VISIBLE
WS_VSCROLL

Ejemplo:

```
hWnd = CreateWindow(
  "TESIS",
  cadena_titulos,
     WS_OVERLAPPED  |     WS_SYSMENU  |  WS_CLIPCHILDREN  |WS_THICKFRAME  |
WS_MINIMIZEBOX
  , 4,4,
```

```
130,150,
NULL, NULL,
hInstance, (LPSTR)NULL);
```

API : BOOL Ellipse(hDC,X1,Y1,X2,Y2)

Función.

Esta función dibuja una elipse.

Parámetros.

hDC (HDC) identifica el contexto del dispositivo.
X1 (int) define la coordenada x del la esquina superior izquierda del rectángulo del delimitador.
Y1 (int) define la coordenada y del la esquina superior izquierda del rectángulo del delimitador.
X2 (int) define la coordenada x del la esquina inferior derecha del rectángulo del delimitador.
Y2 (int) define la coordenada y del la esquina inferior derecha del rectángulo del delimitador.

Valor devuelto.

Verdadero (TRUE) si tiene éxito, en caso contrario falso (FALSE).

Notas.

La anchura y altura del rectángulo no deben superar las 32767 unidades.

Ejemplo:

```
HBRUSH BROCHA, BROCHA1;
HPEN PLUMA,PLUMA1;
HDC hdc;
hdc=GetDC(hWnd);
   BROCHA= CreateSolidBrush(RGB(127,0,0));
   BROCHA1 = SelectObject(hdc,BROCHA);
   PLUMA= CreatePen(PS_SOLID,5,RGB(0,0,0));
   PLUMA1= SelectObject(hdc,PLUMA);

Ellipse(hdc,70,50,200,210);

SelectObject(hdc,BROCHA1);
DeleteObject(BROCHA1);
SelectObject(hdc,PLUMA1);
DeleteObject(PLUMA1);
ReleaseDC(hWnd,hdc);
```

Ejecución

Función Ellipse

API : BOOL EnableMenuItem(hMenu,wIDActivaElem,wActivar)

Función.

Esta función activa, desactiva o pone en gris un elemento de un menú.

Parámetros.

hMenu (HMENU) identifica al menú.
wIDActivaElem (WORD) o nIDActivaElem (UINT) especifica el elemento del menú / menú instantáneo a verificar.
wActivar (WORD) o nActivar (UINT) especifica la acción a realizar. Estas opciones se pueden combinar con un OR lógico:

Opción.	Descripción.
MF_BYCOMMAND	Indica que el parámetro wIdActivaElem
MF_BYPOSITION	Contiene el identificador (ID) del elemento del menú.
MF_DISABLED	Indica que el parámetro wIDActivaElem contiene la posición del elemento del menú.
MF_ENABLED	Activa el elemento del menú.
MF_GRAYED	Pone en gris el elemento del menú.

Valor devuelto.

El valor devuelto representa el estado previo del elemento de menú.

Notas.

Se puede usar el mensaje WM_SYSCOMMAND para activar o desactivar la entrada en una barra de menús.

Ejemplo:

```
case WM_INITMENUPOPUP:
      if(ayuda==TRUE){
        EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_GRAYED);
        EnableMenuItem(wParam,TXT_DEMO_1,MF_GRAYED);
        EnableMenuItem(wParam,TXT_DEMO_2,MF_GRAYED);
        EnableMenuItem(wParam,TXT_DEMO_3,MF_GRAYED);
        EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_GRAYED);}
      else {
```

```
        EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_ENABLED);
        EnableMenuItem(wParam,TXT_DEMO_1,MF_ENABLED);
        EnableMenuItem(wParam,TXT_DEMO_2,MF_ENABLED);
        EnableMenuItem(wParam,TXT_DEMO_3,MF_ENABLED);
        EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_ENABLED);}
        break;
```

API : void EndPaint(hWnd, lpPaint)

Función.

Esta función señala que se ha completado el dibujo de una determinada ventana.

Parámetros.

hWnd (HWND) especifica la ventana que ha sido redibujada.
lpPaint (LPPAINSTRUCT) apunta a una estructura PAINSTEUCT que contiene la información obtenida mediante la llamada a la función BeginPaint.

Valor devuelto.

Ninguno.

Notas.

Se debe realizar una llamada a EndPaint por cada llamada a la función BeginPaint. Mientras que una llamada a la función BeginPaint puede ocultar el punto de inserción, si se hace una llamada a EndPaint, se mostrará dicho punto de inserción.

API : int FillRect (hDC, lpRect, hPincel)

Función.

Esta función rellena un rectángulo, usando el pincel especificado.

Parámetros

hDC (HDC) especifica el contexto de dispositivo.
lpRect (LPRECT) apunta a una estructura RECT que contiene las coordenadas del rectángulo que se va a
rellenar usando el pincel especificado.
hPincel (HBRUSH) selecciona el pincel usado para rellenar el rectángulo.

Valor devuelto.

La función devuelve un valor entero que no se usa, y por tanto, se ignora.

Notas.

El rectángulo no se puede rellenar a menos que se haya creado previamente un pincel usando las funciones CreateSolidBrush, CreatePatternBrush o CreateHatchBrush. El rectángulo se rellena incluyendo el borde superior y el borde izquierdo. Los bordes inferior y derecho no se pintan.

Ejemplo:

```
HDC hDC;
HBRUSH Brocha1, Brocha;
RECT rcClientArea;
```

```
HPEN Pluma1, Pluma;
Brocha= CreateSolidBrush(RGB(0,0,128));
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
hDC= GetDC(hWnd);
Brocha1= SelectObject(hDC,Brocha);
Pluma1= SelectObject(hDC,Pluma);
GetClientRect(hWnd,&rcClientArea);
FillRect(hDC,&rcClientArea,Brocha);
SelectObject(hDC,Pluma1);
SelectObject(hDC,Brocha1);
ReleaseDC(hWnd,hDC);
DeleteObject(Brocha);
DeleteObject(Pluma);
```

Ejecución:

API : void GetClientRect (hWnd, lpRect)

Función.

Esta Función copia las coordenadas de el área de trabajo de una ventana en la estructura de datos lpRect.

Parámetros.

hWnd (HWND) especifica la ventana asociada al área de trabajo en cuestión.
lpRect (LPRECT) apunta a una estructura RECT.

Valor devuelto.

Ninguno.

Notas.

Las coordenadas devueltas representan las esquinas superior izquierda e inferior derecha del área de trabajo. Las coordenadas del área de trabajo son relativas a la esquina superior izquierda (0,0) de la ventana.

Ejemplo:

```
RECT rcClientArea;
GetClientRect(hWnd,&rcClientArea);
```

API : CDC* GetDC ()

Función.

Esta función devuelve un manipulador de un contexto de presentación para el área de trabajo de la ventana especificada.

Parámetros.

hWnd (HWND) especifica la ventana cuyo contexto de presentación se va a devolver.

Valor devuelto.

Si es un valor distinto de cero, indica el contexto de presentación. Una llamada sin éxito devuelve un NULL.

Notas.

EL valor devuelto se puede usar en posteriores llamadas a funciones GDI que dibujen en el área de trabajo.

Ejemplo:

```
HDC hDC;
hDC= GetDC(hWnd);
....
....
.....
.....
ReleaseDC(hWnd,hDC);
```

API : BOOL GetMessage(lpMsg, hWnd, wMsgFiltroMin, wMsgFiltroMax)

Función.

Esta función toma un mensaje de la cola de mensajes de la aplicación y lo coloca en la estructura de datos MSG. Si no hay mensajes disponibles, se cede el control.

Parámetros.

lpMsg (LPMSG) apunta a una estructura MSG que contiene información sobre el mensaje, suministrada por la cola de aplicaciones de Windows.
hWnd (HWND) especifica la ventana cuyos mensajes se van a obtener.
Si este parámetro es NULL, la función devolverá cualquier mensaje destinado a la ventana y que pertenezca a la aplicación llamante.
wMsgFiltroMin (unsigned) es un valor entero que especifica el mensaje más bajo a examinar.
wMsgFiltroMax (unsigned) es un valor entero que especifica el mensaje más alto a examinar.

Valor devuelto.

VERDADERO (TRUE) indica que se ha obtenido algún mensaje distinto de WM_QUIT. FALSO (FALSE) indica que el mensaje obtenido fue WM_QUIT.

Notas.

Si se usan las constantes WM_KEYFIRST y WM_KEYLAST sólo se obtendrán los mensajes relacionados con la entrada por teclado. Las constantes WM_MOUSEFIRST y WM_MOUSELAST pueden servir de filtro y obtener sólo los mensajes relacionados con el ratón.

Función.

Esta función devuelve un manipulador de un contexto de presentación para el área de trabajo de la ventana especificada.

Parámetros.

hWnd (HWND) especifica la ventana cuyo contexto de presentación se va a devolver.

Valor devuelto.

Si es un valor distinto de cero, indica el contexto de presentación. Una llamada sin éxito devuelve un NULL.

Notas.

EL valor devuelto se puede usar en posteriores llamadas a funciones GDI que dibujen en el área de trabajo.

Ejemplo:

```
HDC hDC;
hDC= GetDC(hWnd);
....
....
.....
.....
ReleaseDC(hWnd,hDC);
```

API : BOOL GetMessage(lpMsg, hWnd, wMsgFiltroMin, wMsgFiltroMax)

Función.

Esta función toma un mensaje de la cola de mensajes de la aplicación y lo coloca en la estructura de datos MSG. Si no hay mensajes disponibles, se cede el control.

Parámetros.

lpMsg (LPMSG) apunta a una estructura MSG que contiene información sobre el mensaje, suministrada por la cola de aplicaciones de Windows.
hWnd (HWND) especifica la ventana cuyos mensajes se van a obtener.
Si este parámetro es NULL, la función devolverá cualquier mensaje destinado a la ventana y que pertenezca a la aplicación llamante.
wMsgFiltroMin (unsigned) es un valor entero que especifica el mensaje más bajo a examinar.
wMsgFiltroMax (unsigned) es un valor entero que especifica el mensaje más alto a examinar.

Valor devuelto.

VERDADERO (TRUE) indica que se ha obtenido algún mensaje distinto de WM_QUIT. FALSO (FALSE) indica que el mensaje obtenido fue WM_QUIT.

Notas.

Si se usan las constantes WM_KEYFIRST y WM_KEYLAST sólo se obtendrán los mensajes relacionados con la entrada por teclado. Las constantes WM_MOUSEFIRST y WM_MOUSELAST pueden servir de filtro y obtener sólo los mensajes relacionados con el ratón.

Ejemplo:

```
while (GetMessage(&msg,0,0,0))
  {
  if(TranslateAccelerator(Gestor_vent, hAccel, &msg))
    continue;
  TranslateMessage(&msg);
  DispatchMessage(&msg);
  }
```

API : int GetScrollPos (hWnd, nBarra)

Función.

Esta función devuelve la posición actual del cuadro de desplazamiento de la barra de desplazamiento especificada.

Parámetros.

hWnd (HWND) identifica la ventana que contiene la barra de desplazamiento estándar.
nBarra (int) identifica la barra de desplazamiento a examinar, usando uno de los siguientes valores:

Valor	Descripción.
SB_CTL hWnd	Devuelve la posición de un control de barra de desplazamiento, suponiendo que apunta a un manipulador de ventana de un control de tipo barra de desplazamiento.
SB_HORZ	Devuelve la posición de la barra de desplazamiento horizontal de una ventana.
SB_VERT	Devuelve la posición de la barra de desplazamiento vertical de una ventana.

Notas.

El valor devuelto es un valor relativo, que depende del rango de desplazamiento actual; esto es, en un rango de 0 a 50 tendría una posición intermedia de valor 25.

API ; void GetScrollRange(hWnd, nBarra, lpPosMin, lpPosMax)

Función.

Esta función devuelve las posiciones mínima y máxima actuales de la barra de desplazamiento definida.

Parámetros.

hWnd (HWND) especifica la ventana que posee una barra de desplazamiento estándar o un control de barra de desplazamiento.

Valor	Descripción.
SB_CTL hWnd	Devuelve la posición de un control de barra de desplazamiento, suponiendo que apunta a un manipulador de ventana de un control de tipo barra de desplazamiento.

SB_HORZ Devuelve la posición de la barra de desplazamiento horizontal de una ventana.

SB_VERT Devuelve la posición de la barra de desplazamiento vertical de una ventana.

lpPosMin (LPINT) apunta a una variable entera que va a recibir el valor de la posición mínima de la barra de desplazamiento.
lpPosMax (LPINT) apunta a una variable entera que va a recibir el valor de la posición máxima de la barra de desplazamiento.

Valor devuelto.
Ninguno.

Notas.

El rango de valores por defecto de una barra de desplazamiento estándar varía entre 0 y 100.

API : int GetSystemMetrics (nIndice)

Función

Esta función devuelve la métrica del sistema. Las medidas representan la anchura y la altura de los diversos elementos de la pantalla.

Parámetros

nIndice (int) es la medida del sistema que se va a obtener.

Posibles índices de medida del sistema

Indice	Descripción
SM_CXBORDER	Grosor de un marco de ventana no redimencionable
SM_CYBORDER	Altura de un marco de ventana no redimencionable
SM_CYCAPTION	Altura de la barra de título
SM_CXCURSOR	Anchura del cursor
SM_CYCURSOR	Altura del cursor
SM_CXDLGFRAME	Anchura de una ventana de estilo WS_DLGFRAME
SM_CYDLGFRAME	Altura de una ventana de estilo WS_DLGFRAME
SM_CXFRAME	Grosor del marco redimensionable de ventana
SM_CYFRAME	Altura del marco redimensionable de ventana
SM_CXFULLSCREEN	Anchura de la ventana de área de trabajo a pantalla completa
SM_CYFULLSCREEN	Altura de la ventana de área de trabajo a pantalla completa
SM_CXICON	Anchura del icono.
SM_CYICON	Altura del icono.
SM_CYMENU	Altura de una barra de menú de una sola línea
SM_CXMIN	Anchura Mínima de la ventana
SM_CYMIN	Altura Mínima de la ventana
SM_CXMINTRACK	Anchura Mínima de seguimiento de la ventana
SM_CYMINTRACK	Altura Mínima de seguimiento de la ventana
SM_CXSCREEN	Anchura de la pantalla
SM_CYSCREEN	Altura de la pantalla
SM_CXHSCROLL	Anchura de la matriz de puntos de flecha en una barra de desplazamiento horiz
SM_CYHSCROLL	Altura de la matriz de puntos de flecha en una barra de desplazamiento horiz

SM_CXVSCROLL	Anchura de la matriz de puntos de flecha en una barra de desplazamiento vert.
SM_CYVSCROLL	Altura de la matriz de puntos de flecha en una barra de desplazamiento vert
SM_CXSIZE	Anchura de la matriz de puntos de la barra de título
SM_CYSIZE	Altura de la matriz de puntos de la barra de título
SM_CXHTHUMB	Anchura del cuadro de desplazamiento de una barra de desplazamiento horiz.
SM_CYVTHUMB	Altura del cuadro de desplazamiento de una barra de desplazamiento horiz.
SM_DEBUG	Distinto de cero en una versión de depuración de Windows.
SM_MOUSEPRESENT	Distinto de cero si se ha instalado hardware de ratón
SM_SWAPBUTTON	distinto de cero si se han intercambiado los botones izquierdo y derecho del ratón

Valor devuelto

Indica la métrica del sistema especificada.

Notas

La función también puede indicar si la versión de Windows que se está usando está preparada para depuración, si hay un ratón, si lo hay, si se han intercambiado sus botones derecho e izquierdo.

API : BOOL Lineto (hDC, X, Y)

Función.

Esta función dibuja una línea, usando el lápiz actual, desde la posición actual hasta el punto especificado por X e Y, sin incluirlo. Tras esta operación, la posición actual pasa a ser (X, Y)

Parámetros

hDC (HDC) especifica el contexto de dispositivo.
X (int) identifica la coordenada lógica x del último punto de la línea.
Y (int) identifica la coordenada lógica y del último punto de la línea.
punto es una estructura POINT o un objeto CPoint.

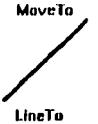
Valor devuelto.

VERDADERO (TRUE) si tiene éxito, o FALSO (FALSE) en caso contrario.

Ejemplo:

```
HPEN Pluma1, Pluma;
HDC hdc;
hdc=GetDC(hwnd,hdc);
 MoveTo(hdc,500,100);
 LineTo(hdc,400,300);
 ReleaseDC(hwnd,hdc);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
```

Ejecución:



API : HCURSOR LoadCursor (hInstancia, lpNomCursor)

Función.

Esta función carga el cursor seleccionado del archivo ejecutable asociado al módulo al que apunta hInstancia.

Parámetros.

hInstancia (HANDLE) especifica la instancia del módulo cuyo archivo ejecutable contiene el cursor a cargar. Cuando el parámetro es NULL, se puede usar esta función para cargar un icono predefinido de Windows, siendo lpNomCursor uno de los siguientes:

Opción.	Descripción.
IDC_ARROW	Cursor flecha estándar de Windows.
IDC_CROSS	Cursor estándar de Windows en forma de cruz.
IDC_IBEAM	Cursor estándar de Windows de línea vertical.
IDC_SIZENESW	Cursor con dos flechas que apuntan a NE y SW.
IDC_SIZENS	Cursor con dos flechas que apuntan al norte y al sur.
IDC_SIZEWE	Cursor con dos flechas que apuntan al este (E) y al oeste (W).
IDC_SIZENWSE	Cursor con dos flechas que apuntan al NW (Noroeste) y al SE (Sureste).
IDC_UPARROW	Cursor flecha vertical estándar de Windows.
IDC_WAIT	Cursor de reloj de arena estándar de Windows.

lpNomCursor (LPSTR) es un puntero a una cadena de caracteres terminada con NULL en la que se indica el nombre del cursor.
nIDCursor (UINT) es un valor identificador asignado al cursor.

Valor devuelto.

Indica el cursor seleccionado , o NULL si el cursor especificado no existe.

Notas.

Esta función se puede usar para cargar un cursor creado mediante una llamada a la función MakeIntResource, usando la palabra de menor peso de lpNomCursor.

API : int MessageBox (hWndPadre, lpTexto,lpTitulo,wTipo)

Función.

Esta función crea y visualiza una ventana que contiene mensajes, títulos, iconos y botones de pulsación suministrados por la aplicación.

Parámetros.

hWndPadre (HWND) especifica la ventana que contiene el cuadro de mensaje.
lpTexto (LPSTR) apunta a la cadena de mensajes terminada con NULL que se va a mostrar.
lpTitulo (LPSTR) apunta a una cadena terminada con NULL que se va a usar como título del cuadro de diálogo. Cuando es NULL, el título que se muestra es "error!".
wTipo (WORD) o nTipo (UINT) identifica el contenido del cuadro de diálogo y puede ser alguno de los valores (o una combinación con OR de los mismos) mostrados en la sección "Notas".

Valor devuelto.

Si se devuelve el valor cero, indica que no hay suficiente memoria para crear el cuadro de mensaje. Si se ha podido crear el cuadro, se devuelve uno de los siguientes elementos de menú devueltos por el cuadro de diálogo:

Opción.	Descripción.
IDABORT	Se pulsó el botón Abortar
IDCANCEL	Se pulsó el botón Salir o Cancelar.
IDIGNORE	Se pulsó el botón Ignorar.
IDNO	Se pulsó el botón No.
IDOK	Se pulsó el botón Ok.
IDRETRY	Se pulsó el botón Reintentar.
IDYES	Se pulsó el botón Sí.

Notas.

A continuación se muestra una lista, por orden alfabético, de los posibles contenidos de un cuadro de diálogo.

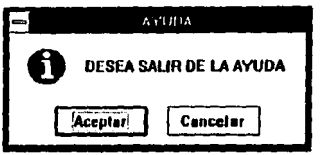
Identificador.	Descripción. (indica que botones y figura contiene el cuadro de mensaje).
MB_ABORTRETRYIGNORE	Tres botones: Abortar, Reintentar e Ignorar.
MB_APPLMODAL	El usuario debe responder al cuadro de mensaje.
MB_DEFBUTTON1	El primer botón es el botón por defecto.
MB_DEFBUTTON2	El segundo botón es el botón por defecto.
MB_DEFBUTTON3	El tercer botón es el botón por defecto.
MB_ICONASTERISK	Se muestra el icono de un asterisco en el cuadro de mensaje.
MB_ICONINFORMATION.	Muestra la figura de una "i" encerrada en un círculo.
MB_ICONEXCLAMATION.	Se muestra el icono de una exclamación en el cuadro de mensaje.
MB_ICONHAND	Muestra el icono de una mano en el cuadro de mensaje.
MB_ICONSTOP.	Aparece un signo de stop en el cuadro de mensaje.
MB_ICONQUESTION	Muestra un signo de interrogación en el cuadro de mensaje.
MB_OK.	Un botón, que dice OK.
MB_OKCANCEL	Dos botones: uno dice OK y el otro Cancelar.
MB_RETRYCANCEL	Dos botones: Reintentar y Cancelar.
MB_SYSTEMMODAL	Suspende todas las aplicaciones debido a la seriedad del suceso.
MB_TASKMODAL	Igual que MB_APPLMODAL pero las ventanas no gestionan igual.
MB_YESNO	Dos botones: Uno de Si y el otro de No.
MB_YESNOCANCEL.	Tres botones: Si , No y Cancelar.

Ejemplo:

```
    mensaje=MessageBox(GetFocus(),"DESEA SALIR DE LA AYUDA","AYUDA",MB_OKCANCEL
| MB_ICONINFORMATION);
    if(mensaje == IDOK){
        .....
        .....
```

}

Ejecución:



API : DWORD MoveTo (hDC, X, Y)

Función.

Esta función mueve la posición actual a las coordenadas x u y especificadas.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
X (int) identifica la coordenada x lógica de la nueva posición.
Y (int) identifica la coordenada y lógica de la nueva posición.

Valor devuelto.

Contiene las coordenadas de la posición anterior. La palabra de mayor peso contiene la coordenada y, y la de menor peso la coordenada x.

Notas.

Una llamada a MoveTo afecta a muchas funciones que usan la posición actual.

Ejemplo:

```
HPEN Pluma1, Pluma;
HDC hdc;
hdc=GetDC(hwnd,hdc);
 MoveTo(hdc,300,120);
 LineTo(hdc,500,330);
 ReleaseDC(hwnd,hdc);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
```

Ejecución:

API : BOOL Polygon (hDC,lpPuntos, nCuenta)

Función.

Esta función dibuja un polígono que está formado por dos o más puntos unidos por líneas. Esta función une al final el primer punto con el último punto, por lo tanto la figura queda cerrada.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
lpPuntos (LPPOINT) apunta a un array de estructura POINT que contiene los vértices del polígono.
nCuenta (int) indica el número de elementos del array.

Valor devuelto.

VERDADERO indica que la operación ha tenido éxito; en caso contrario,el resultado es FALSO (FALSE).

Notas

El polígono no usa la posición actual, ni la actualiza después de dibujar el polígono. Este se dibuja utilizando el modo actual de relleno de polígonos. En modo ALTERNATE, se usa el lápiz actual para dibujar las líneas desde el primer punto hasta el siguiente, rellenando el interior con el pincel actual. En modo WINDING, el lápiz actual se utiliza para dibujar un borde que se calcula usando todos los puntos . El interior también se rellena usando el pincel actual.

Ejemplo:

```
HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;
POINT puntos1[7];

puntos1[0].x = 580;  puntos1[0].y = 225;
puntos1[1].x = 605;  puntos1[1].y = 225;
puntos1[2].x = 605;  puntos1[2].y = 215;
puntos1[3].x = 635;  puntos1[3].y = 240;
puntos1[4].x = 605;  puntos1[4].y = 265;
puntos1[5].x = 605;  puntos1[5].y = 255;
puntos1[6].x = 580;  puntos1[6].y = 255;
Brocha= CreateSolidBrush(RGB(0,0,128));
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Brocha1= SelectObject(hDC,Brocha);
Pluma1= SelectObject(hDC,Pluma);
Polygon(hdc,puntos1,7);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1),
```

Ejecución:

Función Polygon

API : BOOL PolyLine(hDC, lpPuntos, nCuenta)

Función.

Esta función dibuja un conjunto de segmentos conectados. Es semejante a la función Polygon, solo que esta no une al final los puntos, es decir, la figura queda abierta.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
lpPuntos (LPPOINT) apunta a un array de estructuras POINT que contiene los vértices del polígono.
nCuenta (int) indica el número de elementos del array.

Valor devuelto.

VERDADERO indica que la operación ha tenido éxito; en caso contrario, el resultado es FALSO (FALSE).

Notas.
PolyLine no usa la posición actual, ni la actualiza después de dibujar la polilínea.

Ejemplo:

```
POINT puntos1[7];

puntos1[0].x = 580;  puntos1[0].y = 225;
puntos1[1].x = 605;  puntos1[1].y = 225;
puntos1[2].x = 605;  puntos1[2].y = 215;
puntos1[3].x = 635;  puntos1[3].y = 240;
puntos1[4].x = 605;  puntos1[4].y = 265;
puntos1[5].x = 605;  puntos1[5].y = 255;
puntos1[6].x = 580;  puntos1[6].y = 255;
PolyLine(hdc,puntos1,7);
```

API : BOOL Rectangle (hDC, X1,Y1,X2,Y2)

Función.

Esta función dibuja un rectángulo y rellena el interior usando el lápiz seleccionado.

Parámetros.

X1 (int) es la coordenada x de la esquina superior izquierda.
Y1 (int) es la coordenada y de la esquina superior izquierda.
X2 (int) es la coordenada x de la esquina inferior derecha.

Y2 (int) es la coordenada y de la esquina inferior derecha.

Valor devuelto.

VERDADERO (TRUE) si tiene éxito; FALSO (FALSE) en caso contrario.

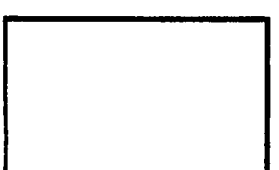
Notas.

La anchura y la altura del rectángulo, definidos por el valor absoluto de X2 - X1 e Y2 - Y1, no debe superar el valor 32767.

Ejemplo:

```
HBRUSH BROCHA,BROCHA1;
HPEN PLUMA,PLUMA1;
HDC hdc;
hdc=GetDC(hWnd);

 BROCHA= CreateSolidBrush(RGB(255,255,0));
 BROCHA1=SelectObject(hdc,BROCHA);
 PLUMA= CreatePen(PS_SOLID,2,RGB(0,0,128));
 PLUMA1=SelectObject(hdc,PLUMA);

Rectangle(hdc,70,50,350,200);

SelectObject(hdc,BROCHA1);
DeleteObject(BROCHA1);
SelectObject(hdc,PLUMA1);
DeleteObject(PLUMA1);
ReleaseDC(hWnd,hdc);
```

Ejecución:



API : int ReleaseDC (hWnd,hDC)

Función.

Libera el contexto de dispositivo, poniéndolo a disposición de otras aplicaciones.

Parámetros.

hWnd (HWND) es un manipulador de la ventana cuyo contexto de dispositivo se va a liberar.
hDC (HDC) especifica el contexto de dispositivo.
pDC (CDC*) apunta al contexto de dispositivo.

Valor devuelto.

Un valor distinto de cero si se ha podido liberar el contexto con éxito en caso contrario se devuelve NULL.

Notas.

Por cada llamada a GetWindowDC o GetDC para obtener un contexto de dispositivo común ,se debe realizar una llamada a ReleaseDC.

API : HANDLE SelectObject (hDC, hObjeto)

Función.

Esta función selecciona un objeto lógico para el contexto de dispositivo especificado.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
hObjeto (HANDLE) o pObjeto (CGdiObject*) especifica el objeto lógico a seleccionar y puede ser alguna de las siguientes funciones

Objeto	Nombre Función.
Matriz de puntos	Create Bitmap
	CreateBitmapIndirect
	CreateCompatibleBitmap
Pincel	CreateBrushIndirect
	CreateHatchBrush
	CreatePatternBrush
	CreateSolidBrush
Fuente	CreateFont
	CreateFontIndirect
Lápiz	CreatePen
	CreatePenIndirect
Objeto	Nombre función
Región	CombineRgn
	CreateEllipticRgn
	CreateEllipticRgnIndirect
	CreatePolyRgn
	CreateRectRgn
	CreateRectRgnIndirect

Valor devuelto

Identifica al objeto sustituido por hObjeto (del mismo tipo). En caso de producirse un error se devuelve NULL.

Notas.

Los objetos seleccionados se toman como objetos por defecto en muchas funciones GDI que escriben textos, dibujan líneas, rellenan interiores y generan la salida para los dispositivos seleccionados. Los contextos de dispositivo pueden tener hasta cinco objetos seleccionados, si bien sólo se puede usar uno cada vez. Cada llamada a SelecObject hace que el GDI asigne espacio para el objeto en el segmento de datos. Se debe llamar a DeleteObject siempre que un objeto seleccionado (fuente, lápiz o pincel) no se necesita más, a fin de ahorrar memoria. Además. las matrices de puntos sólo se pueden usar en un contexto de dispositivo cada vez.

API : DWORD SetBkColor (hDC, rgbColor)

Función.

Esta función selecciona el color especificado como color de fondo actual. La función eligirá el color lógico más próximo a uno de los colores del dispositivo, si no se puede establecer una correspondencia directa.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
rgbColor o crColor (DWORD) selecciona un color RGB que se va a usar como color de fondo.

Valor devuelto.

Contiene el color rgb DE FONDO ANTERIOR. si SE DEVUELVE EL VALOR 80000000 (hexadecimal) se ha producido algún error.

Notas.

Cuando el modo de fondo es OPAQUE, se usa el color de fondo para rellenar los espacios entre líneas con estilo, tramas de pinceles y celdillas de caracteres. El interfaz de dispositivo gráficos (GDI) usa también el color de fondo para convertir las matrices de puntos de color a monocromas y a la inversa.

API : int SetBkMode (hDC, nBkMode)

Función.

Esta función establece el modo de fondo.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
nBkMode (int) selecciona el modo de fondo.

Modo.	Descripción.
OPAQUE	Cuando el modo de fondo es OPAQUE, el color de fondo se usa para rellenar los huecos existentes entre líneas con estilos, tramas de pinceles y celdas de caracteres.
TRANSPARENT	Deja inalterado el fondo.

Valor devuelto.

Devuelve el modo de fondo anterior OPAQUE, o bien TRANSPARENT.

Notas.

La función indica al GDI si debe eliminar o no los colores de fondo de la superficie del dispositivo antes de dibujar texto, pinceles entramados o cualquier estilo de lápiz no sólido.

API : HCURSOR SetCursor (hCursor)

Función.

Esta función establece la forma del cursor del sistema.

Parámetros.

hCursor (HCURSOR) especifica un recurso de tipo cursor previamente cargado (LoadCursor).

Valor devuelto.

Representa al recurso de tipo cursor que define la forma anterior del cursor. Un valor cero significa que no existía una forma previa.

Notas.

La forma del cursor sólo se debería establecer cuando está situado en el área de trabajo o cuando está capturando toda la entrada proveniente del ratón.

Ejemplo:

```
HCURSOR hCursor;
hCursor= LoadCursor(NULL,IDC_ARROW);
```

Ejecución:

Cursor

API : int SetMapMode (hDC,nModoProy)

Función.

Esta función establece el modo de proyección del contexto de dispositivo seleccionado.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
nModoProy (int) selecciona el nuevo modo de proyección, de entre los que se muestran a continuación:

Modo.	Descripción.
MM_ANISOTROPIC.	Proyecta unidades lógicas en unidades arbitrarias con ejes arbitrariamente escalados.
MM_HIENGLISH	Hace corresponder cada unidad lógica con 0.001 pulgadas.
MM_HIMETRIC	Hace corresponder cada unidad lógica con 0.01 milímetros.
MM_ISOTROPIC	Proyecta unidades lógicas en unidades arbitrarias, usando ejes con idéntica escala.
MM_LOMETRIC	Hace corresponder cada unidad lógica con 0.1 milímetro.
MM_ LOENGLISH	Hace corresponder cada unidad lógica con 0.1 pulgadas.
MM_TEXT	Hace corresponder una unidad lógica con un pixel del dispositivo.
MM_TWIPS	Hace corresponder cada unidad lógica con 1/20 puntos de impresora o, aproximadamente, 1/1440 pulgadas.

Valor devuelto.

Representa el modo de proyección anterior.

Notas.

MM_HIENGLISH, MM_HIMETRIC, MM_LOMETRIC, MM_LOENGLISH y MM_TWIPS se usan principalmente en aplicaciones que hacen dibujos utilizando unidades con sentido físico, tales como milímetros o pulgadas. El modo MM_TEXT permite el uso de píxeles específicos del dispositivo, cuyo tamaño varía de un dispositivo a otro. MM_ISOTROPIC permite una relación de aspecto 1:1, que resulta muy útil para conservar la forma exacta de una imagen. El modo MM_ANISOTROPIC posibilita el ajuste de las coordenadas x u y por separado.

Ejemplo:

```
HDC hdc;
GetClientRect(hwnd, &limite),
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 640,442);
SetViewportExt(hdc, limite.right, limite.bottom);
```

API : int SetScrollPos (hWnd, nBarra, nPos, bRedibujar)

Función.

Esta función establece la posición actual del cuadro de desplazamiento de una barra de desplazamiento.

Parámetros.

hWnd (HWND) especifica la ventana cuya barra de desplazamiento se va a modificar.
nBarra (int) especifica qué cuadro de desplazamiento se va a modificar.

Valor	Descripción.
SB_CTL hWnd	Establece la posición de un control de una barra de desplazamiento, suponiendo que apunta al manipulador de ventana de ese control de barra de desplazamiento.
SB_HORZ	Establece la posición de una barra de desplazamiento horizontal.
SB_VERT	Establece la posición de una barra de desplazamiento vertical.

nPos (int) especifica la nueva posición dentro del rango de desplazamiento válido.
bRedibujar (BOOL) indica si se debe redibujar o no la barra de desplazamiento. Un valor distinto de cero indica que se debe redibujar la barra de desplazamiento. Si es cero, no se redibuja.

Valor devuelto.

Indica la posición anterior del cuadro de desplazamiento de la barra.

API : void SetScrollRange (hWnd, nBarra, nPosMin, nPosMax, bRedibujar)

Función.

Esta función selecciona el valor máximo y el valor mínimo de posición para la barra de desplazamiento seleccionada.

Parámetros.

hWnd (HWND) especifica la ventana cuya barra de desplazamiento se ve afectada.
nBarra (int) especifica que cuadro de desplazamiento se va a modificar:

Valor	Descripción.
SB_CTL hWnd	Establece la posición de un control de una barra de desplazamiento, suponiendo que apunta al manipulador de ventana de ese control de barra de desplazamiento.
SB_HORZ	Establece la posición de una barra de desplazamiento horizontal.
SB_VERT	Establece la posición de una barra de desplazamiento vertical.

nPosMin (int) establece la posición mínima de la barra de desplazamiento
nPosMax (int) establece la posición máxima de la barra de desplazamiento.
bRedibujar (BOOL) indica si se debe redibujar o no la barra de desplazamiento.
Un valor distinto de cero indica que se debe redibujar la barra de desplazamiento. Si es cero, no se redibuja.

Valor devuelto.
Ninguno.

Notas.

Si se llama a SetScrollRange justo después de llamar a SetScrollPos, se debe poner a cero bRedibujar, para evitar que la barra de desplazamiento se dibuje dos veces.

API : DWORD SetTextColor (hDC, rgbColor)

Función.
Esta función establece el color del texto.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
rgbColor o crColor (DWORD) selecciona un valor RGB del color que se va usar para la salida de texto.

Valor devuelto.

Indica el valor RGB del color usado para la salida de texto.

Notas.

SetBkColor se usa para seleccionar el color de fondo.

API : int SetTextJustification (hDC, nEspExtra, nCont)

Función.

Esta función justifica un texto usando los parámetros nespExtra y nCont.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
nEspExtra (int) selecciona la cantidad total de espacio adicional que se añade a la línea de texto.

nCont (int) selecciona el número de caracteres de separación que se añaden a la línea.

Valor devuelto.

Indica el resultado de la función. El valor 1 indica que la operación ha tenido éxito; en caso contrario, se devuelve el valor 0.

Notas.

El carácter de separación usado para delimitar las palabras es el ASCII 32 o carácter de espacio en blanco. Si se llama a GetTextMetrics, se puede obtener el carácter de separación de la fuente actual.

API :DWORD SetViewportExt (hDC, X, Y)

Función.

Esta función establece las dimensiones x u y de la superficie de proyección del contexto de dispositivo seleccionado.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo X (int), usando unidades del dispositivo, especifica el alcance horizontal de la superficie de proyección.
y (int), usando unidades del dispositivo, especifica el alcance vertical de la superficie de proyección.

Valor devuelto.

Contiene las anteriores dimensiones (alcances) de la superficie de proyección; la palabra de mayor peso contiene el anterior alcance vertical, la de menor peso, el anterior alcance horizontal . Si se devuelve el valor NULL, indica que se a producido un error. CSize contiene los alcances anteriores.

Notas.

Cuando los siguientes modos de proyección están activados, se ignoran las posteriores llamadas a SetWindowExt o SetViewportExt: MM_HIENGLISH, MM_HIMETRIC, MM_LOENGLISH, MM_LOMETRIC, MM_TEXT o MM_TWIPS.

API : DWORD SetWindowExt (hDC, X, Y)

Función.

Esta función establece el alcance horizontal y vertical de la ventana del contexto de dispositivo seleccionado.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
X (int), usando unidades del dispositivo, especifica el alcance horizontal de la ventana.
Y (int), usando unidades del dispositivo, especifica el alcance vertical de la ventana.

Valor devuelto.

Contiene las anteriores dimensiones (alcances) de la ventana; la palabra de mayor peso contiene el anterior alcance vertical, la de menor peso, el anterior alcance horizontal. Si se devuelve el valor NULL, indica que se ha producido un error.

Notas.

Cuando los siguientes modos de proyección están activados, se ignoran las posteriores llamadas a SetWindowExt: MM_HIENGLISH, MM_HIMETRIC, MM_LOENGLISH, MM_LOMETRIC, MM_TEXT o MM_TWIPS.

API : BOOL TextOut(hDC, X, Y, lpCadena, nCon)

Función.
Esta función escribe una cadena de caracteres en la pantalla seleccionada.

Parámetros.

hDC (HDC) especifica el contexto de dispositivo.
X (int) especifica la coordenada lógica x del principio de la cadena.
Y (int) especifica la coordenada lógica y del principio de la cadena.
lpCadena (LPSTR) apunta una cadena terminada con el carácter NULL que se va a dibujar.
nCont (int) indica el número de caracteres de la cadena a dibujar.

Valor devuelto.
VERDADERO (TRUE) si la cadena se dibujó con éxito, en caso contrario, se devuelve FALSE.

Notas.

TextOut no usa la posición actual ni la actualiza. El origen de todos los caracteres se define como la esquina superior izquierda de la posición del carácter.

Capítulo II.- DESARROLLO DE FUNCIONES.

II-1-- Estructura , objetivo y Programación de las funciones a través de "C".

Hoy en día, todos los programadores se enfrentan al reto de mejorar lo existente o bien de desarrollar alguna nueva aplicación. En la elaboración de una nueva aplicación, hay que considerar varios factores como son el ambiente de trabajo y el lenguaje a utilizar para la programación. La correcta selección de estos factores se vera reflejada en la calidad del producto final. En este capítulo desarrollaremos algunas funciones bajo una plataforma Windows, a través del lenguaje C. Las funciones que desarrollaremos están estructuradas del tal forma que pueden adaptarse fácilmente a diferentes aplicaciones. En el desarrollo de estas se utilizaron algunas de las funciones de Windows antes descritas.

Como mencionamos antes , todas las aplicaciones Windows poseen menús , barras de desplazamiento, cuadros de diálogo, iconos, botones y otras muchas características de un interfaz gráfico, para lo cual Windows incorpora herramientas que nos permiten el fácil desarrollo de cualquiera de estas opciones. Como programadores tenemos que estar preparados para afrontar cualquier problema, habrá ocasiones en las que será necesaria la elaboración de funciones básicas que se acondiciones a nuestras necesidades, con funciones básicas me refiero a todas aquellas funciones que nos servirán de apoyo en la elaboración de nuestra aplicación. La finalidad de estas funciones es mas que nada obtener una mejor presentación en nuestras aplicaciones. Si desea ver la ejecución independiente de las funciones que a continuación se explicaran, podrá consultar el software de apoyo "Manual".

- Funciones de texto.

Durante el desarrollo de la aplicación de ejemplo que se presenta junto con este libro, se presentan algunos efectos de texto, como es el del texto que va apareciendo progresivamente ó bien los efectos de los números que aparentan ir avanzando. Estos efectos tienen como fin dar una mejor presentación a la aplicación . A continuación se presenta la estructura de esas dos sencillas funciones y si desea ver la ejecución de estas funciones junto con su código podrá consultar el software de apoyo denominado "Manual".

--- Función de aparición de texto.

En funciones de texto tenemos dos muy semejantes. Ambas funciones nos presentan el texto de manera progresiva en la pantalla, pero difieren en que una nos presenta el texto dentro de un recuadro y en una posición fija, mientras que la otra función presenta al texto sin recuadro y en lugares arbitrarios. Ambas funciones son muy semejantes en su código , pero la primera presenta un código más completo , por lo tanto la utilizaremos como modelo para explicarla a continuación.

Esta función dentro de la aplicación se identifica con el nombre "VER_CADENA_CON", la cual para su ejecución requiere de los siguientes parámetros:

```
void VER_CADENA_CON ( HWND hwnd   // identifica al gestor de la ventana.
, int f_r
, int f_v
, int f_a  // estas tres variables definen la intensidad de los cañones "RGB" para el color del texto
, int f_r
, int f_v
, int f_a  // estas tres variables definen la intensidad de los cañones "RGB" para el color del fondo .
,char *c
,char a/f  // estas variables van a recibir el texto que se presentara en la pantalla.
,int s  // dependiendo del valor ( 0 o 1) que reciba esta variable, se emitirá un sonido al escribir el texto.
,int x1
,int y1  // estas variables contienen las coordenadas X y Y para la colocación de la primer cadena.
,int x2
,int y2  // estas variables contienen las coordenadas X y Y para la colocación de la segunda cadena.
,int t1  // contiene una variable entera con el tamaño de la primera cadena de texto.
```

,int t2 // contiene una variable entera con el tamaño de la segunda cadena de texto.
)

Dentro del código de esta función, se encuentran varias declaraciones de variables, como son:

HBRUSH Brocha, Brocha1; // declara variables para crear la brocha que dará el color de fondo.
HPEN Pluma, Pluma1; // declara variables para crear la pluma que dará el color de contorno.
HDC hdc; // gestor del contexto de visualización.

También encontramos algunas funciones de Windows tales como:

GetClientRect	// recupera el tamaño de la ventana.
hdc=GetDC	// establece el contexto de visualización.
SetMapMode	// establece el modo de coordenadas de pantalla arbitrarias.
SetWindowExt	// define el tamaño lógico de la pantalla
SetViewportExt	// establece el tamaño lógico de la pantalla.
TextOut	// esta función escribe el texto en la pantalla
Rectangle	// dibuja un rectángulo en la pantalla con el color de la brocha y de la pluma.
CreateFont	// esta función establece un tipo de letra con las características deseadas.

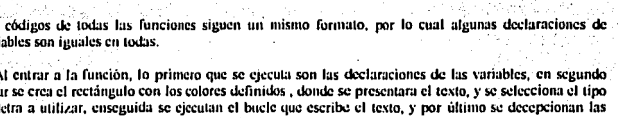
Este es el bucle que escribe el texto:

```
while(a<t1){
TextOut(hdc,x1,y1,c,a);
TIEMPO(cuenta1);
a++; }
while(b!=t2){
TextOut(hdc,x2,y2,f,b);
TIEMPO(cuenta1);
b++; }
```

Los códigos de todas las funciones siguen un mismo formato, por lo cual algunas declaraciones de variables son iguales en todas.

Al entrar a la función, lo primero que se ejecuta son las declaraciones de las variables, en segundo lugar se crea el rectángulo con los colores definidos , donde se presentara el texto, y se selecciona el tipo de letra a utilizar, enseguida se ejecutan el bucle que escribe el texto, y por último se decepcionan las variables para liberar la memoria que ocupaban

Imagen que ejemplifica la ejecución de la función:



--- Función de movimiento de letras.

Esta función se utiliza dentro de la aplicación en la parte correspondiente al modulo "Memoria" y en el modulo "Control". Esta función da la apariencia de que los números se van desplazando. Parte del código que conforma a esta función se explica a continuación. Si desea ver solamente la ejecución de esta función, consulte el software de apoyo "Manual".

Para empezar se declaran las variables a utilizar:

int t,v,z,w,f,e g,h i,n; // se declaran algunas variables enteras

```
char cadena1[22]="110100          ";   // declaración de las cadenas a mostrar.
char *s,*d;                            // declaración de punteros para las cadenas de texto.

HBRUSH Brocha, Brocha1;                // variables para la brocha
HPEN Pluma1, Pluma;                    // variables para la pluma

POINT flecha4[7];                      // se declaran arreglos de puntos para crear las flechas.
```

A continuación se dan las coordenadas para dibujar las flechas:

```
flecha4[0].x = 170;  flecha4[0].y = 200;
flecha4[1].x = 170;  flecha4[1].y = 170;
flecha4[2].x = 300;  flecha4[2].y = 170;
flecha4[3].x = 300;  flecha4[3].y = 160;
flecha4[4].x = 330;  flecha4[4].y = 185;
flecha4[5].x = 300;  flecha4[5].y = 210;
flecha4[6].x = 300;  flecha4[6].y = 200;
```

Aquí se selecciona el color de la brocha y de la pluma y se dibujan las flechas.

```
SetCursor(Cursor_Cruz);
Pluma= CreatePen(PS_SOLID,1,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= CreateSolidBrush(RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);

    Polygon(hdc,flecha4,7);            // función para dibujar las coordenadas de puntos (flechas).

SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);          // estas instrucciones liberan la memoria de la brocha y de la pluma.

VER_CADENA_SIN(hwnd,23,10,"LA U. DE CONTROL BUSCA Y TRAE UNA INSTRUCCION DE
LA MEMORIA ENVIANDO",68,0,0,255);          // aquí se aplica la función que explicamos
anteriormente.
```

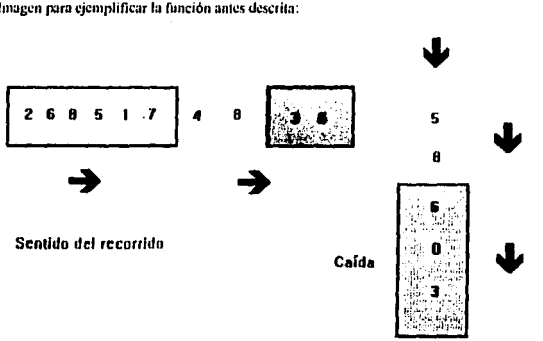
Este es el bucle que da movimiento a los números.

```
s=cadena;
f=55;e=0;
w=6;g=0;
i=0;h=0;
v=0;
n=290;

  TIEMPO(cuenta);

while(e!=7){
  f=f+(10*e);
  h=w-e;
 while(g!=h){
  s=&cadena[g];
  TextOut(hdc,f,180,s,1);
  f=f+12;
```

```
g++;
  }
 s=&cadena[6-i];
 TextOut(hdc,n,180,s,1);
 MessageBeep(0);
  TIEMPO(cuenta);
 Rectangle(hdc,42,155,135,195);

 i++;
 e++;
 g=0;
 n=12;
 f=50;
 }

SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
```

El efecto para dar movimiento es muy sencillo. Se toma el arreglo que contiene a la cadena que se va a mostrar, posteriormente este arreglo entra en un ciclo while, el cual toma un carácter del arreglo y lo muestra en la posición indicada, enseguida esta otro ciclo while que realiza la misma función pero con el siguiente carácter del arreglo, y así sucesivamente. De esta manera la cadena de texto se va recorriendo a lo largo de los While, desde el primero hasta el último.

Imagen para ejemplificar la función antes descrita:



- Funciones de gráficos.

Nombramos funciones de gráficos, a aquellas funciones que realizan tareas gráficas. Dentro de esta categoría mostraremos dos funciones: la primera es una función que dibuja los rectángulos bidimensionales, en la segunda se muestra el manejo de la función "Polygon" para dibujar figuras en base a coordenadas de puntos, y también se muestra como simular figuras que estén parpadeando.

--- Función para crear rectángulos bidimensionales

Esta función se utiliza muy frecuentemente dentro de la aplicación, debido al continuo manejo de figuras que simulan dispositivos en la pantalla. El código de esta figura inicia con declaraciones de variables, posteriormente viene el código que dibuja las figuras, y por último se liberan las herramientas utilizadas para liberar la memoria que ocupaban.

El prototipo y los parámetros de la función son los siguientes:

```
 void CREA_RECTANGULO_BIDIMENSIONAL (HWND hwnd   //  gestor de la ventana.
,int x1
,int y1
,int x2
,int y2              //  estas variables establecen el tamaño y ubicación de la figura.
,int c1
,int c2
,int c3              //  estas variables establecen el color de relleno de la figura.
,int orientación   //  esta variable indica la orientación que tendrá la figura.
 )

-- Declaración de variables.

HBRUSH Brocha, Brocha1;          //  declaración de variables para las brochas.
 HPEN Pluma1, Pluma;              //  declaración de variables para las plumas.

POINT puntos[4];
POINT puntos1[4];          //  se declaran dos arreglos de 4 puntos.

HDC hdc;                          //  contexto de visualización
GetClientRect(hwnd, &limite);     //  obtiene el tamaño de la ventana de trabajo.
hdc=GetDC(hwnd);                  //  declara el contexto de visualización.
SetMapMode(hdc, MM_ANISOTROPIC);      //  se selecciona el modo arbitrario de coordenadas.
SetWindowExt(hdc, 624,442);                    //  establece el tamaño lógico de la pantalla.
SetViewportExt(hdc, limite.right, limite.bottom);   //  establece las coordenadas lógicas.
- Este bucle da las coordenadas y orientación a la figura.

switch(orientación){
 case 1:
        puntos[0].x=x1;  puntos[0].y=y1;
        puntos[1].x=x1-20;  puntos[1].y=y1-20;
        puntos[2].x=x2-20;  puntos[2].y=y1-20;
        puntos[3].x=x2;  puntos[3].y=y1;

        puntos1[0].x=x1;  puntos1[0].y=y1;
        puntos1[1].x=x1-20;  puntos1[1].y=y1-20;
        puntos1[2].x=x1-20;  puntos1[2].y=y2-20;
        puntos1[3].x=x1;  puntos1[3].y=y2;
  break;
 case 2:
        puntos[0].x=x1;  puntos[0].y=y1;
        puntos[1].x=x1+20;  puntos[1].y=y1-20;
        puntos[2].x=x2+20,  puntos[2].y=y1-20;
        puntos[3].x=x2,  puntos[3].y=y1;

        puntos1[0].x=x2;  puntos1[0].y=y2;
        puntos1[1].x=x2+20,  puntos1[1].y=y2-20;
        puntos1[2].x=x2+20;  puntos1[2].y=y1-20;
        puntos1[3].x=x2;  puntos1[3].y=y1;
  break;
```

}

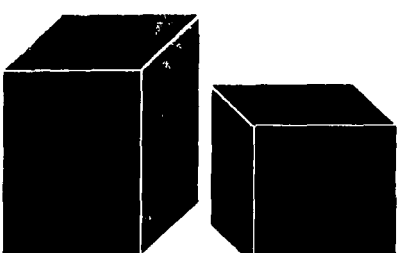
- En esta parte se dibuja la figura.

```
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= CreateSolidBrush(RGB(c1,c2,c3));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,x1,y1,x2,y2);                    //   dibuja la figura.
Polygon(hdc,puntos,4);
Polygon(hdc,puntos1,4);                        //   dibuja la proyección de la figura.
```

-- Se liberan las herramientas.

```
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
ReleaseDC(hwnd,hdc);
```

Imagen de ejemplo:



-- Función para dibujar figuras a base de puntos y simular que parpadean.

La construcción de figuras por medio de puntos se utiliza mucho en la aplicación, por ejemplo en la construcción de las flechas que muestran los flujos. En esta función se encuentran las coordenadas para dibujar todas las flechas, pero debido a que es muy larga solo se mostrara una parte del código.

En esta función también se muestra como se puede simular el parpadeo en las figuras. Para lograr esto, solamente se tienen que declarar varias opciones de color en las brochas y plumas, todas ellas se ponen dentro de un "case" y dependiendo del valor en cada llamada a la función, el color de la figura será diferente. Lo único que restaría es poner la llamada a la función dentro de un ciclo por, en el cual cada incremento variara el color de la figura.

- A continuación se muestra el prototipo y parámetros de la función:

```
void FLECHAS_DATOS(HDC hdc     // gestor o manejador de la ventana.
,int k                         // esta variable establece el color de la brocha y de la pluma.
,int y                         // esta variable selecciona la flecha a dibujar.
,int g )                       // esta variable elige el sonido que se emitirá al crear la figura.
```

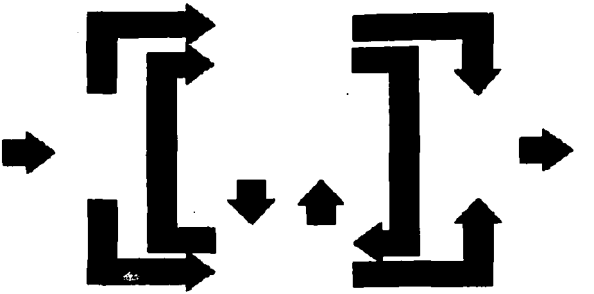
- Esta parte es la de declaración de variables.

```
HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;                    // variables de brocha y pluma.
POINT puntos1[7];
POINT puntos2[7];
POINT puntos3[7];                      // establece las estructuras que almacenaran las coordenadas.

puntos1[0].x = 580-X;  puntos1[0].y = 225+Y;
puntos1[1].x = 605-X;  puntos1[1].y = 225+Y;
puntos1[2].x = 605-X;  puntos1[2].y = 215+Y;
puntos1[3].x = 635-X;  puntos1[3].y = 240+Y;
puntos1[4].x = 605-X;  puntos1[4].y = 265+Y;
puntos1[5].x = 605-X;  puntos1[5].y = 255+Y;
puntos1[6].x = 580-X;  puntos1[6].y = 255+Y;    // coordenadas para la flecha.

puntos2[0].x = 25-X;  puntos2[0].y = 225+Y;
puntos2[1].x = 50-X;  puntos2[1].y = 225+Y;
puntos2[2].x = 50-X;  puntos2[2].y = 215+Y;
puntos2[3].x = 80-X;  puntos2[3].y = 240+Y;
puntos2[4].x = 50-X;  puntos2[4].y = 265+Y;
puntos2[5].x = 50-X;  puntos2[5].y = 255+Y;
puntos2[6].x = 25-X;  puntos2[6].y = 255+Y;     // coordenadas para otra flecha.

puntos3[0].x = 305-X;  puntos3[0].y = 275+Y;
puntos3[1].x = 305-X;  puntos3[1].y = 300+Y;
puntos3[2].x = 315-X;  puntos3[2].y = 300+Y;
puntos3[3].x = 290-X;  puntos3[3].y = 330+Y;
puntos3[4].x = 265-X;  puntos3[4].y = 300+Y;
puntos3[5].x = 275-X;  puntos3[5].y = 300+Y;
puntos3[6].x = 275-X;  puntos3[6].y = 275+Y;    // coordenadas para la última flecha.


switch(k){          // en este case se elige el color que tendrán las flechas

case 1:                                          // primer color.
  Brocha= CreateSolidBrush(RGB(0,0,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

case 2:                                          // segundo color.
  Brocha= CreateSolidBrush(RGB(0,0,127));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

case 3:                                          // tercer color.
  Brocha= CreateSolidBrush(RGB(127,127,127));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

  }
```

```
switch(y){                    // en este case se elige la flecha a dibujar, el efecto para que aparenten
                                 que parpadean consiste en crear un bucle por que contenga dos
                              llamadas
                              a esta función, pero cada llamada con diferente color.

 case 1:
  Polygon(hdc,puntos1,7);
  switch(g){                          // este case elige el sonido a emitir
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 2:
  Polygon(hdc,puntos2,7);
  switch(g){                          // este case elige el sonido a emitir
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 3:
  Polygon(hdc,puntos3,7);
  switch(g){                          // este case elige el sonido a emitir
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;

  }

 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);              // se liberan y se borran las herramientas utilizadas.

}
```

Imagen de la ejecución de la función.

Nota: Cada flecha requiere de una llamada a la función.



- Funciones de pantalla (manejo de mapas de bits).

Windows nos ofrece funciones para el manejo de mapas de bits. Con esto podemos manejar el tamaño de las imágenes, combinar imágenes, encimar imágenes, crear efectos de animación, etc. Estas funciones nos dan la posibilidad de crear todos los efectos que imaginemos, pudiendo combinar estos con las demás funciones gráficas de Windows.

Dentro del tutorial tenemos algunos efectos de pantalla, pero como todos siguen el mismo principio, explicaremos solamente el efecto que se ejecuta cuando se entra inicialmente al tutorial.

-- Efecto de cortina (manejo de mapa de bits).

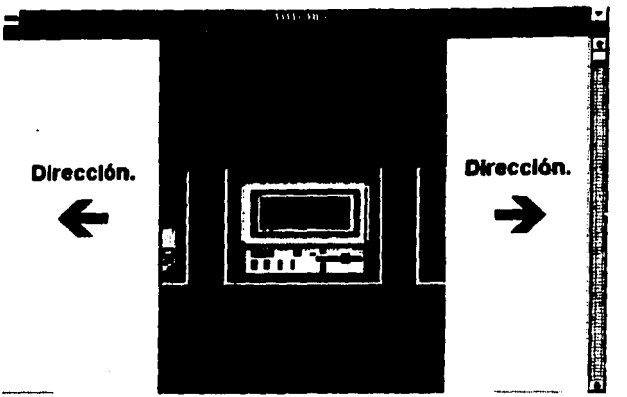
Esta función a pesar de ser muy pequeña, genera un buen efecto. El efecto se basa principalmente en el manejo de bucles , los cuales controlan las coordenadas de pantalla, y para mostrar la imagen en la pantalla se utilizo la función "BitBlt".

El código para esta función es el siguiente:

```
int r,d1=0,d2=320;    // se declaran variables para iniciar valores.
HDC hdc;              // variable para el contexto de visualización.
hdc=GetDC(hwnd);                    // establece el contexto de visualización.

hbmpSeleccion = LoadBitmap(hInst, "MyBitmap3");   //  se carga el mapa de bits.
hdcSeleccion = CreateCompatibleDC(hdc);                  //  Se crea un contexto visual en memoria.
hbmpSeleccion1 = SelectObject(hdcSeleccion, hbmpSeleccion);}  // se selecciona el mapa de bits.

PatBlt(hdc,0,0,640,442,BLACKNESS);                  // pone la ventana de trabajo en color negro.
while(d1<=320){                                     // bucle para controlar la aparición de la imagen.
BitBlt(hdc,320,0,d1,442,hdcSeleccion,320,0,SRCCOPY);    //    función para copiar la imagen en la
pantalla.
BitBlt(hdc,d2,0,d1,442,hdcSeleccion,d2,0,SRCCOPY);      //    función para copiar la imagen en la
pantalla.
d1+=2;
d2-=2;    // variables para incrementar la porción de imagen visible.
 }
```

```
ReleaseDC(hwnd,hdc);        //  se libera el contexto de visualización.

}
```

Imagen tomada durante la ejecución de la función.



- Funciones para crear botones.

Dentro de las utilerías que incorpora Borlande C++, se encuentra la herramienta "WorkShop", esta utilería sirve para crear menús, cajas de diálogos, iconos, botones, bitmaps, etc. En la aplicación que se presenta, se utilizan muchos botones de diferente tipos, pero no se generaron con la herramienta "WorkShop", debido a que esta no se adecuaba a las necesidades, y por que para poderlos utilizar se

En esta aplicación se presenta una alternativa para la generación de botones. El tutorial cuenta con sus propias funciones para crear botones, de esta manera si no se cuenta con la herramienta WorkShop, el usuario podrá generar sus botones con las funciones que aquí se presentan.

A continuación se presenta el código de esta función:

- Prototipo y parámetros de la función.

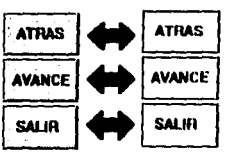
```
void CREA_BOTONES (HDC hdc    // gestor o manejador de la ventana.
,int a
, int b
, int c
, int d          //  estas variables determinan la posición y el tamaño del botón.
,char *s         //  este puntero contiene el texto que contendrá el botón
,int f           //  esta variable contiene el tamaño de la cadena de texto del botón.
,int p           //  este botón determina la posición ON / OFF del botón.
)

{
```

– Declaración de variables.

```
int w,x,y,z,g=0,h=0,i=0,j=0,k=0,l=0,v=0;
HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;

w=a; x=b; y=c; z=d;              //  asigna a otras variables la ubicación y tamaño del botón.
Rectangle(hdc,a,b,c,d);          //  crea con los colores por default el contorno del botón.

switch(p){                       // en este bucle se elige la posición ON (1) / OFF (2) que tendrá el botón.

case 1:

SetTextColor(hdc,RGB(0,0,0));              //  se elige el color del texto.
SetBkMode(hdc,TRANSPARENT);                //  se elige el color de fondo.

Pluma= CreatePen(PS_SOLID,2,RGB(127,127,127));    // se crea una pluma ( ancho y color).
Pluma1= SelectObject(hdc,Pluma);                  // se selecciona la pluma.
MoveTo(hdc,w+1,z-2);                       // posiciona la pluma en estas coordenadas.
LineTo(hdc,y-2,z-2);                       // dibuja una linea del punto anterior a estas coordenadas.
LineTo(hdc,y-2,x+1);                       // dibuja una linea del punto anterior a estas coordenadas.
                                           //  estas líneas dibujan la sombra del botón.
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);                      // se selecciona y se borra la pluma.


  Brocha= CreateSolidBrush(RGB(191,191,191));          //  se crea una brocha sólida.
  Pluma= CreatePen(PS_SOLID,1,RGB(191,191,191));       //  se crea una pluma.
  Brocha1= SelectObject(hdc,Brocha);                   //  se selecciona la brocha.
  Pluma1= SelectObject(hdc,Pluma);                     //  se selecciona la pluma.

h=f*7; v=y-w; g=v-h; i=(g/2);
j=z-x; k=j-8; l=(k/2)+x-4;                    //  estas lineas sirven para centrar el texto dentro del
botón.

Rectangle(hdc,a+3,b+3,c-3,d-3);          // dibuja el botón.
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);                    //  se deseleccionan y borran las brochas y las plumas.
TextOut(hdc,a+i-4,l,s,f);                //  escribe el texto dentro del botón.

break;

case 2:                          //  para esta opción se repite los parámetro anteriores, solo cambia el color del
botón.

SetTextColor(hdc,RGB(127,0,0));
SetBkMode(hdc,TRANSPARENT);

  Brocha= CreateSolidBrush(RGB(191,191,191));
  Pluma= CreatePen(PS_SOLID,1,RGB(191,191,191));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);

h=f*7; v=y-w; g=v-h; i=(g/2)+1;
j=z-x; k=j-8; l=(k/2)+x-4;
```

```
  Rectangle(hdc,a+1,b+1,c-3,d-3);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  SelectObject(hdc,Pluma1);
  DeleteObject(Pluma1);
  TextOut(hdc,a+i-4,l,s,f);

 break;

   }
}
```

Imagen demostrativa:



Todas las funciones antes mencionadas, conforman la estructura básica del tutorial que se desarrollo. Claro que el tutorial cuenta con muchas más funciones que son las que realmente lo conforman, pero todas ellas se apoyan en las funciones básicas.

Si se analizan las funciones que componen al tutorial, podrá encontrar de interés la gran variedad de combinaciones y opciones que tiene al combinar todas las funciones básicas. Estas funciones serán analizadas en los siguientes capítulos

Capitulo III.- DESARROLLO DE UNA APLICACIÓN UTILIZANDO LAS FUNCIONES DESARROLLADAS EN EL CAPITULO II.

III.1.- Estructura de la aplicación.

En este capitulo se desarrolla una aplicación utilizando las funciones desarrolladas en el capitulo anterior. Esta aplicación no solo pretende ser un ejemplo para mostrar algunas funciones, se pretende que realmente sirva como apoyo en el tema que aborda.

La aplicación esta compuesta aproximadamente de 7000 líneas de código, por lo que pasa a ser más que un simple ejemplo. Todas las funciones siguen un mismo formato, por lo tanto solo analizaremos algunas de ellas. La estructura de las funciones siguen el mismo formato que se ha venido manejando, comenzando por la declaración e inicialización de variables así como de herramientas a utilizar, posteriormente se presenta lo que serie el código principal, y por último tenemos la liberación de la memoria que ocupaban las herramientas utilizadas (brochas, plumas, ,mapas de bits, contexto de visualización, etc). Podría pensarse que cada función desarrollada esta compuesta solamente por llamadas a funciones básicas, pero en realidad se encuentran complementadas con otras funciones de Windows. Esto se debe a que cada función tiene caracteristicas especificas, por lo que no se pueden utilizar funciones generales en estas partes, es decir, la programación de las funciones básicas se realizo considerando el grado de frecuencia con que se requería determinados procesos. Debido a esto, no era viable programar funciones especificas que solo servirían para un caso en especial, por lo que mejor se programaron directamente en las funciones complementando a las funciones básicas.

La aplicación (tutorial) viene dividida por sesiones, donde cada sesión toca algún punto en particular. En los códigos de estas funciones veremos repetidamente el uso de las funciones básicas desarrolladas en el capitulo anterior, y se darán cuenta de la sencillo que es aplicar estas funciones a sus necesidades.

La parte que podría ser un poco mas problemática es el desarrollo del gestor de mensajes, como se menciono antes la clave para entender la programación en Windows es el paso de mensajes. Debido a esto la función correspondiente al gestor de mensajes se explicara con más detalle. Esta función es muy importante, ya que es quien controla realmente a la aplicación, en ella los mensajes se canalizan para generar las acciones correspondientes.

Todos los mensajes se identifican con las siglas "WM_mensaje" , donde "mensaje" es el tipo de mensaje. Existen mensajes para poder captar cualquier evento que pueda generar la aplicación. Aquí solo hablaremos de los mensajes que fueron utilizados en la elaboración de la aplicación.

Iniciaremos por explicar el modulo correspondiente al gestor de mensajes, y posteriormente se dará un ejemplo del funcionamiento de este:

```
// FUNCIÓN GESTOR DE MENSAJES(distribuidor de mensajes)

LONG FAR PASCAL zWndProc ( HWND hWnd     // gestor de la ventana.
, unsigned message          //  indica que tipo de mensaje
, WORD wParam
, LONG lParam               //    contienen información importante asociada a los mensajes.
) {


// primero tenemos el prototipo de la función , y todos sus parámetros.
// se utiliza el convenio de llamadas de Pascal por ser más eficiente y ahorrar espacio.

   PAINTSTRUCT ps;    // especifica la estructura de dibujo que se utiliza para dibujar dentro de una
                         ventana.
```

```
    HDC hdc;                    //  variable de contexto de visualización.
    TEXTMETRIC tm;              //  guarda información del texto.
    GetClientRect(hWnd, &limite);           //  recupera el tamaño de la ventana.
    factor_ancho=(float)limite.right/624;
     factor_alto=(float)limite.bottom/442;      //  estas variables contienen un factor de conversión de
escala.

switch (message)                    //  se canaliza según el mensaje generado.
 {
 case WM_COMMAND:               //  identifica alguna selección del menú.
  switch(wParam)                    //  identafica que opción del menú de activo.
   {
   case TXT_SALIR:                  //  todos los "TXT_ nombre" identifican alguna opción del menú.
   if(seleccion=='TRUE){
    mensaje=MessageBox(GetFocus(),"SALIR DE LA APLICACION","ESCAPE",MB_OKCANCEL |
   MB_ICONSTOP);   //  genera un cuadro de mensaje con dos botones y un icono.
    if(mensaje == IDOK){            //  IDOK es el valor generado al seleccionar "Ok" del cuadro de
mensaje.
    SelectObject(hdcMemory,hbmpOld);   /* ...deseleccionarlo */
    SelectObject(hdcCuadro,hmbCuadro1);
    SelectObject(hdcSeleccion,hbmpSeleccion1);
    SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen1);

    DeleteObject(hbmpUltima_Imagen);
    DeleteObject(hmbCuadro);  /* ...borrar los mapas de bits ocultos... */
    DeleteDC(hdcMemory);        /* ...borrar el DC en memoria... */
    DeleteDC(hdcCuadro),
    DeleteDC(hdcSeleccion);
    SelectObject(hdc,PrelFuente);
    PostQuitMessage(0);}}

                if(ayuda==TRUE){mensaje=MessageBox(GetFocus(),"DESEA SALIR DE LA
   AYUDA","AYUDA",MB_OKCANCEL | MB_ICONINFORMATION);
        if(mensaje==IDOK){
       ayuda=FALSE;
       switch(variable){
     case 0:
      seleccion=TRUE; break;
     case 1:
         ejecucion=TRUE; break;
         case 2:
         ejecucion_1=TRUE; break;
         case 3:
         ejecucion_1_1=TRUE; break;
        case 4:
         evaluacion=TRUE; break;
         }
          hdc=GetDC(hWnd);

BitBlt(hdc,0,0,limite.right,limite.bottom,hdcUltima_Imagen,0,0,SRCCOPY);//StretchBlt(hdc,0,0,limite.
right,limite.bottom,hdcUltima_Imagen,0,0,limite.right,limite.bottom,SRCCOPY);
          ReleaseDC(hWnd,hdc);}}

    break;
   case TXT_CONCEPTOS_BASICOS:
       mensaje=MessageBox(GetFocus(),"CONCEPTOS_BASICOS","BASICOS",MB_OKCANCEL |
MB_ICONINFORMATION);
   if(mensaje == IDOK){
```

```
      if(seleccion==TRUE){variable=0; seleccion=FALSE;}
         if(ejecucion==TRUE){variable=1; ejecucion=FALSE;}
          if(ejecucion_1==TRUE){variable=2, ejecucion_1=FALSE;}
           if(ejecucion_1_1==TRUE){variable=3; ejecucion_1_1=FALSE;}
            if(evaluacion==TRUE){variable=4; evaluacion=FALSE;}

     ayuda=TRUE;
      AYUDA(hWnd);}
      break;
      case TXT_DEMO_1:
            mensaje= MessageBox(GetFocus(),"ESQUEMA GENERAL","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
       if(mensaje==IDOK){
        if(seleccion==TRUE){seleccion=FALSE;}
           if(ejecucion==TRUE){ejecucion=FALSE;}
            if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
             if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
              if(evaluacion==TRUE){evaluacion=FALSE;}
              ejecucion=TRUE;
              var_z=1;       // PARA INICIALIZAR EL PRIMER MODULO
              APLICACIÓN(hWnd,var_z);} break;
      case TXT_DEMO_2:
         mensaje= MessageBox(GetFocus(),"FUNCION POR MODULO","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
       if(mensaje==IDOK){
        if(seleccion==TRUE){seleccion=FALSE;}
           if(ejecucion==TRUE){ejecucion=FALSE;}
            if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
             if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
              if(evaluacion==TRUE){evaluacion=FALSE;}
         ejecucion_1=TRUE;
            APLICACION_1(hWnd);} break;
      case TXT_DEMO_3:
         mensaje    = MessageBox(GetFocus(),"MICROCOMPUTADORA","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
       if(mensaje==IDOK){
        if(seleccion==TRUE){seleccion=FALSE;}
           if(ejecucion==TRUE){ejecucion=FALSE;}
            if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
             if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
              if(evaluacion==TRUE){evaluacion=FALSE;}
            ejecucion_1_1=TRUE;
         micro=TRUE;
            MICROCOMPUTADORA_1(hWnd);} break;
     case TXT_AYUDA_PRINCIPAL:
           LoadString(hInst,TEX_AYUDA,cadena_titulos,Max);
           LoadString(hInst,TEX_AYUDATEX,cadena_textos,MaxText);
           mensaje=MessageBox(GetFocus(),cadena_textos,cadena_titulos,MB_OKCANCEL |
MB_ICONINFORMATION);
       if(mensaje == IDOK){
        if(seleccion==TRUE){variable=0; seleccion=FALSE;}
           if(ejecucion==TRUE){variable=1; ejecucion=FALSE;}
            if(ejecucion_1==TRUE){variable=2; ejecucion_1=FALSE;}
             if(ejecucion_1_1==TRUE){variable=3; ejecucion_1_1=FALSE;}
              if(evaluacion==TRUE){variable=4; evaluacion=FALSE;}

           ayuda=TRUE;
```

```
    AYUDA(hWnd);}
      break;
   default:
    return(DefWindowProc(hWnd, message, wParam, lParam));
  }
  break;

 case WM_INITMENUPOPUP:    //    identifica la visualización activa o inactiva de un menú.
       if(seleccion==TRUE || ayuda==TRUE){
         EnableMenuItem(wParam,TXT_SALIR,MF_ENABLED);}
         else EnableMenuItem(wParam,TXT_SALIR,MF_GRAYED);
        if(ayuda==TRUE){
         EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_GRAYED);
         EnableMenuItem(wParam,TXT_DEMO_1,MF_GRAYED);
         EnableMenuItem(wParam,TXT_DEMO_2,MF_GRAYED);
         EnableMenuItem(wParam,TXT_DEMO_3,MF_GRAYED);
         EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_GRAYED);}
       else {
         EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_ENABLED);
         EnableMenuItem(wParam,TXT_DEMO_1,MF_ENABLED);
         EnableMenuItem(wParam,TXT_DEMO_2,MF_ENABLED);
         EnableMenuItem(wParam,TXT_DEMO_3,MF_ENABLED);
         EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_ENABLED);}
         break;

 case WM_LBUTTONDOWN:    //    detecta la pulsación del botón izquierdo del ratón.
       posicion_x= LOWORD(lParam);
       posicion_y= HIWORD(lParam);       // contienen las coordenadas del ratón.

   if(seleccion==TRUE){CHECA_SELEC(hWnd,posicion_x,posicion_y);}
      if(ejecucion==TRUE){CHECA_APL(hWnd,posicion_x,posicion_y);}
       if(ejecucion_1==TRUE){CHECA_APL_1(hWnd,posicion_x,posicion_y);}
        if(ejecucion_1_1==TRUE){CHECA_MOD(hWnd,posicion_x,posicion_y);}
         if(evaluacion==TRUE){CHECA_EVALUACION(hWnd,posicion_x,posicion_y);}
 break;

 case WM_CREATE:  //  obtiene información  del tamaño del texto actual usando a GetTextMetric.

  hdc=GetDC(hWnd);                    //   establece el contexto de visualización
  GetTextMetrics(hdc,&tm);            //  se utiliza para obtener información del texto actual.
  charwt=tm.tmAveCharWidth;
  charht=tm.tmHeight+tm.tmExternalLeading;   // contienen  al ancho y largo del texto.
  ReleaseDC(hWnd,hdc);                //  libera el contexto de visualización.
  SetScrollPos(hWnd,SB_VERT,posvertscroll,TRUE);   // establece la posición actual del cuadro de
desplazamiento de una barra de desplazamiento.
  SetScrollRange(hWnd,SB_VERT,0,SCROLLLINES,FALSE);    //   esta función selecciona el valor
máximo y el valor mínimo de posición para la barra de desplazamiento seleccionada.
  break;

 case WM_VSCROLL:          //  controla por medio de un bucle la barra de desplazamiento.
  if(ayuda==TRUE){
  switch (wParam)
  {
   case SB_LINEDOWN:
    posvertscroll+=1;
    break;
   case SB_LINEUP:
    posvertscroll-=1;
```

```
      break;
     case SB_PAGEDOWN:
      posvertscroll+=yClientView/charht;
      break;
     case SB_PAGEUP:
      posvertscroll-=yClientView/charht;
     case SB_THUMBPOSITION:
      posvertscroll=LOWORD(lParam);
      break;
     default:
      break;
    }
    if (posvertscroll>SCROLLLINES)
     posvertscroll=SCROLLLINES;
    if (posvertscroll<0)
     posvertscroll=0;
    if (posvertscroll!=GetScrollPos(hWnd,SB_VERT))
    {
     SetScrollPos(hWnd,SB_VERT,posvertscroll,TRUE);
     InvalidateRect(hWnd,NULL,TRUE);
    }
    } //fin del if
    break;

  case WM_PAINT:    //  identifica a los mensajes de ventana, los captura y toma una acción.

         hdc=BeginPaint(hWnd,&ps);   // prepara la ventana para que se dibuje.
         if(ayuda==TRUE){
         LECTOR_AYUDA(hWnd);       // llama a la función que lee el texto.
         }
         else
{BitBlt(hdc,0,0,limite.right,limite.bottom,hdcUltima_Imagen,0,0,SRCCOPY);}//StretchBlt(hdc,0,0,640,
442,hdcUltima_Imagen,0,0,640,442,SRCCOPY);}  // copia un mapa de bits en la ventana.
         EndPaint(hWnd,&ps);
         break;

  case WM_MOUSEMOVE:    //  detecta el desplazamiento del ratón.
         posicion_x= LOWORD(lParam);
         posicion_y= HIWORD(lParam);   contienen la posición X y Y del ratón.
         COORDENADAS(hWnd,posicion_x,posicion_y);    // llamada a una función.
    break;

  case WM_DESTROY:        /* detecta si la ventana esta  siendo destruida... */

     SelectObject(hdcMemory,hbmpOld);   /* ...deseleccionarlo */
     SelectObject(hdcCuadro,hmbCuadro1);
     SelectObject(hdcSeleccion,hbmpSeleccion1);
     SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen1);

     DeleteObject(hbmpUltima_Imagen);
     DeleteObject(hmbCuadro);  /* ...borrar los mapas de bits ocultos... */
     DeleteDC(hdcMemory);         /* ...borrar el DC en memoria... */
     DeleteDC(hdcCuadro);
     DeleteDC(hdcSeleccion);
     SelectObject(hdc,PreFuente);
     PostQuitMessage(0);                    /* limpiar y terminar */

   break;
```

```
 case WM_SYSCOMMAND:              /* detecta  una selección del menú del sistema. */

  if ((wParam & 0xfff0)== SC_MOVE)
   {        /* bloquea cualquier intento de mover la ventana .. */
   break; }

  if ((wParam & 0xfff0)== SC_SIZE)
   { /* bloquea cualquier intento de cambiar de tamaño la ventana */
   break;}

 default:  /* en caso contrario, pasa mensaje al gestor por defecto */
  return(DefWindowProc(hWnd, message, wParam, lParam));
 }
return FALSE;
}
```

Ahora que se ha explicado el funcionamiento del gestor de mensajes, continuaremos con un ejemplo para entender como es que este funciona. Supongamos que estamos trabajando en nuestra aplicación, y tomamos la opción "Demo_Mod" del menú que se corresponde con el identificador TXT_DEMO_2 de la función gestora de mensajes. En el momento se hacer la selección se genera un mensaje de un tipo especifico, este mensaje entra en la función gestora de mensajes y se compara con todos los identificadores de mensajes contenidos en ella. Al encontrar la coincidencia con TXT_DEMO_2, se procesan todas las acciones que este contenga. Existen otros casos que también generan mensajes, como seria el reducir la ventana de la aplicación, la presencia de algún cuadro de mensaje, etc., todos estos mensajes se corresponden con el identificador "WM_PAINT" de la función gestora, esta recibe el mensaje y realiza lo que tenga estipulado en su código. Habrá muchos mensajes que no nos interesen, entonces aún cuando estos se generen no tendrán ninguna respuesta por parte de la aplicación, ya que no existirá algún identificador que los reconozca.

La función anterior es de las mas importantes de toda aplicación Windows, ya que es en ella donde se toman y canalizan todos los eventos que puedan generarse en la aplicación. La mayoría de las llamadas a funciones se hacen desde la función gestora de mensajes, y otras entre las mismas funciones. La función gestora de mensajes puede contener todos los identificadores de mensajes que se requieran, todos ellos del tipo "WM_mensaje".

Hasta aquí solo he explicado el funcionamiento del gestor de mensajes, ahora continuaremos con la explicación de algunas de las funciones que componen a la aplicación, recordando que dichas funciones se basan en las funciones básicas desarrolladas en el capítulo anterior:

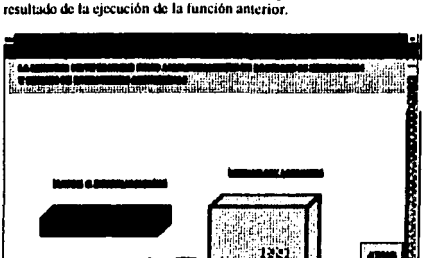
Para ejemplificar el uso de las funciones básicas, no es necesario explicar a todas las funciones que componen a la aplicación completa, por lo tanto solo tomaremos algunas de ellas para su explicación.

El siguiente código corresponde a la función que iniciativa en la aplicación la parte del modulo "memoria" de la opción módulos. La función trata sobre el tipo de datos o información que se almacena en la máquina.

```
void MOD_MEMORIA_1(HWND hwnd,HDC hdc){   // declaración del tipo , nombre y parámetros.

 HBRUSH Brocha, Brocha1;                          // variables para herramientas.
 POINT flecha4[7]; //para crear la flecha          //   arreglos de puntos para dibujar una
flecha.

 flecha4[0].x = 190;  flecha4[0].y = 335;
 flecha4[1].x = 230;  flecha4[1].y = 335;
 flecha4[2].x = 230;  flecha4[2].y = 330;
 flecha4[3].x = 250;  flecha4[3].y = 345;
 flecha4[4].x = 230;  flecha4[4].y = 360;
```

```
 flecha4[5].x = 230;  flecha4[5].y = 355;
 flecha4[6].x = 190;  flecha4[6].y = 355;                    // coordenadas de los puntos de la flecha.

 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 SetCursor(Reloj_Arena);
 Rectangle(hdc,0,0,645,445);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);                                       // esta parte pinta la pantalla del color de la
brocha


CREA_RECTANGULO_BIDIMENSIONAL(hwnd,60,265,240,293,0,127,0,2); // llamada de función
                                                                básica.
 Brocha= CreateSolidBrush(RGB(255,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Polygon(hdc,flecha4,7);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);              // estas líneas dibujas la flecha del color indicado.

 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,320,245,470,360,0,0,127,2); // función básica.
 SetTextColor(hdc,RGB(0,0,0));
 SetBkColor(hdc,RGB(255,255,0));


 TextOut(hdc,350,180,"UNIDAD DE MEMORIA",17);
 TextOut(hdc,70,338,"DIGITO BINARIO",14);
 TextOut(hdc,80,200,"DATOS O INSTRUCCIONES",21);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

VER_CADENA_CON(hwnd,191,191,191,127,0,0,"LA MEMORIA ALMACENA GRUPOS DE
DIGITOS (PALABRA) BINARIOS QUE PUEDEN RE",
            "PRESENTAR INSTRUCCIONES (PROGRAMA) O DATOS QUE LA MAQUINA
EJECUTARA",1,23,10,23,30,70,67);          // llamada de función básica.

TIEMPO(cuenta);                           // función básica.

CADENA_BITS(hdc);                         // función básica.

TIEMPO(cuenta);                           // función básica.

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA MEMORIA SIRVE TAMBIÉN COMO
ALMACENAMIENTO DE RESULTADOS INTERMEDIOS",
            "Y FINALES DE OPERACIONES ARITMETICAS",1,23,10,23,30,70,36);  // función
básica.

BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);            // funciones basicas para dibujar botones.

SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);        // función básica.
var_w=2; j=4;

} //  TERMINA MODULO MEMORIA 1
```

Como podemos observar, el código de la función se compone en gran parte de llamadas a funciones básicas y otra parte de declaraciones de brochas y plumas, así como la utilización de la función TextOut, para la escritura de texto. A simple vista, el código es difícil de entender debido a las llamadas de funciones. Para poder entenderlo tendríamos que hacer todo el seguimiento de estas llamadas, poniendo atención a los parámetros que se le están pasando a las funciones. La siguiente figura muestra el resultado de la ejecución de la función anterior.



A continuación mostraremos la primer función correspondiente al modulo "Microcomputadora". Esta función tiene como objetivo mostrar y explicar los elementos básicos de la microcomputadora.

```
void MICROCOMPUTADORA_1(HWND hwnd){  tipo, nombre y parámetros de la función.

 int c;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;                    //  declaraciones de variables.
HDC hdc;                                //  variable para contexto de visualización
GetClientRect(hwnd, &limite);           //  obtiene el tamaño de la ventana de la aplicación.
hdc=GetDC(hwnd);                        //  declara el contexto de visualización.
SetMapMode(hdc, MM_ANISOTROPIC);        //  establece el modo arbitrario para el tamaño de la
ventana.
SetWindowExt(hdc, 624,442);                     //  valores lógicos de la ventana.
SetViewportExt(hdc, limite.right, limite.bottom);   //  establece los valores lógicos.

 Brocha= CreateSolidBrush(RGB(127,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,0,0,642,444);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);                  //  pinta un rectángulo con los colores de la pluma y de la brocha.

 SetCursor(Reloj_Arena);                //  establece la forma del cursor.
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"EN LA FIGURA SE PRESENTAN LOS ELEMENTOS
BASICOS DE UNA", "MICROCOMPUTADORA ( µC ):",1,15,10,15,30,54,24);  //  función básica.
TIEMPO(cuenta2),                        //  función básica.

CREA_RECTANGULO_BIDIMENSIONAL(hwnd,50,100,150,400,127,0,0,1);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,250,100,550,200,0,255,255,1);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,250,250,350,400,0,64,128,1);
```

```
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,450,250,550,400,0,0,255,1);      //      funciones
básicas.

Brocha= CreateSolidBrush(RGB(0,0,127));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,65,120,135,180);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);              //  pinta un rectángulo con los colores de la pluma y de la brocha.


Brocha= CreateSolidBrush(RGB(0,127,127));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,65,200,135,380);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);          //  pinta un rectángulo con los colores de la pluma y de la brocha.


Brocha= CreateSolidBrush(RGB(0,128,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,270,115,370,185);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);                  //  pinta un rectángulo con los colores de la pluma y de la brocha.

Brocha= CreateSolidBrush(RGB(128,128,64));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,430,115,530,185);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);                  //  pinta un rectángulo con los colores de la pluma y de la brocha.

Brocha= CreateSolidBrush(RGB(127,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,260,260,340,315);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);                   //  pinta un rectángulo con los colores de la pluma y de la brocha.

Brocha= CreateSolidBrush(RGB(127,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,260,335,340,390);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);              //  pinta un rectángulo con los colores de la pluma y de la brocha.

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,460,260,540,315);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);                  //  pinta un rectángulo con los colores de la pluma y de la brocha.

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,460,335,540,390);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);                  //  pinta un rectángulo con los colores de la pluma y de la brocha.


Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,100,185);
LineTo(hdc,100,195);
```

```
MoveTo(hdc,95,190);
LineTo(hdc,100,195);
LineTo(hdc,105,190);

MoveTo(hdc,300,320);
LineTo(hdc,300,330);
MoveTo(hdc,295,325);
LineTo(hdc,300,320);
LineTo(hdc,305,325);

MoveTo(hdc,500,320);
LineTo(hdc,500,330);
MoveTo(hdc,495,325);
LineTo(hdc,500,330);
LineTo(hdc,505,325);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma);          // Estas líneas dibujan otras flechas más delgadas a base de líneas.

Pluma= CreatePen(PS_SOLID,10,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,143,220);
LineTo(hdc,555,220);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma);

Pluma= CreatePen(PS_SOLID,4,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,320,190);
LineTo(hdc,320,210);
MoveTo(hdc,315,195);
LineTo(hdc,320,190);
LineTo(hdc,325,195);
MoveTo(hdc,315,205);
LineTo(hdc,320,210);
LineTo(hdc,325,205);

MoveTo(hdc,485,190);
LineTo(hdc,485,210);
MoveTo(hdc,480,205);
LineTo(hdc,485,210);
LineTo(hdc,490,205);

MoveTo(hdc,300,228);
LineTo(hdc,300,255);
MoveTo(hdc,295,233);
LineTo(hdc,300,228);
LineTo(hdc,305,233);

MoveTo(hdc,505,228);
LineTo(hdc,505,255);
MoveTo(hdc,500,250);
LineTo(hdc,505,255);
LineTo(hdc,510,250);

MoveTo(hdc,160,220);
```

```
 LineTo(hdc,170,210);
 LineTo(hdc,170,230);
 LineTo(hdc,160,220);

 MoveTo(hdc,190,210);
 LineTo(hdc,200,220);
 LineTo(hdc,190,230);
 LineTo(hdc,190,210);

 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);                    //  Estas líneas dibujan otras flechas más delgadas a base de
líneas.

 SetTextColor(hdc,RGB(191,191,191));
 SetBkColor(hdc,RGB(0,0,255));
 TextOut(hdc,335,62,"UNIDAD DE MEMORIA",17);
 TextOut(hdc,20,415,"CPU (ALU Y U.CONTROL)",21);
 TextOut(hdc,266,415,"U.ENTRADA",9);
 TextOut(hdc,470,415,"U.SALIDA",8);          // función de windows para escribir texto.

 Font1= CreateFont(  // para cambiar el texto de tamaño     //  para seleccionar un tipo de letra.
   10,                        /* altura, en pixels */
   0,     /* usar la razón de aspecto para determinar la anchura */
   0,           /*  ángulo de la línea de texto, en .1 grados */
   0,            /*  ángulo de cada carácter cter, en .1 grados */
   FW_BOLD,                   /* ¢ FW_LIGHT ¢ FW_NORMAL */
   FALSE,                  /* TRUE para que sea cursiva */
   TRUE,                 /* TRUE para que est, subrayada */
   FALSE,                  /* TRUE para que est, tachada */
   ANSI_CHARSET,              /* juego de caracteres ANSI */
   OUT_DEFAULT_PRECIS,          /* m,todo de correspondencia */
   CLIP_DEFAULT_PRECIS,            /* m,todo de recorte */
   DRAFT_QUALITY,               /* escalado activado */
   VARIABLE_PITCH | FF_ROMAN,        /* inclinación y familia */
   "Helv");                  /* nombre de la tipografía */
 PrevFont1= SelectObject(hdc,Font1);          /* seleccionar la fuente */
 SetTextColor(hdc,RGB(0,0,0));
 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,67,145,"CRONOMETROS",11);
 TextOut(hdc,80,250,"MICRO",5);
 TextOut(hdc,70,270,"PROCESADOR",10);
 TextOut(hdc,84,290,"( µC )",6);
 TextOut(hdc,310,150,"RAM",3);
 TextOut(hdc,468,150,"ROM",3);
 TextOut(hdc,264,275,"SINCRONIZACION",14);
 TextOut(hdc,274,290,"DE ENTRADA",10);
 TextOut(hdc,464,275,"SINCRONIZACION",14);
 TextOut(hdc,476,290,"DE SALIDA",9);
 TextOut(hdc,274,350,"DISPOSITIVOS",12);
 TextOut(hdc,274,365,"DE ENTRADA",10);
 TextOut(hdc,474,350,"DISPOSITIVOS",12);
 TextOut(hdc,476,365,"DE SALIDA",9);          //  escribe líneas de texto.

 TIEMPO(cuenta4);
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"UNA MICROCOMPUTADORA CONTIENE VARIOS
ELEMENTOS, UNO DE LOS CUALES ES EL",
                    "MICROPROCESADOR,EL CUAL ES LA UNIDAD CENTRAL DE
PROCESAMIENTO (CPU O UCP)",1,13,10,13,30,71,73);  //  función básica.
```

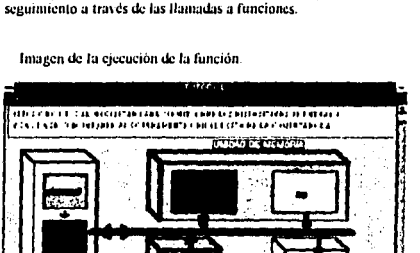
```
 TIEMPO(cuenta4);                    //  función básica.

 for(c=0;c<=3;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(0,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,200,135,380);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,200,135,380);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);               //  Estas líneas dibujan rectángulos.
 TextOut(hdc,80,250,"MICRO",5);
 TextOut(hdc,70,270,"PROCESADOR",10);
 TextOut(hdc,84,290,"( µC )",6);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta4);
 VER_CADENA_CON(hwnd,255,0,0,0,0,0,  "NORMALMENTE EL MICROPROCESADOR ES UNO
O MAS INTEGRADOS 'LSI' QUE CONTIENE",
                                     "TODOS LOS CIRCUITOS DE CONTROL Y ARITMÉTICOS
DE LA MICROCOMPUTADORA",1,13,10,13,30,73,67);
 TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,0,255,0,0,0,0,     "LA UNIDAD DE MEMORIA MUESTRA
DISPOSITIVOS RAM Y ROM, COMUNES DE MUCHAS µC",
                                     ",AUNQUE NO SIEMPRE LOS DOS ESTÁN
PRESENTES",1,13,10,13,30,73,42);
 TIEMPO(cuenta6);
 VER_CADENA_CON(hwnd,0,255,0,0,0,0,  "LA SECCIÓN DE LA RAM CONSTA DE UNO O MAS
INTEGRADOS SEGÚN LA",
                                     "CAPACIDAD DE LA µC ( MICROCOMPUTADORA
)",1,13,10,13,30,60,39);        //  funciones básicas.
 TIEMPO(cuenta4);
 for(c=0;c<=3;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(127,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,270,115,370,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(0,128,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,270,115,370,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TextOut(hdc,310,150,"RAM",3);
 TIEMPO(cuenta5);                    //  estas líneas dibujan rectángulos.
 }
 TIEMPO(cuenta4);                    //  función básica.
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"ESTA SECCIÓN DE LA MEMORIA ALMACENA
PROGRAMAS Y DATOS QUE CAMBIAN CON",
                                     "FRECUENCIA DURANTE EL CURSO DE LA
OPERACION",1,13,10,13,30,69,43);
```

```
TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA SECCIÓN DE LA ROM ALMACENA
INSTRUCCIONES Y DATOS QUE NO VARIAN.", "ESTA MEMORIA NO REQUIERE DE
ENERGÍA ELÉCTRICA PARA ALMACENAR",1,13,10,13,30,66,60);
TIEMPO(cuenta4);                    //  funciones básicas.

for(c=0;c<=3;c++){
MessageBeep(0);
Brocha= CreateSolidBrush(RGB(255,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,430,115,530,185);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TIEMPO(cuenta5);
Brocha= CreateSolidBrush(RGB(128,128,64));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,430,115,530,185);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TextOut(hdc,468,150,"ROM",3);
TIEMPO(cuenta5);                    //  para dibujar rectángulos.
}
TIEMPO(cuenta6);
   VER_CADENA_CON(hwnd,0,255,0,0,0,0,   "LAS SECCIONES DE ENTRADA / SALIDA,
CONTIENEN CIRCUITOS SINCRONIZADORES.",   "",1,13,10,13,30,71,0);  //  funciones basicas.
TIEMPO(cuenta4);
for(c=0;c<=4;c++){
MessageBeep(0);
Pluma= CreatePen(PS_SOLID,2,RGB(0,255,255));
Pluma1= SelectObject(hdc,Pluma);
Brocha= GetStockObject(HOLLOW_BRUSH);
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,250,250,350,400);
Rectangle(hdc,450,250,550,400);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

TIEMPO(cuenta5);
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= GetStockObject(HOLLOW_BRUSH);
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,250,250,350,400);
Rectangle(hdc,450,250,550,400);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);               //  estas líneas dibujan rectángulos en la posición y color indicado.

TIEMPO(cuenta5);
}
TIEMPO(cuenta4);
   VER_CADENA_CON(hwnd,255,255,0,0,0,0,   "ESTOS CIRCUITOS SE NECESITAN PARA
PERMITIR QUE LOS DISPOSITIVOS DE ENTRADA",
                                    "Y SALIDA SE COMUNIQUEN ADECUADAMENTE CON EL
RESTO DE LA COMPUTADORA",1,13,10,13,30,74,67);
```

```
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);                    // funciones básicas.
SetCursor(Cursor_Cruz);             // cambia la forma del cursor.
ULTIMA_IMAGEN(hwnd);                // función básica.
SelectObject(hdc,PrevFont1);
DeleteObject(Font1);

var_w=2; j=7;
ReleaseDC(hwnd,hdc);                // liberación del contexto de visualización.

} // TERMINA FUNCION MICROCOMPUTADORA 1
```

Al igual que la función anterior, esta función presenta un gran número de llamadas a funciones básicas, así como a funciones de windows. Las llamadas a las funciones de windows sirven de apoyo para lograr los efectos buscados, aunque las partes principales se basan en llamadas a funciones básicas. El código a simple vista parece no tener sentido, por lo que para entenderlo se habría que hacer el seguimiento a través de las llamadas a funciones.

Imagen de la ejecución de la función.



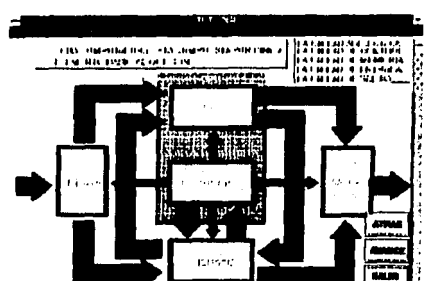
Ahora mostraré el código correspondiente a la primer función del modulo "Función". Esta función nos presenta el esquema básico de una computadora con sus principales módulos.

```
// FUNCION MODULO 1

void MODULO_1(HWND hwnd,HDC hdc){   tipo , nombre y parámetros de la función.

HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;                 // declaración de variables.

SetCursor(Reloj_Arena);             // selecciona forma de cursor.
MÓDULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);7
FLECHAS_CONTROL(hdc,1,5,3);         // funciones basicas.
```

```
Brocha= CreateSolidBrush(RGB(0,0,255));
Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,40,4,430,57);
Rectangle(hdc,440,2,620,82);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);        //  estas lineas pintan  rectángulos con los colores y grueso seleccionados.

SetTextColor(hdc,RGB(191,191,191));
SetBkColor(hdc,RGB(0,0,0));
TextOut(hdc,70,15,"TODA COMPUTADORA ESTA COMPUESTA POR CINCO",41);
TextOut(hdc,70,35, "ELEMENTO BASICOS, QUE SON : ",28);    //  estas lineas escriben texto.
MessageBeep(0);   //  emite un sonido.

TIEMPO(cuenta6);        //  función básica.
SetTextColor(hdc,RGB(255,255,0));
SetBkColor(hdc,RGB(0,0,0));

TextOut(hdc,445,5,"LA UNIDAD ARIT.LOGICA",21);
TIEMPO(cuenta4);  //  función básica.
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,1,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,1,2);
TIEMPO(cuenta);   //  funciones basicas.
}

TIEMPO(cuenta);

TextOut(hdc,445,18,"LA UNIDAD DE CONTROL",20);  //  escribe texto.
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,2,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,2,2);
TIEMPO(cuenta);                  //  funciones basicas.
}

TIEMPO(cuenta);                  //  funciones basicas.

TextOut(hdc,445,33,"LA UNIDAD DE MEMORIA",20);          //  escribe texto.
TIEMPO(cuenta4);     //  función básica.
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,3,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,3,2);
TIEMPO(cuenta);                  //  funciones basicas.
}

TIEMPO(cuenta);              //  funciones basicas.

TextOut(hdc,445,49,"LA UNIDAD DE ENTRADA",20);                  //  escribe texto.
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,4,1);
```

```
TIEMPO(cuenta);
INT_MODULOS(hdc,2,4,2);
TIEMPO(cuenta);                    //   funciones basicas.
}

TIEMPO(cuenta);

TextOut(hdc,445,65,"LA UNIDAD DE SALIDA",19);   //   escribe texto.
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,5,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,5,2);
TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);                   //  funciones basicas
SetCursor(Cursor_Cruz);            //  cambia la forma del cursor.
ULTIMA_IMAGEN(hwnd);               //   función básica.

var_x=2;
} //TERMINA MODULO 1
```

Imagen de la ejecución de la funcion.



Como hemos observado, todas las funciones siguen un mismo formato, manejan funciones basicas junto con funciones de Windows. Pero existen otras funciones que son muy distintas en su código , como es la función que controla las evaluaciones.

Esta función solo utiliza las funciones basicas "TIEMPO" para controlar los intervalos, la función "ULTIMA_IMAGEN", para tener la última pantalla y las funciones "BOTON_EVALUACIÓN" y "BOTON" para crear los botones. Su formato es completamente diferente al de las demás funciones. Su función es controlar todas las sesiones de preguntas. El código para esta función se basa principalmente en decisiones de acuerdo a las respuestas de los usuarios. Su código es el siguiente:

```
void CHECA_EVALUACION(HWND hwnd,int posicion_x,int posición y){
```

```
                                                    // tipo, nombre y parámetros de la
función.
 HBRUSH Brocha, Brocha1;
 HPEN Pluma, Pluma1;                    // declaración de variables.

HDC hdc;                                // variable para contexto de visualización.
GetClientRect(hwnd, &limite);           // obtiene el tamaño de la ventana de la aplicación.
hdc=GetDC(hwnd);                        // establece el contexto de visualización.
SetMapMode(hdc, MM_ANISOTROPIC);        // establece el modo arbitrario de coordenadas lógicas.
SetWindowExt(hdc, 624,442);             // coordenadas lógicas requeridas,
SetViewportExt(hdc, limite.right, limite.bottom); // establece las coordenadas lógicas.
```

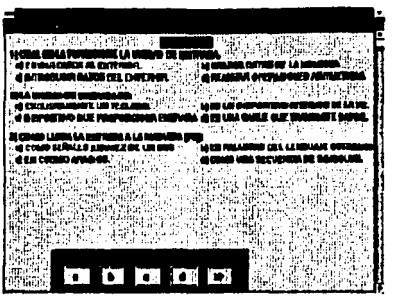
// las siguientes líneas reciben como entrada las coordenadas del ratón en el momento de haber echo la selección, estas coordenadas se comparan con las ubicaciones predefinidas de los botones, y donde se encuentre una coincidencia será la respuesta que el usuario eligió, esta respuesta se compara con la respuesta verdadera de la pregunta, en caso de ser iguales se desplegara el letrero *Bien*, de lo contrario se emitirá un sonido y se desplegara el letrero *Mal*.

```
if(posición y>(factor_alto *395) && posición y<(factor_alto *435)){
 if(posicion_x>(factor_ancho *100) && posicion_x<(factor_ancho *150)){
  BOTONES_EVALUACION(hwnd,2);
  TIEMPO(cuenta5);
  BOTONES_EVALUACION(hwnd,1);
   respuesta='a'; }
 else if(posición x>(factor_ancho *160) && posición x<(factor_ancho *210)){
  BOTONES_EVALUACION(hwnd,4);
  TIEMPO(cuenta5);
  BOTONES_EVALUACION(hwnd,3);
  respuesta='b'; }
  else  if(posición  x>(factor_ancho  *220)  &&  posición  x<(factor_ancho  *270)){
  BOTONES_EVALUACION(hwnd,6);
  BOTONES_EVALUACION(hwnd,6);
  TIEMPO(cuenta5);
  BOTONES_EVALUACION(hwnd,5);
  respuesta='c';}
  else   if(posición x>(factor_ancho *280) && posición x<(factor_ancho *330)){
  BOTONES_EVALUACION(hwnd,8);
  TIEMPO(cuenta5);
  BOTONES_EVALUACION(hwnd,7);
  respuesta='d';}
   // si se selecciona el botón de salir de la evaluación sucede lo siguiente.
  else if(posición x>(factor_ancho *340) && posición x<(factor_ancho *400)){
  BOTONES_EVALUACION(hwnd,10);
  TIEMPO(cuenta5);
  BOTONES_EVALUACION(hwnd,9);
  salir_evaluacion=FALSE;
  switch(indicador){
   case 1: ejecucion_1_1=TRUE; break;
   case 2: ejecucion=TRUE; break;
 }
  evaluacion=FALSE;
  PatBlt(hdc,0,0,645,445,BLACKNESS);
  Brocha= CreateSolidBrush(RGB(127,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,120,120,520,320);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SetTextColor(hdc,RGB(191,191,191));
```

```
SetBkColor(hdc,RGB(0,0,0));
   TextOut(hdc,140,150,"SI ELIGE ATRÁS REGRESARA A LA EVALUACION.",41);
   TextOut(hdc,140,180,"SI ELIGE AVANCE PASARA A LA SIGUIENTE SESION",43);
   TextOut(hdc,140,210,"SI ES QUE TODAVIA HAY.",22);
   TextOut(hdc,140,240,"SI ELIGE SALIR REGRESA AL ULTIMO MENU.",38);
 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
   BOTONES(hwnd,5); }}
  SetBkMode(hdc,TRANSPARENT);
  if(cantidad_preguntas>=numero_pregunta){
if(salir_evaluacion==TRUE){
   if(pregunta[preg]==respuesta){     //   mensaje según la respuesta.
     MessageBeep(0);
     TextOut(hdc,370,375,"BIEN",4);}
     else {MessageBeep(-1);
           MessageBeep(-1);
           MessageBeep(-1);
           MessageBeep(-1);
           MessageBeep(-1);
           MessageBeep(-1);
           TextOut(hdc,370,375,"MAL",3);
           con--;
    preg--;
     numero_pregunta--;
     }

        TIEMPO(cuenta4);
   Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
   Pluma1= SelectObject(hdc,Pluma);
        Brocha= CreateSolidBrush(RGB(0,127,0));
   Brocha1= SelectObject(hdc,Brocha);
   Rectangle(hdc,100,373,408,393);
   SelectObject(hdc,Brocha1);
   DeleteObject(Brocha);
        SelectObject(hdc,Pluma1);
   DeleteObject(Pluma1);

     if(cantidad_preguntas-1>=numero_pregunta){     //   controla el número de pregunta a
contestar.
    switch(con){
    case 0:
        TextOut(hdc,170,375,"RESPONDA PREGUNTA 1",19);
    break;
    case 1:
        TextOut(hdc,170,375,"RESPONDA PREGUNTA 2",19);
    break;
    case 2:
        TextOut(hdc,170,375,"RESPONDA PREGUNTA 3",19);
    break;
    case 3:
        TextOut(hdc,170,375,"RESPONDA PREGUNTA 4",19);
    break;
    case 4:
        TextOut(hdc,170,375,"RESPONDA PREGUNTA 5",19);
    break;
    } } else TextOut(hdc,170,375,"FIN DE LA EVALUACION.",21);
    con++;
    preg++;
```

```
            numero_pregunta++;} }

salir_evaluacion=TRUE;
ULTIMA_IMAGEN(hwnd);                    // repaldo de la última pantalla (función básica).
ReleaseDC(hwnd,hdc);                    // libera el contexto de visualización
}
```

Ejemplo visual de la ejecución de este código. (Nota: esta función no es la encargada del dibujo de los botones, solamente los controla).



Como hemos visto, la mayoría de las funciones contienen un código apoyado principalmente en funciones básicas. Las funciones anteriores, solo son una parte representativa de la aplicación completa. Solamente tienen como objeto mostrar el uso de las funciones básicas. En el capítulo siguiente se explica más detalladamente el objetivo de cada módulo que conforma a la aplicación completa.

Capítulo IV.- FUNCIONAMIENTO DEL TUTORIAL DESARROLLADO.

IV.1.- Función de los módulos que componen al tutorial.

En este capítulo se hablara más detalladamente de la aplicación de ejemplo desarrollada a través de las funciones básicas. Como se menciono anteriormente, la aplicación desarrollada es más que un simple ejemplo, por lo tanto se le a dedicado este capítulo en el cual se explicara la función de cada módulo que compone a la misma.

La aplicación aborda el tema de la *organización y funcionamiento de los diferentes módulos que componen a la computadora*, cabe señalar que el tutorial no pretende ser un material especializado del tema, por lo tanto solo se manejaran los conceptos básicos suficientes para que el usuario comprenda los temas tratados.

Para lograr un mejor resultado, el tutorial esta dividido en tres módulos principales:

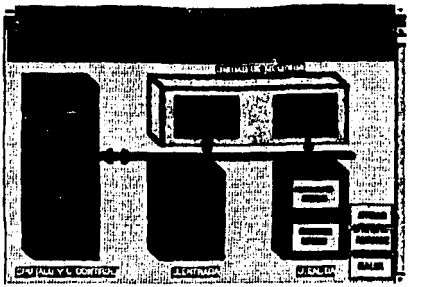
- *MICROCOMPUTADORA*
- *MÓDULOS.*
  - *Unidad de entrada.*
  - *Unidad de salida.*
  - *Unidad de memoria.*
  - *Unidad aritmético / lógica.*
  - *Unidad de control*
  - *CPU*
- *FUNCION.*

Los tres módulos anteriores son el esqueleto principal de la aplicación. Cada módulo y submódulo están compuestos por una serie de sesiones las cuales van explicando los temas correspondientes al módulo. Al final de cada sesión, se presenta una evaluación correspondiente al tema que se acaba de ver, esta evaluación no califica, ya que la calificación obtenida en estas queda al criterio del usuario. Para complementar al tutorial, se integro la sección de apoyo: *Conceptos básicos*, esta opción contiene información acerca de la terminología utilizada en el tutorial, así como la explicación de algunos puntos importantes.

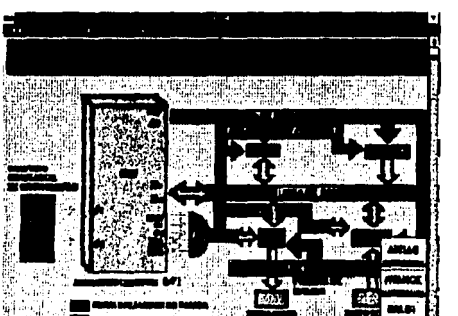
El primer módulo denominado *MICROCOMPUTADORA*, nos habla sobre una computadora digital de aplicación general, compuesta por elementos LSI (integración a gran escala) construidos alrededor de una unidad central de procesamiento (UCP o CPU). La UCP (microprocesador) está controlada por programas, tiene funciones aritméticas y lógicas y un bus en paralelo de aplicación general de entrada/salida (línea). La UCP está contenida en uno o más pequeños chips procesadores y se comunica con RAM (memoria de acceso aleatorio, memoria de lectura/escritura), ROM (memoria de solo lectura); y memorias auxiliares para almacenamiento de instrucciones y datos. Varios dispositivos de entrada y salida (teclado, TRC, unidades de disco, etc.) están conectados para permitir la comunicación con el medio ambiente externo. La computadora tiene su propio suministro de energía.

Este módulo esta compuesto por dos sesiones y cuatro evaluaciones. Ambas sesiones son muy semejantes entre sí, la primera nos presenta una imagen simple con los componentes de una computadora, explicando como interactuan estos entre si , mientras que la segunda vuelve a presentar una imagen semejante a la anterior pero mas detallada, explicando puntos que no están visibles en un esquema básico.

La figura tomada al final de la primera sesión es la siguiente:



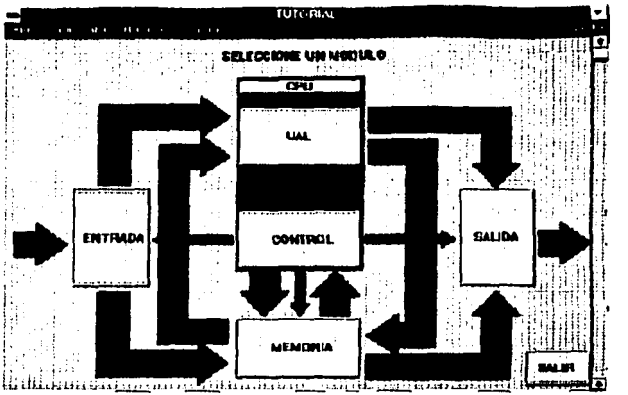
Ninguna de las evaluaciones del turorial se encuentan impresas en este libro, por lo tanto tendra que ejecutar el tutorial para poder verlas. La figura correspondiente al final de la segunda sesión es la siguiente:



Como observamos ambas sesiones son muy semejantes , solo que esta última es mas especializada.

Ahora presentara al segundo módulo denominado *MÓDULOS*. Esta parte nos presenta el esquema básico de un sistema de computación y va explicando cada componente por separado. Su objetivo es dar al usuario una idea del funcionamiento de cada una de las partes del esquema básico de una computadora.

Al diseñar este módulo, se buscaba que el usuario interactuara mas con la máquina, por lo tanto la pantalla para este módulo es a base de botones, donde cada botón corresponde a un componente de la computadora. La figura representativa de esto es la siguiente:

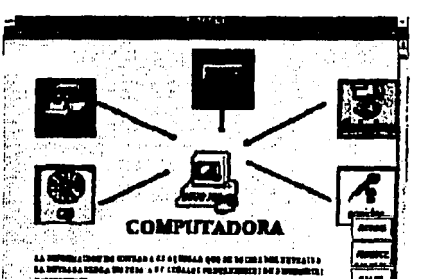


Los componentes para esta figura con los siguientes:

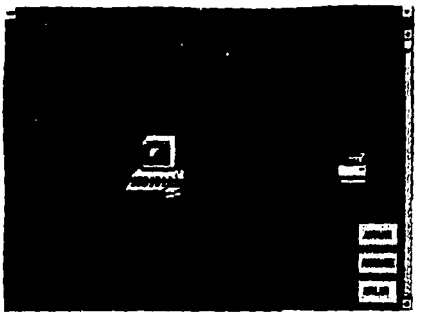
• *Unidad de entrada.*	Función de la unidad de entrada.
• *Unidad de salida*	Función de la unidad de salida.
• *Unidad de memoria.*	Función de la unidad de memoria.
• *Unidad aritmético / lógica.*	Función de ALU (unidad aritmético / lógica).
• *Unidad de control*	Función de la unidad de control.
• *CPU O UCP.*	Componentes y función de la unidad central de procesamiento

El botón correspondiente a la *Unidad de entrada* esta compuesto por una sesión y una evaluación. Esta unidad consta de todos los dispositivos que se usan para tomar información y datos externos a la computadora y colocarlos en la unidad de memoria o la ALU (Unidad Aritmético / Lógica.).

Imagen representativa de la ejecución:

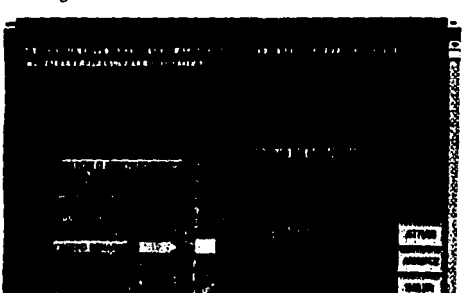


La parte correspondiente a la *Unidad de salida*, también esta formada por una sola sesión y una evaluación. Esta unidad consta de los dispositivos que se usan para transferir datos e información de la computadora al exterior. La siguiente figura fue tomada en tiempo de ejecución:

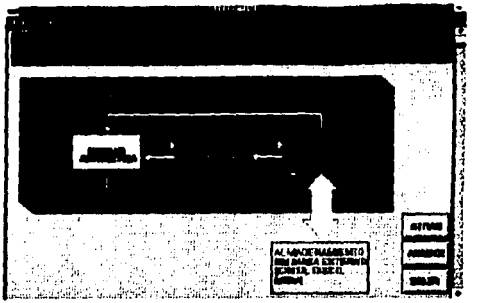


La *unidad de memoria*, esta compuesta por tres sesiones y cuatro evaluaciones. La memoria es un dispositivo que almacena grupos de dígitos binarios (palabra).

La primer sesión trata sobre el tipo de datos o información que maneja la unidad de memoria. Su figura representativa es la siguiente.

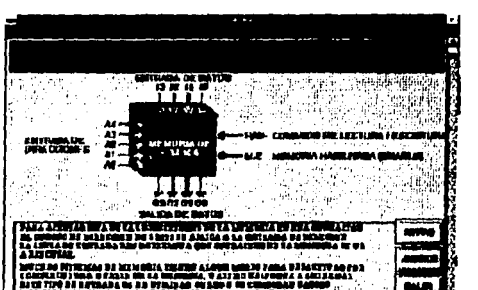


En la segunda sesión encontramos los diferentes tipos de memorias, mostrando a las que normalmente encontramos en un equipo de PC. Su imagen es la siguiente:



En la tercer sesión encontramos información relativa a los tamaños de memoria, líneas que la componen y la manera como se almacenan las palabras (datos o información) en la misma.

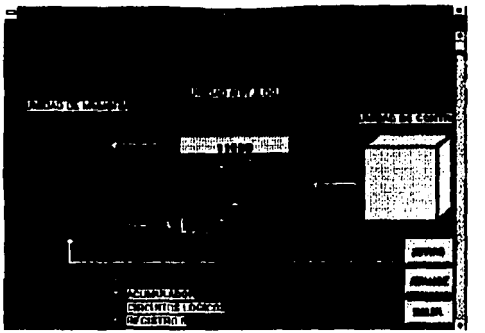
Su figura es la siguiente:



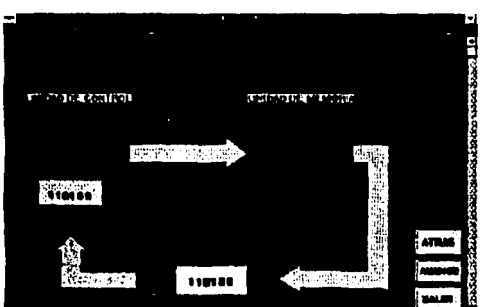
Estas tres sesiones que acabamos de ver son las que componen a la unidad de memoria. Las primeras dos sesiones van acompañadas de una evaluación respectivamente, y la última sesión contiene a dos evaluaciones.

A continuación veremos lo correspondiente a la *Unidad aritmética - lógica* (ALU), que es donde se realizan las operaciones aritméticas y lógicas sobre los datos. La UAL contiene acumuladores, sumadores, circuitos comparadores, registros para conservar operandos y resultados, y circuitos de corrimiento y secuencia para realizar multiplicación, división y otras operaciones matemáticas. La UAL ejecuta la mayoría de los comandos de procesamiento interno de una computadora. Dentro de la sesión se incluye un ejemplo que muestra el funcionamiento de esta unidad. Esta parte esta compuesta por una sesión y dos evaluaciones.

La siguiente figura es representativa de esta sesión:



Ahora presentare la sesión correspondiente al módulo *Unidad de control*. Esta unidad es la encargada de elegir la siguiente instrucción, interpreta su dirección (localización) y partes operativas , y ejecuta la instrucción aplicando las señales adecuadas a la ALU y a otras partes del sistema de computación de acuerdo con su interpretación. La información referente a esta unidad cuenta con una sesión y una evaluación solamente. La siguiente figura es representativa de esta sesión:

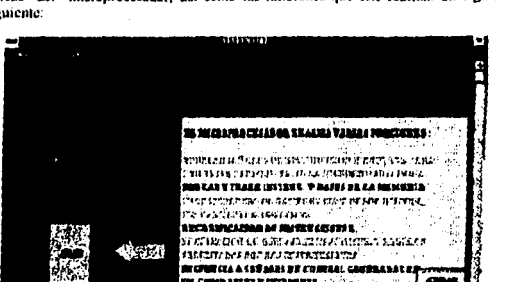


Ahora hablaremos del CPU o UCP (unidad central de procesamiento). El CPU esta compuesto por la ALU (unidad aritmético / lógica) y por la unidad de control. La UCP:

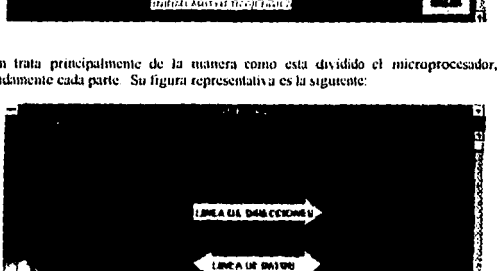
- sigue la posición de las instrucciones que se estén ejecutando.
- recupera de la memoria las instrucciones y las interpreta o descodifica.
- ejecuta las instrucciones utilizando las unidades de hardware de aritmética, de lógica, de prueba y de corrimiento.
- dirige el hardware de entrada/salida cuando el programa requiere instrucciones de E/S.

La UCP contiene registros (contadores) de alta velocidad para cálculo y modificación de dirección, así como circuitos de detección de errores para encontrarlos. La computadora puede ejecutar en forma concurrente tantos programas como UCP's haya disponibles para esa computadora

Este módulo esta dividido en dos partes y en dos evaluaciones. La primer parte trata sobre las principales características del microprocesador, así como las funciones que este realiza. Su figura representativa es la siguiente:



La segunda sesión trata principalmente de la manera como esta dividido el microprocesador, explicando mas detalladamente cada parte. Su figura representativa es la siguiente:



Hasta aquí comprende lo correspondiente al botón *MÓDULOS* de la pantalla principal.

Por último tenemos al botón denominado *FUNCION*, en este módulo trataremos la información referente

al flujo de los datos, información y señales de control que se da entre los elementos básicos de una computadora , así como la relación que existe entre ellos mismos. Este módulo esta constituido por diez sesiones y seis evaluaciones. Todas las sesiones se desarrollan sobre una misma pantalla, en la cual se presentan los elementos básicos de una computadora, las flechas de flujos de datos e información y las flechas de flujo de señales de control. Esto se representa en la siguiente figura que corresponde a la segunda sesión:

A continuación explicare brevemente la función de cada sesión:

sesión 1: Se muestran los elementos básicos de una computadora.
sesión 2: Indican los componentes del CPU y señalan los flujos que existentes en la computadora.
sesión 3: Se explica la función de la ALU (unidad aritmético /lógica).
sesión 4: Se da información acerca de la unidad de memoria.
sesión 5: Presenta información acerca de la unidad de control.
sesión 6: Da mas información de la unida de memoria.
sesión 7: Nos muestra la función de la unidad de entrada.
sesión 8: Presenta información de la unidad de salida.
sesión 9: Nos da información acerca de la sincronización en las unidades de entrada / salida.
sesión 10: Da información acerca de la ejecución de las instrucciones.

La aplicación tutorial como se acaba de ver, esta dividida en tres módulos principales, pero además se cuenta en la barra de menú con otras opciones de apoyo como son los *conceptos básicos* y la *ayuda*. Estas opciones contienen información de conceptos básicos, abreviaturas utilizadas y de algunas palabras técnicas.

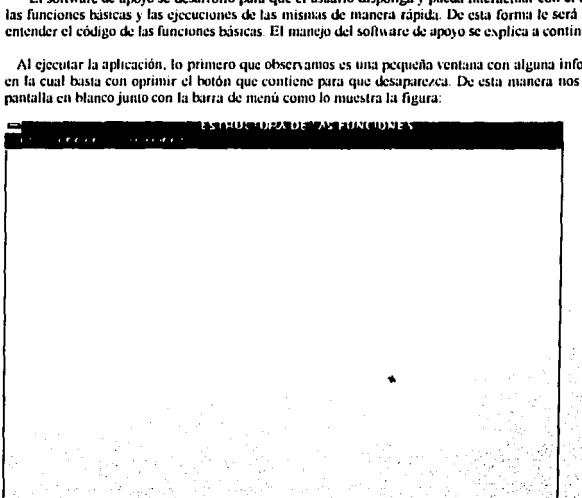
Capítulo V.- FUNCIONAMIENTO DEL SOFTWARE DE APOYO.

V.1.- Función del programa de apoyo.

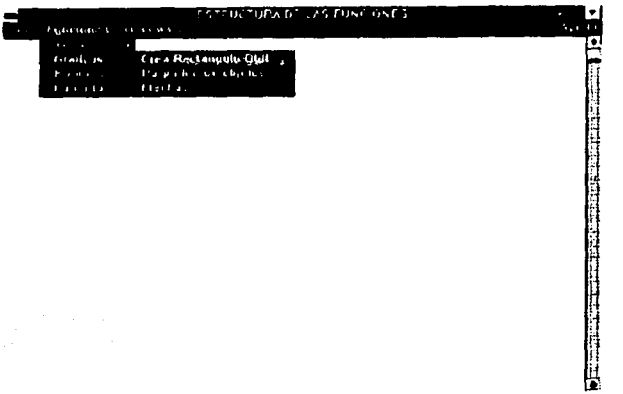
La aplicación tutorial, cuenta con otro pequeño programa que se denomina *software de apoyo*, este programa no esta dentro del tutorial, sino que es un software independiente.

El software de apoyo se desarrollo para que el usuario disponga y pueda interactuar con el código de las funciones básicas y las ejecuciones de las mismas de manera rápida. De esta forma le será más fácil entender el código de las funciones básicas. El manejo del software de apoyo se explica a continuación.

Al ejecutar la aplicación, lo primero que observamos es una pequeña ventana con alguna información, en la cual basta con oprimir el botón que contiene para que desaparezca. De esta manera nos queda la pantalla en blanco junto con la barra de menú como lo muestra la figura:

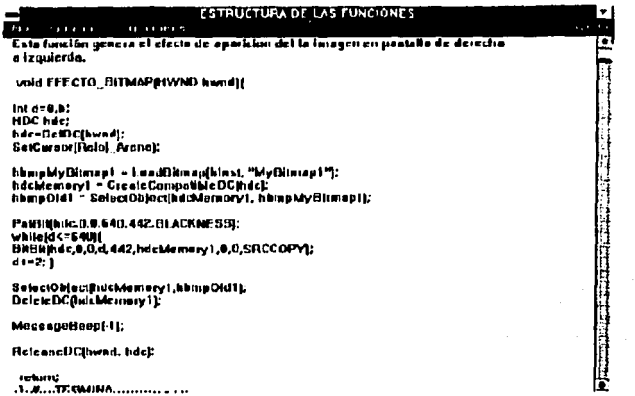


En el menú podemos observar del lado izquierdo a tres opciones, la primer opción (file), lo único que hace es terminar con la ejecución del software. La segunda opción (funciones) contiene a las principales funciones básicas que se desarrollaron en el capítulo dos, y cada una de estás opciones que se despliegan a su vez contiene otras opciones, tal como lo indica la figura:



Cabe mencionar que mientras no se elija alguna de estas funciones, la tercer opción del menú (opciones) permanecerá inactiva, y de igual forma al elegirse alguna de las funciones básicas, la segunda opción del menú quedara inactiva permanentemente hasta que no se borre la función que se esta analizando actualmente mediante la comando *limpiar*.

Una vez que se a elegido alguna función básica, a continuación de desplegara su código en la pantalla, con la posibilidad de poderlo ejecutar a través de la opción *ejecuta* del menú "opciones", de esta manera se puede estar interactuando entre el código de alguna de las funciones y su correspondiente ejecución. Un ejemplo de este procedimiento es el siguiente: Primero elegimos la función *efecto bitmap_1* de la opcion "Pantalla" de la segunda opción del menú, con lo cual obtenemos el código de dicha función:



Del lado derecho podemos observar a la barra de desplazamiento, con la cual podemos recorrernos a lo largo de todo el código. Una vez que tenemos el código podemos ejecutarlo con solo elegir la opción *ejecutar* del menú "opciones"

Del lado derecho de la barra de menu, podemos ver la opción "Ayuda", esta opción contiene información referente a las funciones utilizadas en los programas, explicando brevemente su función y parámetros requeridos

Básicamente esta es la estructura general del software de apoyo, su función como se menciono anteriormente, solamente es apoyar al usuario a comprender mejor el manejo y desarrollo de funciones en "C" bajo Windows.

Básicamente esta es la estructura general del software de apoyo, su función como se menciono anteriormente, solamente es apoyar al usuario a comprender mejor el manejo y desarrollo de funciones en "C" bajo Windows.

A CONTINUACIÒN SE PRESENTA EL LISTADO COMPLETO DE LA APLICACIÒN.

```
//-------------------------------------------------------------------------------
---
//-------------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "FUNCION.H"
#include "VARIOS.H"
#include "CONTROL.H"
#include "PREGUNTA.H"
#include "MODULOS.H"
#include "MICRO.H"
#include "TESIS.H"
#include "PREG_FUN.H"
#include <STRING.H>
#include <stdio.h>

#define SCROLLLINES 233 //NUMERO MAXIMO DE RENGLONES

#define cuenta 40
#define cuenta1 5
#define cuenta2 70
#define cuenta3 10
#define cuenta4 100
#define cuenta5 20
#define cuenta6 200
#define cuenta7 150

static void COORDENADAS(HWND ,int,int );

HANDLE hAccel;
HANDLE hInst;
HWND Gestor_vent;
HCURSOR Reloj_Arena;
HCURSOR Cursor_Flecha;
HCURSOR Cursor_Cruz;
HICON Icono_Tesis;
int indicador; //   se utiliza para saber que variable SE NIEGA CUANDO ENTRAN LAS
EVALUACIONES
int variable; // indica que variable se va a negar cuando entra la ayuda
int cantidad_preguntas;
int numero_pregunta;
int con;  // se utiliza en CHECA_EVALUACION para que pida la respuesta de la sig. pregunta
int preg;
int Max= 50;
int var_z=0,var_w=0,j=0;
int mensaje;// toma el valor devuelto por messagebox
int var_mouse;
int posicion_x,posicion_y;
static int charwt,charht,xClientView,yClientView,posvertscroll;
char cadena_textos[250];
char pregunta[5],respuesta;
char Tex_Boton[10];
```

```
int MaxText= 249;
int p,r,q;
RECT limite;
HFONT Font,PrevFont,Fuente,PreFuente;  /* gestor de la fuente  */
float factor_ancho;
float factor_alto;
char cadena_titulos[51];
 BOOL salir_evaluacion=FALSE;
 BOOL evaluacion=FALSE;
 BOOL ejecucion=FALSE;
 BOOL imagen =FALSE;
 BOOL ejecucion_1=FALSE;
 BOOL ejecucion_1_1=FALSE;
 BOOL seleccion=FALSE;
 BOOL boton_selec=FALSE;
 BOOL micro=FALSE;
 BOOL ayuda=FALSE;
 BOOL ultima_imagen=TRUE;
 BOOL puntos=TRUE;

HDC  hdcMemory,hdcCuadro,hdcTiempo,hdcSeleccion,hdcU_Ent_Sal,hdcMicro,hdcUltima_Imagen;  //
contextos de visualizacion de pag. oculta
HBITMAP hbmpMyBitmap,hbmpOld,hmbCuadro,hmbCuadro1,
            hmbTiempo,hmbTiempo1,hbmpSeleccion,hbmpSeleccion1,
            hbmpU_Ent_Sal,hbmpU_Ent_Sal1,
            hbmpMicro,hbmpMicro1,
        hbmpUltima_Imagen,hbmpUltima_Imagen1;

//*****************************************************************************
****
// FUNCION PRINCIPAL

      int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                  LPSTR lpCmdLine, int nCmdShow)
{
 MSG msg;
 HWND hWndPrev;

// COMPRUEBA UNA SOLA INSTANCIA

hWndPrev = FindWindow("TESIS", NULL);       /* ¯ Otras instancias? */
if (hWndPrev != NULL)                       /* si devuelve un gestor... */
 {      /* entonces es que ya se est  ejecutando otra instancia... */
 BringWindowToTop(hWndPrev);    /* convertirla en la ventana activa */
 return 0;                          /* y cancelar esta instancia */
 }

// PARA CREAR LA VENTANA PRINCIPAL (llamadas a funciones)

hInst = hInstance;
if (!inicializa_clase(hInstance)) return FALSE;
Gestor_vent = inicializa_ventana(hInstance);
if (!Gestor_vent) return FALSE;
UpdateWindow(Gestor_vent);
hAccel = LoadAccelerators(hInstance,"MENUS");
ShowWindow(Gestor_vent, SW_SHOWMAXIMIZED);
imagen=TRUE;
SELECCION(Gestor_vent);
```

```
// PARA CHECAR LA  PRESENCIA DEL MOUSE
var_mouse = GetSystemMetrics(SM_MOUSEPRESENT);
if (!var_mouse)
  {
  MessageBox(GetFocus(),"EL MOUSE NO ESTA PRESENTE","ERROR",MB_OK);
  }

// BUCLE DE MENSAJES (verifica si existen mensajes)
while (GetMessage(&msg,0,0,0))
  {
  if(TranslateAccelerator(Gestor_vent, hAccel, &msg))
    continue;
  TranslateMessage(&msg);
  DispatchMessage(&msg);
  }

return(msg.wParam);
}

//*****************************************************************************
****
// TERMINA FUNCION PRINCIPAL

// GESTOR DE MENSAJES(distribuidor de mensajes)
LONG FAR PASCAL zWndProc(HWND hWnd, unsigned message,
                WORD wParam, LONG lParam)
{

    PAINTSTRUCT ps;
    HDC hdc;

    TEXTMETRIC tm;
    GetClientRect(hWnd, &limite);
    factor_ancho=(float)limite.right/624;
    factor_alto=(float)limite.bottom/442;

switch (message)
  {
  case WM_COMMAND:
    switch(wParam)
      {
      case TXT_SALIR:
      if(seleccion==TRUE){
        mensaje=MessageBox(GetFocus(),"SALIR DE LA APLICACION","ESCAPE",MB_OKCANCEL |
MB_ICONSTOP);
      if(mensaje == IDOK){
       SelectObject(hdcMemory,hbmpOld);   /* ...deseleccionarlo */
       SelectObject(hdcCuadro,hmbCuadro1);
       SelectObject(hdcSeleccion,hbmpSeleccion1);
       SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen1);

       DeleteObject(hbmpUltima_Imagen);
       DeleteObject(hmbCuadro);  /* ...borrar los mapas de bits ocultos... */
       DeleteDC(hdcMemory);          /* ...borrar el DC en memoria... */
       DeleteDC(hdcCuadro);
       DeleteDC(hdcSeleccion);
       SelectObject(hdc,PreFuente),
```

```
    PostQuitMessage(0);}}

    if(ayuda==TRUE){
                    mensaje=    MessageBox(GetFocus(),"DESEA    SALIR    DE    LA
AYUDA","AYUDA",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
          ayuda=FALSE;
          switch(variable){
       case 0:
        seleccion=TRUE; break;
       case 1:
            ejecucion=TRUE; break;
           case 2:
            ejecucion_1=TRUE; break;
           case 3:
           ejecucion_1_1=TRUE; break;
           case 4:
            evaluacion=TRUE; break;
            }
            hdc=GetDC(hWnd);

BitBlt(hdc,0,0,limite.right,limite.bottom,hdcUltima_Imagen,0,0,SRCCOPY);//StretchBlt(hdc,0,0,limite.
right,limite.bottom,hdcUltima_Imagen,0,0,limite.right,limite.bottom,SRCCOPY);
            ReleaseDC(hWnd,hdc);}}

     break;
    case TXT_CONCEPTOS_BASICOS:
         mensaje=MessageBox(GetFocus(),"CONCEPTOS_BASICOS","BASICOS",MB_OKCANCEL |
MB_ICONINFORMATION);
     if(mensaje == IDOK){

     if(seleccion==TRUE){variable=0; seleccion=FALSE;}
          if(ejecucion==TRUE){variable=1; ejecucion=FALSE;}
           if(ejecucion_1==TRUE){variable=2; ejecucion_1=FALSE;}
            if(ejecucion_1_1==TRUE){variable=3; ejecucion_1_1=FALSE;}
             if(evaluacion==TRUE){variable=4; evaluacion=FALSE;}

    ayuda=TRUE;
     AYUDA(hWnd);}
     break;
     case TXT_DEMO_1:
          mensaje= MessageBox(GetFocus(),"ESQUEMA  GENERAL","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
     if(mensaje==IDOK){
      if(seleccion==TRUE){seleccion=FALSE;}
          if(ejecucion==TRUE){ejecucion=FALSE;}
           if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
            if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
             if(evaluacion==TRUE){evaluacion=FALSE;}
             ejecucion=TRUE;
             var_z=1;       // PARA INICIALIZAR EL PRIMER MODULO
             APLICACION(hWnd,var_z);} break;
     case TXT_DEMO_2:
          mensaje= MessageBox(GetFocus(),"FUNCION POR MODULO","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
     if(mensaje==IDOK){
      if(seleccion==TRUE){seleccion=FALSE;}
          if(ejecucion==TRUE){ejecucion=FALSE;}
```

```
          if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
           if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
            if(evaluacion==TRUE){evaluacion=FALSE;}
        ejecucion_1=TRUE;
           APLICACION_1(hWnd);} break;
    case TXT_DEMO_3:
       mensaje    = MessageBox(GetFocus(),"MICROCOMPUTADORA","DEMO",MB_OKCANCEL |
MB_ICONINFORMATION);
     if(mensaje==IDOK){
      if(seleccion==TRUE){seleccion=FALSE;}
          if(ejecucion==TRUE){ejecucion=FALSE;}
           if(ejecucion_1==TRUE){ejecucion_1=FALSE;}
            if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE;}
             if(evaluacion==TRUE){evaluacion=FALSE;}
            ejecucion_1_1=TRUE;
        micro=TRUE;
           MICROCOMPUTADORA_1(hWnd);} break;
    case TXT_AYUDA_PRINCIPAL:
          LoadString(hInst,TEX_AYUDA,cadena_titulos,Max);
          LoadString(hInst,TEX_AYUDATEX,cadena_textos,MaxText);
          mensaje=MessageBox(GetFocus(),cadena_textos,cadena_titulos,MB_OKCANCEL |
MB_ICONINFORMATION);
     if(mensaje == IDOK){
      if(seleccion==TRUE){variable=0; seleccion=FALSE;}
          if(ejecucion==TRUE){variable=1; ejecucion=FALSE;}
           if(ejecucion_1==TRUE){variable=2; ejecucion_1=FALSE;}
            if(ejecucion_1_1==TRUE){variable=3; ejecucion_1_1=FALSE;}
             if(evaluacion==TRUE){variable=4; evaluacion=FALSE;}

          ayuda=TRUE;
       AYUDA(hWnd);}
          break;
     default:
       return(DefWindowProc(hWnd, message, wParam, lParam));
   }
   break;

   case WM_INITMENUPOPUP:
          if(seleccion==TRUE || ayuda=='TRUE){
            EnableMenuItem(wParam,TXT_SALIR,MF_ENABLED);}
            else EnableMenuItem(wParam,TXT_SALIR,MF_GRAYED);
          if(ayuda==TRUE){
            EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_GRAYED);
            EnableMenuItem(wParam,TXT_DEMO_1,MF_GRAYED);
            EnableMenuItem(wParam,TXT_DEMO_2,MF_GRAYED);
            EnableMenuItem(wParam,TXT_DEMO_3,MF_GRAYED);
            EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_GRAYED);}
         else {
            EnableMenuItem(wParam,TXT_CONCEPTOS_BASICOS,MF_ENABLED);
            EnableMenuItem(wParam,TXT_DEMO_1,MF_ENABLED);
            EnableMenuItem(wParam,TXT_DEMO_2,MF_ENABLED);
            EnableMenuItem(wParam,TXT_DEMO_3,MF_ENABLED);
            EnableMenuItem(wParam,TXT_AYUDA_PRINCIPAL,MF_ENABLED);}
            break;

   case WM_LBUTTONDOWN:
          posicion_x= LOWORD(lParam);
          posicion_y= HIWORD(lParam);
```

```
        if(seleccion==TRUE){CHECA_SELEC(hWnd,posicion_x,posicion_y);}
            if(ejecucion==TRUE){CHECA_APLI(hWnd,posicion_x,posicion_y);}
              if(ejecucion_1==TRUE){CHECA_APLI_1(hWnd,posicion_x,posicion_y);}
               if(ejecucion_1_1==TRUE){CHECA_MOD(hWnd,posicion_x,posicion_y);}
                 if(evaluacion==TRUE){CHECA_EVALUACION(hWnd,posicion_x,posicion_y);}
      break;

//  prueba de barras

      case WM_CREATE:
       hdc=GetDC(hWnd);
       GetTextMetrics(hdc,&tm);
       charwt=tm.tmAveCharWidth;
       charht=tm.tmHeight+tm.tmExternalLeading;
       ReleaseDC(hWnd,hdc);
      SetScrollPos(hWnd,SB_VERT,posvertscroll,TRUE);
      SetScrollRange(hWnd,SB_VERT,0,SCROLLLINES,FALSE);
       break;

      case WM_VSCROLL:
       if(ayuda==TRUE){
       switch (wParam)
       {
        case SB_LINEDOWN:
          posvertscroll+=1;
          break;
        case SB_LINEUP:
          posvertscroll-=1;
          break;
        case SB_PAGEDOWN:
          posvertscroll+=yClientView/charht;
          break;
        case SB_PAGEUP:
          posvertscroll-=yClientView/charht;
        case SB_THUMBPOSITION:
          posvertscroll=LOWORD(lParam);
          break;
        default:
          break;
       }
       if (posvertscroll>SCROLLLINES)
        posvertscroll=SCROLLLINES;
       if (posvertscroll<0)
        posvertscroll=0;
       if (posvertscroll!=GetScrollPos(hWnd,SB_VERT))
       {
        SetScrollPos(hWnd,SB_VERT,posvertscroll,TRUE);
        InvalidateRect(hWnd,NULL,TRUE);
       }
       } //fin del if
      break;
//  fin prueba de barras iiiiiiiiiiiiiiiiii

     case WM_PAINT:

           hdc=BeginPaint(hWnd,&ps);
//       SetMapMode(hdc, MM_ANISOTROPIC);
```

```
//      SetWindowExt(hdc, 624,442);
//      SetViewportExt(hdc, limite.right, limite.bottom);
          if(ayuda==TRUE){
          LECTOR_AYUDA(hWnd);
          }
          else
{BitBlt(hdc,0,0,limite.right,limite.bottom,hdcUltima_Imagen,0,0,SRCCOPY);}//StretchBlt(hdc,0,0,640,
442,hdcUltima_Imagen,0,0,640,442,SRCCOPY);}
          EndPaint(hWnd,&ps);
          break;

 case WM_MOUSEMOVE:
          posicion_x= LOWORD(lParam);
          posicion_y= HIWORD(lParam);
          COORDENADAS(hWnd,posicion_x,posicion_y);
    break;

 case WM_DESTROY:        /* si la ventana est  siendo destruida... */

    SelectObject(hdcMemory,hbmpOld);   /* ...deseleccionarlo */
    SelectObject(hdcCuadro,hmbCuadro1);
    SelectObject(hdcSeleccion,hbmpSeleccion1);
    SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen1);

    DeleteObject(hbmpUltima_Imagen);
    DeleteObject(hmbCuadro);  /* ...borrar los mapas de bits ocultos... */
    DeleteDC(hdcMemory);          /* ...borrar el DC en memoria... */
    DeleteDC(hdcCuadro);
    DeleteDC(hdcSeleccion);
    KillTimer(hWnd,1);
    SelectObject(hdc,PreFuente);
    PostQuitMessage(0);                    /* limpiar y terminar */

  break;

  case WM_SYSCOMMAND:              /* si es una orden del sistema... */

  if ((wParam & 0xfff0)== SC_MOVE)
    {         /* bloquea cualquier intento de mover la ventana... */
    break; }

  if ((wParam & 0xfff0)== SC_SIZE)
    { /* bloquea cualquier intento de cambiar de tamaño la ventana */
    break;}

 default:  /* en caso contrario, pasa mensaje al gestor por defecto */
  return(DefWindowProc(hWnd, message, wParam, lParam));
 }
return FALSE;
}

//*****************************************************************************
****
// ESTA FUNCION INICIALIZA LAS CARACTERISTICAS QUE TENDRA LA VENTANA
// UTILIZA LA CLASE WndClass QUE SIRVE PARA DICHO PROPOSITO

BOOL inicializa_clase(HANDLE hInstance)
{
```

```
WNDCLASS WndClass;
WndClass.style= 0;
WndClass.lpfnWndProc= &WndProc;
WndClass.cbClsExtra= 0;
WndClass.cbWndExtra= 0;
WndClass.hInstance= hInstance;
WndClass.hIcon= LoadIcon(NULL,Icono_Tesis);
WndClass.hCursor= LoadCursor(NULL,IDC_CROSS);
WndClass.hbrBackground= CreateSolidBrush(RGB(255,255,255));
WndClass.lpszMenuName= "TESIS";
WndClass.lpszClassName= "TESIS";
return RegisterClass(&WndClass);
}


// CREACION DE LA VENTANA PRINCIPAL

HWND inicializa_ventana(HANDLE hInstance)
{
 HWND hWnd;
LoadString(hInstance,TEX_TITULO,cadena_titulos,Max);
Reloj_Arena= LoadCursor(NULL,IDC_WAIT);
Cursor_Flecha=LoadCursor(NULL,IDC_ARROW);
Cursor_Cruz=LoadCursor(NULL,IDC_CROSS);
SetCursor(Reloj_Arena);
hWnd = CreateWindow(
 "TESIS",
 cadena_titulos,
     WS_OVERLAPPED  |     WS_SYSMENU  |  WS_CLIPCHILDREN  |WS_THICKFRAME  |
WS_MINIMIZEBOX ,
 -4,-4,
 1300,1300,
 NULL,NULL,
 hInstance, (LPSTR)NULL);

return hWnd;
}

// ABRE Y LEE ARCHIVOS

 void LECTOR_AYUDA(HWND hwnd){

 FILE *fp;
 int i,t,ch,sLength;
 static char szBuffer[150]=" ";

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 640,442);
SetViewportExt(hdc, limite.right, limite.bottom);

   t=0;
   if ((fp=fopen("AYUDA.DOC","r"))!=NULL)
   {
    while(!feof(fp))
    {
```

```
        ch=fgetc(fp);
            i=0;
        while((ch!='\n') && (ch!=EOF))
        {
              szBuffer[i]=(char)ch;
          ch=fgetc(fp);
          i++;
        }
            TextOut(hdc,10,charht*(t-posvertscroll),szBuffer,i);
        t++;
      }
    }
    fclose(fp);

ReleaseDC(hwnd,hdc);
return;
  }

//*****************************************************************************
****
//   FUNCION COORDENADAS

static void COORDENADAS(HWND hwnd,int posicion_x,int posicion_y){

HDC hdc;
hdc=GetDC(hwnd);
   if(ejecucion==TRUE){
      if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
                        if(posicion_y>(factor_alto      *394)      &&      posicion_y<(factor_alto
*435)){SetCursor(Cursor_Flecha);}
                    else      if(posicion_y>(factor_alto      *349)      &&      posicion_y<(factor_alto
*390)){SetCursor(Cursor_Flecha);}
                    else      if(posicion_y>(factor_alto      *304)      &&      posicion_y<(factor_alto
*345)){SetCursor(Cursor_Flecha);}
          else SetCursor(Cursor_Cruz);}
        else SetCursor(Cursor_Cruz);}
   else

   if(ejecucion_1==TRUE){
        if(posicion_x>(factor_ancho *248) && posicion_x<(factor_ancho *380)){
                        if(posicion_y>(factor_alto      *50)      &&      posicion_y<(factor_alto
*70)){SetCursor(Cursor_Flecha);}
                    else      if(posicion_y>(factor_alto      *86)      &&      posicion_y<(factor_alto
*159)){SetCursor(Cursor_Flecha);}
                    else      if(posicion_y>(factor_alto      *215)      &&      posicion_y<(factor_alto
*289)){SetCursor(Cursor_Flecha);}
                    else      if(posicion_y>(factor_alto      *352)      &&      posicion_y<(factor_alto
*425)){SetCursor(Cursor_Flecha);}
          else SetCursor(Cursor_Cruz);}
      else

        if(posicion_x>(factor_ancho *75) && posicion_x<(factor_ancho *154)){
                        if(posicion_y>(factor_alto      *189)      &&      posicion_y<(factor_alto
*310)){SetCursor(Cursor_Flecha);}
          else SetCursor(Cursor_Cruz);}
        else

        if(posicion_x>(factor_ancho *484) && posicion_x<(factor_ancho *563)){
```

```
                    if(posicion_y>(factor_alto *190) && posicion_y<(factor_alto
*311)){SetCursor(Cursor_Flecha);}
                else SetCursor(Cursor_Cruz);}
              else

        if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
                    if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto
*435)){SetCursor(Cursor_Flecha);}
                else SetCursor(Cursor_Cruz);}
              else SetCursor(Cursor_Cruz);}

    else

    if(ejecucion_1_1==TRUE){
          if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
                    if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto
*435)){SetCursor(Cursor_Flecha);}
                else  if(posicion_y>(factor_alto *349) && posicion_y<(factor_alto
*390)){SetCursor(Cursor_Flecha);}
                else      if(posicion_y>(factor_alto *304) && posicion_y<(factor_alto
*345)){SetCursor(Cursor_Flecha);}
              else SetCursor(Cursor_Cruz);}
          else SetCursor(Cursor_Cruz);}

    else

    if(seleccion==TRUE){
          if(posicion_y>(factor_alto *161) && posicion_y<(factor_alto *307)){
                    if(posicion_x>(factor_ancho *33) && posicion_x<(factor_ancho
*198)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *238) && posicion_x<(factor_ancho
*404)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *442) && posicion_x<(factor_ancho
*604)){SetCursor(Cursor_Flecha);}
              else SetCursor(Cursor_Cruz);}
          else
          if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto *435)){
                    if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho
*623)){SetCursor(Cursor_Flecha);}
              else SetCursor(Cursor_Cruz);}
          else SetCursor(Cursor_Cruz);}

    else

    if(evaluacion==TRUE){
          if(posicion_y>(factor_alto *395) && posicion_y<(factor_alto *435)){
                    if(posicion_x>(factor_ancho *100) && posicion_x<(factor_ancho
*150)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *160) && posicion_x<(factor_ancho
*210)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *220) && posicion_x<(factor_ancho
*270)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *280) && posicion_x<(factor_ancho
*330)){SetCursor(Cursor_Flecha);}
                else      if(posicion_x>(factor_ancho *340) && posicion_x<(factor_ancho
*400)){SetCursor(Cursor_Flecha);}
              else SetCursor(Cursor_Cruz);}
        else SetCursor(Cursor_Cruz);}
```

```
   else SetCursor(Cursor_Cruz);

ReleaseDC(hwnd,hdc);
return;

} // fin de la funcion COORDENADAS


**************************************

//-------------------------------------------------------------------------------
---
//-------------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "VARIOS.H"
#include "TESIS.H"
extern int cantidad_preguntas;
extern int var_z,var_w,j;
extern char pregunta[5],respuesta;
extern RECT limite;


//*****************************************************************************
****
// FUNCION EVALUACION MODULO 1 Y 2

void EVALUACION_MODULO_1_2(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,'TRANSPARENT');

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1)  CUANTOS  MODULOS  BASICOS  CONFORMAN  A  LA
COMPUTADORA:",54);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) 3",4);
  TextOut(hdc,330,50,"b) 2",4);
  TextOut(hdc,20,70,"c) 6",4);
  TextOut(hdc,330,70,"d) 5",4);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) LA UNIDAD ARIT. \ LOGICA Y LA DE CONTROL CONFORMAN:",53);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) AL CPU.",10);
  TextOut(hdc,315,120,"b) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,20,140,"c) LA UNIDAD DE MEMORIA.",24);
  TextOut(hdc,315,140,"d) LA MICROCOMPUTADORA.",23);
 pregunta[1]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) EN UN SISTEMA  DE COMPUTACION EXISTEN FLUJOS DE:",51);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) UNICAMENTE SEÑALES DE CONTROL.",32);
```

```
  TextOut(hdc,330,190,"b) UNICAMENTE DE INFORMACION.",28);
  TextOut(hdc,20,210,"c) DE SEÑALES DE CONTROL Y DE DATOS.",35);
  TextOut(hdc,330,210,"d) NO EXISTEN FLUJOS.",21);
 pregunta[2]='c';

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_z=4;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
// FUNCION EVALUACION MODULO 3 Y 4

void EVALUACION_MODULO_3_4(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) AQUI SE REALIZAN LAS OPERACIONES ARITMETICO \ LOGICAS:",56);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) EN EL CPU.",13);
  TextOut(hdc,290,50,"b) EN LA UNIDAD DE MEMORIA.",27);
  TextOut(hdc,20,70,"c) EN LA UNIDA DE SALIDA.",25);
  TextOut(hdc,290,70,"d) EN LA ALU.",13);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) EL TIPO DE OPERACION A REALIZAR LA DETERMINA LA UNIDA
DE:",60);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) ENTRADA.",11);
  TextOut(hdc,330,120,"b) MEMORIA.",11);
  TextOut(hdc,20,140,"c) SALIDA.",10);
  TextOut(hdc,330,140,"d) CONTROL.",11);
 pregunta[1]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LOS DATOS QUE SERAN UTILIZADOS POR LA UAL PROVIENEN
DE:",58);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) LA U.DE MEMORIA. Y U. DE ENTRADA.",36);
  TextOut(hdc,330,190,"b) LA U.DE CONTROL U U.MEMORIA.",31);
  TextOut(hdc,20,210,"c) LA U.ENTRADA Y U.CONTROL.",28);
  TextOut(hdc,330,210,"d) DE CUALQUIER MODULO.",23);
 pregunta[2]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) LOS RESULTADOS DE OPERACIONES EN LA UAL SE TRANSFIEREN A
LA UNIDAD DE:",73);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) ENTRADA Y SALIDA.",20);
  TextOut(hdc,330,260,"b) CONTROL.",11);
  TextOut(hdc,20,280,"c) ENTRADA Y CONTROL.",21);
```

```
  TextOut(hdc,330,280,"d) MEMORIA Y DE SALIDA.",23);
  pregunta[3]='d';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_z=7;
ReleaseDC(hwnd,hdc);
}


// FUNCION EVALUACION MODULO 3 Y 4 NUMERO 2

 void EVALUACION_MODULO_3_4_1(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) ALMACENA GRUPOS DE DIGITOS BINARIOS:",39);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LA UNIDAD DE SALIDA.",23);
  TextOut(hdc,300,50,"b) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,20,70,"c) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,300,70,"d) LA ALU.",10);
  pregunta[0]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2)  ALMACENA  RESULTADOS  INTERMEDIOS  Y  FINALES  DE
OPERACIONES:",60);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,330,120,"b) UNIDAD DE CONTROL.",21);
  TextOut(hdc,20,140,"c) UNIDAD DE ENTRADA.",21);
  TextOut(hdc,330,140,"d) UNIDAD DE SALIDA.",20);
  pregunta[1]='a';


cantidad_preguntas=2;
ULTIMA_IMAGEN(hwnd);
var_z=8;
ReleaseDC(hwnd,hdc);
}



//**********************************************************************************
****
// FUNCION EVALUACION MODULO 5 Y 6

 void EVALUACION_MODULO_5_6(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
```

```
TextOut(hdc,10,30,"1) QUIEN CONTROLA LAS OPERACIONES DE LECTURA O
ESCRITURA:",57);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,310,50,"b) LA UNIDAD DE CONTROL.",24);
  TextOut(hdc,20,70,"c) LA ALU.",10);
  TextOut(hdc,310,70,"d) LA MEMORIA.",14);
 pregunta[0]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) PARA ACCESAR UNA LOCALIDAD DE MEMORIA REQUERIMOS:",52);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) UN CODIGO DE DIRECCION.",26);
  TextOut(hdc,330,120,"b) UNA INSTRUCCION.",19);
  TextOut(hdc,20,140,"c) UN COMANDO DE LECTURA.",25);
  TextOut(hdc,330,140,"d) NADA.",8);
 pregunta[1]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LA INFORMACION ALMACENADA EN LA MEMORIA PROVIENE DE
LA UNIDAD DE:",68);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) ENTRADA Y LA ALU.",20);
  TextOut(hdc,230,190,"b) ENTRADA Y DE CONTROL.",24);
  TextOut(hdc,20,210,"c) SALIDA Y ALU.",16);
  TextOut(hdc,230,210,"d) DE CUALQUIER UNIDAD.",23);
 pregunta[2]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) LA INFORMACION DE LA MEMORIA PUEDE LEERSE EN:",48);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) LA ALU Y LA U. DE ENTRADA.",29);
  TextOut(hdc,330,260,"b) U. DE SALIDA Y LA ALU.",25);
  TextOut(hdc,20,280,"c) LA UNIDA DE CONTROL.",23);
  TextOut(hdc,330,280,"d) LA UNIDAD DE CONTROL Y LA ALU.",33);
 pregunta[3]='b';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_z=11;
ReleaseDC(hwnd,hdc);
}

//*****************************************************************************
****
//  FUNCION EVALUACION MODULO 7 Y 8

 void EVALUACION_MODULO_7_8(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) ESTA CONFORMADA POR DISPOSITIVOS QUE INTRODUCEN DATOS O
INFORMACION:",71);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LA UNIDAD DE SALIDA.",23);
```

```
  TextOut(hdc,310,50,"b) UNIDAD DE CONTROL.",21);
  TextOut(hdc,20,70,"c) UNIDAD DE ENTRADA.",21);
  TextOut(hdc,310,70,"d) UNIDAD DE MEMORIA.",21);
 pregunta[0]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) LOS DATOS O INFORMACION DE ENTRADA SE COLOCAN EN LA
UNIDAD DE:",65);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) MEMORIA O DE SALIDA.",23);
  TextOut(hdc,330,120,"b) ALU O U. DE MEMORIA.",23);
  TextOut(hdc,20,140,"c) CONTROL O U. DE SALIDA.",26);
  TextOut(hdc,330,140,"d) ENTRADA O ALU.",17);
 pregunta[1]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) ES QUIEN DETERMINA HACIA DONDE SE MANDA LA INFORMACION
DE ENTRADA:",69);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,330,190,"b) CPU.",7);
  TextOut(hdc,20,210,"c) UNIDAD DE CONTROL.",21);
  TextOut(hdc,330,210,"d) ALU.",7);
 pregunta[2]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) ES QUIEN TRANSFIERE DATOS O INFORMACION AL EXTERIOR:",55);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) UNIDAD ARITMETICO / LOGICA.",30);
  TextOut(hdc,330,260,"b) EL CPU.",10);
  TextOut(hdc,20,280,"c) UNIDAD DE CONTROL.",21);
  TextOut(hdc,330,280,"d) UNIDAD DE SALIDA.",20);
 pregunta[3]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,310,"5) LOS DISPOSITIVOS DE SALIDA PUEDEN RECIBIR DATOS DE:",54);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,330,"a) UNIDAD ARITMETICO / LOGICA.",30);
  TextOut(hdc,330,330,"b) ALU O U. DE MEMORIA.",23);
  TextOut(hdc,20,350,"c) CPU O U.DE CONTROL.",22);
  TextOut(hdc,330,350,"d) CPU O ALU.",13);
 pregunta[4]='a';

cantidad_preguntas=5;
ULTIMA_IMAGEN(hwnd);
var_z=14;
ReleaseDC(hwnd,hdc);
}

//**********************************************************************************
****
// FUNCION EVALUACION MODULO 9 Y 10

void EVALUACION_MODULO_9_10(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);
```

```
SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) ES EL ASPECTO MAS IMPORTANTE DE LAS UNIDADES DE ENTRADA \
SALIDA:",68);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LOS REGISTROS.",17);
  TextOut(hdc,370,50,"b) LAS LINEA DE ENTRADA.",24);
  TextOut(hdc,20,70,"c) LOS CIRCUITOS LOGICOS.",25);
  TextOut(hdc,310,70,"d) LA SINCRONIZACION.",20);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) SIRVE PARA LOGRAR LA COMPATIBILIDAD ENTRE DISPOSITIVOS
DISIMILARES:",70);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) LOS ACUMULADORES.",20);
  TextOut(hdc,330,120,"b) LAS LINEAS DE SALIDA.",24);
  TextOut(hdc,20,140,"c) LA SINCRONIZACION.",20);
  TextOut(hdc,400,140,"d) LOS REGISTROS.",17);
 pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) ES DONDE SE ELIGE LA SIGUIENTE INSTRUCCION Y SE INTERPRETA
SE DIRECCION:",75);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,420,190,"b) UNIDAD DE SALIDA.",20);
  TextOut(hdc,20,210,"c) ALU.",7);
  TextOut(hdc,230,210,"d) UNIDAD DE CONTROL.",21);
 pregunta[2]='d';

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_z=17;
ReleaseDC(hwnd,hdc);
}

//-----------------------------------------------------------------------------------------------
---
//-----------------------------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "STRING.H"
#include "TESIS.H"

#define cuenta1 5

extern HANDLE hInst;
extern char Tex_Boton[10];
extern BOOL respaldo_pantalla;
extern BOOL ultima_imagen;
extern RECT limite;
extern HDC hdcMemory,hdcUltima_Imagen; // contextos de visualizacion de pag. oculta
extern HBITMAP hbmpMyBitmap,hbmpOld, hbmpUltima_Imagen,hbmpUltima_Imagen1;
extern RECT limite;
extern HFONT Font,PrevFont,PreFuente,Fuente;
```

```
// ULTIMA IMAGEN

 void ULTIMA_IMAGEN(HWND hwnd){
HDC hdc;
hdc=GetDC(hwnd);
if(ultima_imagen==FALSE){
  SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen1);
  DeleteObject(hbmpUltima_Imagen);}
hdcUltima_Imagen=CreateCompatibleDC(hdc);
hbmpUltima_Imagen=CreateCompatibleBitmap(hdc,limite.right,limite.bottom);
hbmpUltima_Imagen1=SelectObject(hdcUltima_Imagen,hbmpUltima_Imagen);
BitBlt(hdcUltima_Imagen,0,0,limite.right,limite.bottom,hdc,0,0,SRCCOPY);
ultima_imagen=FALSE;
ReleaseDC(hwnd,hdc);
}



//**********************************************************************************
****
//  FUNCION CREA BOTONES

 void CREA_BOTONES(HDC hdc,int a, int b, int c, int d,char *s,int f,int p){
  // F= NUMERO DE CARACTERES  Y  P  POSICION ON / OFF DEL BOTON
  int w,x,y,z,g=0,h=0,i=0,j=0,k=0,l=0,v=0;

  HBRUSH Brocha, Brocha1;
  HPEN Pluma, Pluma1;

  w=a; x=b, y=c; z=d;

Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,a,b,c,d);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

  switch(p){

  case 1:

  SetTextColor(hdc,RGB(0,0,0));
  SetBkMode(hdc,TRANSPARENT);

  Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
  Pluma1= SelectObject(hdc,Pluma);
  MoveTo(hdc,w+2,z-3);
  LineTo(hdc,y-3,z-3);
  LineTo(hdc,y-3,x+3);
  SelectObject(hdc,Pluma1);
  DeleteObject(Pluma1);


    Brocha= CreateSolidBrush(RGB(191,191,191));
    Pluma= CreatePen(PS_SOLID,2,RGB(191,191,191));
    Brocha1= SelectObject(hdc,Brocha);
    Pluma1= SelectObject(hdc,Pluma);
```

```
h=f*7; v=y-w; g=v-h; i=(g/2);
j=z-x; k=j-8; l=(k/2)+x-4;

Rectangle(hdc,a+4,b+4,c-3,d-3);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TextOut(hdc,a+i-4,l,s,f);

break;

case 2:


// SetTextColor(hdc,RGB(0,0,0));
// SetBkMode(hdc,TRANSPARENT);

Brocha= GetStockObject(HOLLOW_BRUSH);
Brocha1= SelectObject(hdc,Brocha);
Pluma= CreatePen(PS_SOLID,3,RGB(191,191,191));
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,a+1,b+1,c-1,d-1);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= GetStockObject(HOLLOW_BRUSH);
Brocha1= SelectObject(hdc,Brocha);
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,a,b,c,d);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);


//MoveTo(hdc,w+2,z-3);
//LineTo(hdc,y-3,z-3);
//LineTo(hdc,y-3,x+3);
//SelectObject(hdc,Pluma1);
//DeleteObject(Pluma1);


//  Brocha= CreateSolidBrush(RGB(191,191,191));
// Pluma= CreatePen(PS_SOLID,2,RGB(191,191,191));
//  Brocha1= SelectObject(hdc,Brocha);
//  Pluma1= SelectObject(hdc,Pluma);

//h=f*7; v=y-w; g=v-h; i=(g/2);
//j=z-x; k=j-8; l=(k/2)+x-4;

//TextOut(hdc,a+i-4,l,s,f);
```

```
/*                //127
 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);

  Brocha= CreateSolidBrush(RGB(191,191,191));
  Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);

 h=f*7; v=y-w; g=v-h; i=(g/2)+1;
 j=z+x; k=j-8; l=(k/2)+x-4;
 Rectangle(hdc,a,b,c,d);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);
 TextOut(hdc,a+i-4,l,s,f);  */

 break;

  }

 }


//    DIBUJA BOTONES

  void BOTONES(HWND hwnd,int s){

  HDC hdc;
  GetClientRect(hwnd, &limite);
  hdc=GetDC(hwnd);
  SetMapMode(hdc, MM_ANISOTROPIC);
  SetWindowExt(hdc, 624,442);
  SetViewportExt(hdc, limite.right, limite.bottom);
switch(s){

  case 1:
  strcpy(Tex_Boton,"SALIR");
  CREA_BOTONES(hdc,555,394,623,435,Tex_Boton,5,1);
  break;
  case 2:
  strcpy(Tex_Boton,"SALIR");
  CREA_BOTONES(hdc,555,394,623,435,Tex_Boton,5,2);
  break;
  case 3:
  strcpy(Tex_Boton,"AVANCE");
  CREA_BOTONES(hdc,555,349,623,390,Tex_Boton,6,1);
  break;
  case 4:
  strcpy(Tex_Boton,"AVANCE");
  CREA_BOTONES(hdc,555,349,623,390,Tex_Boton,6,2);
  break;
  case 5:
  strcpy(Tex_Boton,"ATRAS");
  CREA_BOTONES(hdc,555,304,623,345,Tex_Boton,5,1);
  break;
```

```
  case 6:
  strcpy(Tex_Boton,"ATRAS");
  CREA_BOTONES(hdc,555,304,623,345,Tex_Boton,5,2);

  break;
  }
  ReleaseDC(hwnd, hdc);

  return;
  }


//*****************************************************************************
****
// FUNCION TIEMPO


 void TIEMPO(int c){
int a,b;
for(a=0;a<=c;a++){
 for(b=0;b<=20000,b++);}

}


//*****************************************************************************
****
//  FUNCION CADENA DE APARICION SIN VENTANA

 void VER_CADENA_SIN(HWND hwnd,int a,int b,char *c,int lon,int rojo,int verde,int azul){
 int q=0;
 HDC hdc;
 lon+=1;
 GetClientRect(hwnd, &limite);
 hdc=GetDC(hwnd);
 SetMapMode(hdc, MM_ANISOTROPIC);
 SetWindowExt(hdc, 624,442);
 SetViewportExt(hdc, limite.right, limite.bottom);
  Font= CreateFont(   // para cambiar el texto de tamaño
    14,                          /* altura, en pixels */
    0,     /* usar la raz¢n de aspecto para determinar la anchura */
    0,              /*  angulo de la l¡nea de texto, en .1 grados */
    0,               /*  ngulo de cada car cter, en .1 grados */
    FW_BOLD,                     /* ¢ FW_LIGHT ¢ FW_NORMAL */
    FALSE,                    /* TRUE para que sea cursiva */
    FALSE,                   /* TRUE para que est, subrayada */
    FALSE,                    /* TRUE para que est, tachada */
    ANSI_CHARSET,               /* juego de caracteres ANSI */
    OUT_DEFAULT_PRECIS,           /* m,todo de correspondencia */
    CLIP_DEFAULT_PRECIS,             /* m,todo de recorte */
    DRAFT_QUALITY,                /* escalado activado */
    VARIABLE_PITCH | FF_SWISS,        /* inclinaci¢n y familia */
    "Tms Rmn");                  /* nombre de la tipograf¡a */
 PrevFont= SelectObject(hdc,Font);          /* seleccionar la fuente */

 SetTextColor(hdc,RGB(rojo,verde,azul));
 SetBkMode(hdc,TRANSPARENT);
MessageBeep(0);
```

```
while(q!=lon){
TextOut(hdc,a,b,c,q);
TIEMPO(cuenta1);
q++;
}
SelectObject(hdc,PrevFont);
DeleteObject(Font);
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//   FUNCION CADENA DE APARICION CON VENTANA

 void VER_CADENA_CON(HWND hwnd,int t_r,int t_v,int t_a,int f_r,int f_v,
            int f_a,char *c,char *f,int s,int x1,int y1,int x2,int y2,int t1,int t2)
{

int a=0,b=0;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma, Pluma1;
 HDC hdc;
 GetClientRect(hwnd, &limite);
 hdc=GetDC(hwnd);
 SetMapMode(hdc, MM_ANISOTROPIC);
 SetWindowExt(hdc, 624,442);
 SetViewportExt(hdc, limite.right, limite.bottom);
t1+=1;
t2+=1;

SetBkColor(hdc,RGB(f_r,f_v,f_a));
SetTextColor(hdc,RGB(t_r,t_v,t_a));

Brocha= CreateSolidBrush(RGB(0,0,0));
Pluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);
Pluma1= SelectObject(hdc,Pluma);

Rectangle(hdc,8,5,621,58);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 switch(s){

   case 1:
    MessageBeep(s); break;
   case 0:
     break;

    }
 Font= CreateFont(   // para cambiar el texto de tamaño
    14,                          /* altura, en pixels */
    0,      /* usar la razon de aspecto para determinar la anchura */
    0,             /* angulo de la linea de texto, en .1 grados */
    0,               /* angulo de cada caracter, en .1 grados */
```

```
    FW_BOLD,                    /* e FW_LIGHT e FW_NORMAL */
    FALSE,                    /* TRUE para que sea cursiva */
    FALSE,                   /* TRUE para que est. subrayada */
    FALSE,                  /* TRUE para que est. tachada */
    ANSI_CHARSET,             /* juego de caracteres ANSI */
    OUT_DEFAULT_PRECIS,         /* m,todo de correspondencia */
    CLIP_DEFAULT_PRECIS,          /* m,todo de recorte */
    DRAFT_QUALITY,              /* escalado activado */
    VARIABLE_PITCH | FF_SWISS,     /* inclinacien y familia */
    "Tms Rmn");                /* nombre de la tipograf|a */
  PrevFont= SelectObject(hdc,Font);       /* seleccionar la fuente */

while(a!=t1){
TextOut(hdc,x1,y1,c,a);
TIEMPO(cuenta1);
a++;  }
while(b!=t2){
TextOut(hdc,x2,y2,f,b);
TIEMPO(cuenta1);
b++; }

SelectObject(hdc,PrevFont);
DeleteObject(Font);
ReleaseDC(hwnd,hdc);

}


// CREA RECTANGULOS BIDIMENSIONALES

 void CREA_RECTANGULO_BIDIMENSIONAL(HWND hwnd,int x1,int y1,int x2,int y2,
                                    int c1,int c2,int c3,int orientacion){
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;

POINT puntos[4];
POINT puntos1[4];

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);

switch(orientacion){
 case 1:
        puntos[0].x=x1;  puntos[0].y=y1;
        puntos[1].x=x1-20;  puntos[1].y=y1-20;
        puntos[2].x=x2-20;  puntos[2].y=y1-20;
        puntos[3].x=x2;  puntos[3].y=y1;

        puntos1[0].x=x1;  puntos1[0].y=y1;
        puntos1[1].x=x1-20,  puntos1[1].y=y1-20;
        puntos1[2].x=x1-20,  puntos1[2].y=y2-20;
        puntos1[3].x=x1;  puntos1[3].y=y2;
  break;
 case 2:
```

```
        puntos[0].x=x1;  puntos[0].y=y1;
        puntos[1].x=x1+20;  puntos[1].y=y1-20;
        puntos[2].x=x2+20;  puntos[2].y=y1-20;
        puntos[3].x=x2;  puntos[3].y=y1;

        puntos1[0].x=x2;  puntos1[0].y=y2;
        puntos1[1].x=x2+20;  puntos1[1].y=y2-20;
        puntos1[2].x=x2+20;  puntos1[2].y=y1-20;
        puntos1[3].x=x2;  puntos1[3].y=y1;
 break;
}

 Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(c1,c2,c3));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,x1,y1,x2,y2);
 Polygon(hdc,puntos,4);
 Polygon(hdc,puntos1,4);

 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 ReleaseDC(hwnd,hdc);
} //  TERMINA CEREA RECTANGULO BIDIMENSIONAL


//*****************************************************************************
****
//*  FUNCION INT_MODULOS (dibuja los modulos intermitentes

 void INT_MODULOS(HDC hdc,int a,int b,int c){

 HBRUSH Brocha, wBrocha1;
 HPEN wPluma, wPluma1;

 RECT alu    ={248,86,380,159};
 RECT control ={248,215,380,289};
 RECT memoria ={248,352,380,425};
 RECT entrada ={75,189,154,310};
 RECT salida  ={484,190,563,311};
switch(a){

 case 1:
  Brocha= CreateSolidBrush(RGB(255,255,255));
  wPluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
  wBrocha1= SelectObject(hdc,Brocha);
  wPluma1= SelectObject(hdc,wPluma);

  SetTextColor(hdc,RGB(0,0,255));
  SetBkColor(hdc,RGB(255,255,255));
  break;

 case 2:
  Brocha= CreateSolidBrush(RGB(0,0,0));
  wPluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
  wBrocha1= SelectObject(hdc,Brocha);
```

```
  wPluma1= SelectObject(hdc,wPluma);

  SetTextColor(hdc,RGB(255,255,0));
  SetBkColor(hdc,RGB(0,0,0));
  break;

 case 3:
  Brocha= CreateSolidBrush(RGB(255,255,0));
  wPluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
  wBrocha1= SelectObject(hdc,Brocha);
  wPluma1= SelectObject(hdc,wPluma);

  SetTextColor(hdc,RGB(0,0,0));
  SetBkColor(hdc,RGB(255,255,0));
  break;

 }

switch(b){

 case 1:
   Rectangle(hdc,alu.left,alu.top,alu.right,alu.bottom);
   TextOut(hdc,300,115,"UAL",3);
  switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 2:
   Rectangle(hdc,control.left,control.top,control.right,control.bottom);
   TextOut(hdc,287,247,"CONTROL",7);
  switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 3:
   Rectangle(hdc,memoria.left,memoria.top,memoria.right,memoria.bottom);
   TextOut(hdc,287,384,"MEMORIA",7);
  switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 4:
   Rectangle(hdc,entrada.left,entrada.top,entrada.right,entrada.bottom);
   TextOut(hdc,85,245,"ENTRADA",7);
  switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 5:
   Rectangle(hdc,salida.left,salida.top,salida.right,salida.bottom);
```

```
    TextOut(hdc,500,245,"SALIDA",6);
   switch(c){
    case 1: MessageBeep(0); break;
    case 2: MessageBeep(-1); break;
    case 3: break;
          }
   break;
 case 6:
   Rectangle(hdc,alu.left,alu.top,alu.right,alu.bottom);
   TextOut(hdc,300,115,"UAL",3);
   Rectangle(hdc,control.left,control.top,control.right,control.bottom);
   TextOut(hdc,287,247,"CONTROL",7);
   Rectangle(hdc,memoria.left,memoria.top,memoria.right,memoria.bottom);
   TextOut(hdc,287,384,"MEMORIA",7);
   Rectangle(hdc,entrada.left,entrada.top,entrada.right,entrada.bottom);
   TextOut(hdc,85,245,"ENTRADA",7);
   Rectangle(hdc,salida.left,salida.top,salida.right,salida.bottom);
   TextOut(hdc,500,245,"SALIDA",6);

   switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
   break;

 case 7:
   Rectangle(hdc,alu.left,alu.top,alu.right,alu.bottom);
   TextOut(hdc,300,115,"UAL",3);
   Rectangle(hdc,control.left,control.top,control.right,control.bottom);
   TextOut(hdc,287,247,"CONTROL",7);

   switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
   break;

 case 8:
   Rectangle(hdc,entrada.left,entrada.top,entrada.right,entrada.bottom);
   TextOut(hdc,85,245,"ENTRADA",7);
   Rectangle(hdc,salida.left,salida.top,salida.right,salida.bottom);
   TextOut(hdc,500,245,"SALIDA",6),

   switch(c){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
   break;
 }

SelectObject(hdc,wBrocha1);
DeleteObject(Brocha);
SelectObject(hdc,wPluma1);
DeleteObject(wPluma);
}
```

```
//**********************************************************************************
****
// FUNCION MODULOS
 void MODULOS(HDC hdc){

hbmpMyBitmap = LoadBitmap(hInst, "MyBitmap");
hdcMemory = CreateCompatibleDC(hdc);
hbmpOld = SelectObject(hdcMemory, hbmpMyBitmap);

BitBlt(hdc,0,0,640,442,hdcMemory,0,0,SRCCOPY);
SelectObject(hdcMemory,hbmpOld);
DeleteDC(hdcMemory);

}


//          Limpiar el  Area de trabajo de la ventana.


 void zResetFunc(HWND hWnd)
{
  HDC hDC;
  HBRUSH Brocha1, Brocha;
  RECT rcClientArea;
  HPEN Pluma1, Pluma;
Brocha= CreateSolidBrush(RGB(0,64,128));
Pluma= CreatePen(PS_SOLID,1,RGB(0,64,128));
hDC= GetDC(hWnd);
Brocha1= SelectObject(hDC,Brocha);
Pluma1= SelectObject(hDC,Pluma);
GetClientRect(hWnd,&rcClientArea);
FillRect(hDC,&rcClientArea,Brocha);
SelectObject(hDC,Pluma1);
SelectObject(hDC,Brocha1);
ReleaseDC(hWnd,hDC);
DeleteObject(Brocha);
DeleteObject(Pluma);
return;
}


//     DIBUJA BOTONES EVALUACION

   void BOTONES_EVALUACION(HWND hwnd,int s){

  HDC hdc;
  GetClientRect(hwnd, &limite);
  hdc=GetDC(hwnd);
  SetMapMode(hdc, MM_ANISOTROPIC);
  SetWindowExt(hdc, 624,442);
  SetViewportExt(hdc, limite.right, limite.bottom);
switch(s){

  case 1:
  strcpy(Tex_Boton,"a");
  CREA_BOTONES(hdc,100,395,150,435,Tex_Boton,1,1);
```

```
break;
case 2:
strcpy(Tex_Boton,"a");
CREA_BOTONES(hdc,100,395,150,435,Tex_Boton,1,2);
break;
case 3:
strcpy(Tex_Boton,"b");
CREA_BOTONES(hdc,160,395,210,435,Tex_Boton,1,1);
break;
case 4:
strcpy(Tex_Boton,"b");
CREA_BOTONES(hdc,160,395,210,435,Tex_Boton,1,2);
break;
case 5:
strcpy(Tex_Boton,"c");
CREA_BOTONES(hdc,220,395,270,435,Tex_Boton,1,1);
break;
case 6:
strcpy(Tex_Boton,"c");
CREA_BOTONES(hdc,220,395,270,435,Tex_Boton,1,2);
break;
case 7:
strcpy(Tex_Boton,"d");
CREA_BOTONES(hdc,280,395,330,435,Tex_Boton,1,1);
break;
case 8:
strcpy(Tex_Boton,"d");
CREA_BOTONES(hdc,280,395,330,435,Tex_Boton,1,2);
break;
case 9:
strcpy(Tex_Boton,">>");
CREA_BOTONES(hdc,340,395,390,435,Tex_Boton,2,1);
break;
case 10:
strcpy(Tex_Boton,">>");
CREA_BOTONES(hdc,340,395,390,435,Tex_Boton,2,2);
break;
}
ReleaseDC(hwnd, hdc);
return;
}

//**********************************************************************************
****
// FUNCION BOTONES SELECCION

void BOTON_SELECCION(HDC hdc,int q,int a,int b,int c,int d){

HPEN Pluma, Pluma1;
HBRUSH Brocha, Brocha1;

Pluma= CreatePen(PS_SOLID,4,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= GetStockObject(HOLLOW_BRUSH);
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,a-4,b-4,c+4,d+4);
SelectObject(hdc,Pluma1);
```

```
 DeleteObject(Pluma1);
  switch(q){

 case 1:

 Pluma= CreatePen(PS_SOLID,4,RGB(191,191,191));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,a,b,c,d);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
 Pluma= CreatePen(PS_SOLID,4,RGB(127,127,127));
 Pluma1= SelectObject(hdc,Pluma);
 MoveTo(hdc,a+2,d-1);
 LineTo(hdc,c-1,d-1);
 LineTo(hdc,c-1,b+2);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

 break;

 case 2:

 Pluma= CreatePen(PS_SOLID,4,RGB(0,0,128));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,a,b,c,d);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

 break;

 case 3:

 Pluma= CreatePen(PS_SOLID,4,RGB(128,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,a,b,c,d);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

 break;

 case 4:

 Pluma= CreatePen(PS_SOLID,4,RGB(0,128,128));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,a,b,c,d);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

 break;

 }

 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 }

//     DIBUJA BOTONES 1
```

```
void BOTONES_1(HWND hwnd,int s){

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
switch(s){

case 1:
strcpy(Tex_Boton,"UAL");
CREA_BOTONES(hdc,248,86,380,159,Tex_Boton,3,1);
break;
case 2:
strcpy(Tex_Boton,"UAL");
CREA_BOTONES(hdc,248,86,380,159,Tex_Boton,3,2);
break;
case 3:
strcpy(Tex_Boton,"CONTROL");
CREA_BOTONES(hdc,248,215,380,289,Tex_Boton,7,1);
break;
case 4:
strcpy(Tex_Boton,"CONTROL");
CREA_BOTONES(hdc,248,215,380,289,Tex_Boton,7,2);
break;
case 5:
strcpy(Tex_Boton,"MEMORIA");
CREA_BOTONES(hdc,248,352,380,425,Tex_Boton,7,1);
break;
case 6:
strcpy(Tex_Boton,"MEMORIA");
CREA_BOTONES(hdc,248,352,380,425,Tex_Boton,7,2);
break;
case 7:
strcpy(Tex_Boton,"ENTRADA");
CREA_BOTONES(hdc,75,189,154,310,Tex_Boton,7,1);
break;
case 8:
strcpy(Tex_Boton,"ENTRADA");
CREA_BOTONES(hdc,75,189,154,310,Tex_Boton,7,2);
break;
case 9:
strcpy(Tex_Boton,"SALIDA");
CREA_BOTONES(hdc,484,190,563,311,Tex_Boton,6,1);
break;
case 10:
strcpy(Tex_Boton,"SALIDA");
CREA_BOTONES(hdc,484,190,563,311,Tex_Boton,6,2);
break;
case 11:
strcpy(Tex_Boton,"CPU");
CREA_BOTONES(hdc,248,50,380,70,Tex_Boton,3,1);
break;
case 12:
strcpy(Tex_Boton,"CPU");
CREA_BOTONES(hdc,248,50,380,70,Tex_Boton,3,2);
```

```
  break;
  }

  ReleaseDC(hwnd, hdc);

  return;
  }

//-----------------------------------------------------------------------------
---
//-----------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "VARIOS.H"
#include "TESIS.H"

#define cuenta2 70
#define cuenta4 100
#define cuenta5 20
#define cuenta6 200
#define cuenta7 150

extern HANDLE hInst;
extern HCURSOR Reloj_Arena;
extern HCURSOR Cursor_Cruz;
extern HDC hdcMicro; // contextos de visualizacion de pag. oculta
extern HBITMAP hbmpMicro,hbmpMicro1;
extern RECT limite;
extern int var_z,var_w,j;
extern HFONT PrevFont,Font,PreFuente,Fuente;
HFONT Font1,PrevFont1;


//*****************************************************************************
****
//  FUNCION MICROCOMPUTADORA 1

 void MICROCOMPUTADORA_1(HWND hwnd){

 int c;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;
HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);

 Brocha= CreateSolidBrush(RGB(127,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,0,0,642,444);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 SetCursor(Reloj_Arena);
```

```
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"EN LA FIGURA SE PRESENTAN LOS ELEMENTOS
BASICOS DE UNA",
                                    "MICROCOMPUTADORA ( µC ):",1,15,10,15,30,54,24);
TIEMPO(cuenta2);

CREA_RECTANGULO_BIDIMENSIONAL(hwnd,50,100,150,400,127,0,0,1);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,250,100,550,200,0,255,255,1);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,250,250,350,400,0,64,128,1);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,450,250,550,400,0,0,255,1);

 Brocha= CreateSolidBrush(RGB(0,0,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,120,135,180);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,200,135,380);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(0,128,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,270,115,370,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(128,128,64));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,430,115,530,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(127,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,260,260,340,315);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(127,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,260,335,340,390);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,460,260,540,315);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,460,335,540,390);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
```

```
Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,100,185);
LineTo(hdc,100,195);
MoveTo(hdc,95,190);
LineTo(hdc,100,195);
LineTo(hdc,105,190);

MoveTo(hdc,300,320);
LineTo(hdc,300,330);
MoveTo(hdc,295,325);
LineTo(hdc,300,320);
LineTo(hdc,305,325);

MoveTo(hdc,500,320);
LineTo(hdc,500,330);
MoveTo(hdc,495,325);
LineTo(hdc,500,330);
LineTo(hdc,505,325);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma);

Pluma= CreatePen(PS_SOLID,10,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,143,220);
LineTo(hdc,555,220);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma);

Pluma= CreatePen(PS_SOLID,4,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,320,190);
LineTo(hdc,320,210);
MoveTo(hdc,315,195);
LineTo(hdc,320,190);
LineTo(hdc,325,195);
MoveTo(hdc,315,205);
LineTo(hdc,320,210);
LineTo(hdc,325,205);

MoveTo(hdc,485,190);
LineTo(hdc,485,210);
MoveTo(hdc,480,205);
LineTo(hdc,485,210);
LineTo(hdc,490,205);

MoveTo(hdc,300,228);
LineTo(hdc,300,255);
MoveTo(hdc,295,233);
LineTo(hdc,300,228);
LineTo(hdc,305,233);

MoveTo(hdc,505,228);
LineTo(hdc,505,255);
MoveTo(hdc,500,250);
```

```
LineTo(hdc,505,255);
LineTo(hdc,510,250);

MoveTo(hdc,160,220);
LineTo(hdc,170,210);
LineTo(hdc,170,230);
LineTo(hdc,160,220);

MoveTo(hdc,190,210);
LineTo(hdc,200,220);
LineTo(hdc,190,230);
LineTo(hdc,190,210);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma);

SetTextColor(hdc,RGB(191,191,191));
SetBkColor(hdc,RGB(0,0,255));
TextOut(hdc,335,62,"UNIDAD DE MEMORIA",17);
TextOut(hdc,20,415,"CPU (ALU Y U.CONTROL)",21);
TextOut(hdc,266,415,"U.ENTRADA",9);
TextOut(hdc,470,415,"U.SALIDA",8);

Font1= CreateFont(   // para cambiar el texto de tamaño
   10,                        /* altura, en pixels */
   0,     /* usar la razon de aspecto para determinar la anchura */
   0,              /*  angulo de la l¡nea de texto, en .1 grados */
   0,               /*  ngulo de cada car cter, en .1 grados */
   FW_BOLD,                   /* ¢ FW_LIGHT ¢ FW_NORMAL */
   FALSE,                 /* TRUE para que sea cursiva */
   TRUE,                /* TRUE para que est, subrayada */
   FALSE,               /* TRUE para que est, tachada */
   ANSI_CHARSET,              /* juego de caracteres ANSI */
   OUT_DEFAULT_PRECIS,          /* m,todo de correspondencia */
   CLIP_DEFAULT_PRECIS,             /* m,todo de recorte */
   DRAFT_QUALITY,               /* escalado activado */
   VARIABLE_PITCH | FF_ROMAN,        /* inclinacion y familia */
   "Helv");              /* nombre de la tipograf¡a */
PrevFont1= SelectObject(hdc,Font1),          /* seleccionar la fuente */
SetTextColor(hdc,RGB(0,0,0));
SetBkColor(hdc,RGB(255,255,0));
TextOut(hdc,67,145,"CRONOMETROS",11);
TextOut(hdc,80,250,"MICRO",5);
TextOut(hdc,70,270,"PROCESADOR",10);
TextOut(hdc,84,290,"( µC )",6);
TextOut(hdc,310,150,"RAM",3);
TextOut(hdc,468,150,"ROM",3);
TextOut(hdc,264,275,"SINCRONIZACION",14);
TextOut(hdc,274,290,"DE ENTRADA",10);
TextOut(hdc,464,275,"SINCRONIZACION",14);
TextOut(hdc,476,290,"DE SALIDA",9);
TextOut(hdc,274,350,"DISPOSITIVOS",12);
TextOut(hdc,274,365,"DE ENTRADA",10);
TextOut(hdc,474,350,"DISPOSITIVOS",12);
TextOut(hdc,476,365,"DE SALIDA",9);

TIEMPO(cuenta4);
```

```
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"UNA MICROCOMPUTADORA CONTIENE VARIOS
ELEMENTOS, UNO DE LOS CUALES ES EL",
                                    "MICROPROCESADOR,EL CUAL ES LA UNIDAD CENTRAL DE
PROCESAMIENTO (CPU O UCP)",1,13,10,13,30,71,73);
 TIEMPO(cuenta4);

 for(c=0,c<=3;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(0,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,200,135,380);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,65,200,135,380);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TextOut(hdc,80,250,"MICRO",5);
 TextOut(hdc,70,270,"PROCESADOR",10);
 TextOut(hdc,84,290,"( µC )",6);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta4);
 VER_CADENA_CON(hwnd,255,0,0,0,0,0,  "NORMALMENTE EL MICROPROCESADOR ES UNO
O MAS INTEGRADOS 'LSI' QUE CONTIENE",
                                    "TODOS LOS CIRCUITOS DE CONTROL Y ARITMETICOS
DE LA MICROCOMPUTADORA",1,13,10,13,30,73,67);
 TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,0,255,0,0,0,0,     "LA UNIDAD DE MEMORIA MUESTRA
DISPOSITIVOS RAM Y ROM, COMUNES DE MUCHAS µC",
                                    ",AUNQUE NO SIEMPRE LOS DOS ESTAN
PRESENTES",1,13,10,13,30,73,42);
 TIEMPO(cuenta6);
 VER_CADENA_CON(hwnd,0,255,0,0,0,0,  "LA SECCION DE LA RAM CONSTA DE UNO O MAS
INTEGRADOS SEGUN LA",
                                    "CAPACIDAD DE LA µC ( MICROCOMPUTADORA
)",1,13,10,13,30,60,39);
 TIEMPO(cuenta4);
 for(c=0,c<=3;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(127,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,270,115,370,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(0,128,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,270,115,370,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TextOut(hdc,310,150,"RAM",3);
 TIEMPO(cuenta5);
 }
 TIEMPO(cuenta4);
```

```
  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"ESTA SECCION DE LA MEMORIA ALMACENA
PROGRAMAS Y DATOS QUE CAMBIAN CON",
                                    "FRECUENCIA DURANTE EL CURSO DE LA
OPERACION",1,13,10,13,30,69,43);
 TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA SECCION DE LA ROM ALMACENA
INSTRUCCIONES Y DATOS QUE NO VARIAN.",
                                    "ESTA MEMORIA NO REQUIERE DE ENERGIA ELECTRICA
PARA ALMACENAR",1,13,10,13,30,66,60);
 TIEMPO(cuenta4);
 for(c=0;c<=3;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(255,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,430,115,530,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(128,128,64));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,430,115,530,185);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TextOut(hdc,468,150,"ROM",3);
 TIEMPO(cuenta5);
 }
 TIEMPO(cuenta6);
   VER_CADENA_CON(hwnd,0,255,0,0,0,0,    "LAS SECCIONES DE ENTRADA / SALIDA,
CONTIENEN CIRCUITOS SINCRONIZADORES.",
                                    "",1,13,10,13,30,71,0);
 TIEMPO(cuenta4);
 for(c=0;c<=4;c++){
 MessageBeep(0);
 Pluma= CreatePen(PS_SOLID,2,RGB(0,255,255));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= GetStockObject(HOLLOW_BRUSH);
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,250,250,350,400);
 Rectangle(hdc,450,250,550,400);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 TIEMPO(cuenta5);
 Pluma= CreatePen(PS_SOLID,2,RGB(0,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= GetStockObject(HOLLOW_BRUSH);
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,250,250,350,400);
 Rectangle(hdc,450,250,550,400);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 TIEMPO(cuenta5);
 }
```

```
 TIEMPO(cuenta4);
   VER_CADENA_CON(hwnd,255,255,0,0,0,0,   "ESTOS CIRCUITOS SE NECESITAN PARA
PERMITIR QUE LOS DISPOSITIVOS DE ENTRADA",
                                          "Y SALIDA SE COMUNIQUEN ADECUADAMENTE CON EL
RESTO DE LA COMPUTADORA",1,13,10,13,30,74,67);
 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 SelectObject(hdc,PrevFont1);
 DeleteObject(Font1);

 var_w=2; j=7;
 ReleaseDC(hwnd,hdc);
} //  TERMINA FUNCION MICROCOMPUTADORA 1



//*****************************************************************************
****
//  FUNCION MICROCOMPUTADORA 2

 void MICROCOMPUTADORA_2(HWND hwnd){
 int d=0,c;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;

 HDC hdc;
 GetClientRect(hwnd, &limite);
 hdc=GetDC(hwnd);
 SetMapMode(hdc, MM_ANISOTROPIC);
 SetWindowExt(hdc, 624,442);
 SetViewportExt(hdc, limite.right, limite.bottom);

 SetCursor(Reloj_Arena);
 hbmpMicro = LoadBitmap(hInst, "MyBitmap2");
 hdcMicro = CreateCompatibleDC(hdc);
 hbmpMicro1 = SelectObject(hdcMicro, hbmpMicro);

 PatBlt(hdc,0,0,645,445,BLACKNESS);
  TIEMPO(cuenta7);
   VER_CADENA_CON(hwnd,255,255,0,0,0,0,"TODAS LAS ESTRUCTURAS POSIBLES EN LAS
MICROCOMPUTADORAS (µC), SON",
                                          "ESENCIALMENTE   IGUALES   EN
PRINCIPIO.",1,23,10,23,30,65,35);
  TIEMPO(cuenta7);
VER_CADENA_CON(hwnd,191,191,191,0,0,0,"LO QUE VARIA ES EL TAMAÑO DE LINEAS DE
DATOS Y DE DIRECCIONES,ASI COMO",
                                          "LOS TIPOS DE SEÑALES DE CONTROL QUE
UTILIZAN.",1,20,10,20,30,70,45);
 TIEMPO(cuenta7);
   VER_CADENA_CON(hwnd,0,255,255,0,0,0,"RETOMANDO LA FIGURA ANTERIOR PARA
ANALIZARLA MAS DETALLADAMENTE,",
                                          "OBSERVAMOS A LA µC, ASI COMO LAS LINEAS QUE LA
INTERCONECTAN.",1,23,10,23,30,64,61);
TIEMPO(cuenta7);
PatBlt(hdc,0,0,645,445,BLACKNESS);
```

```
if(limite.right==624){
 if(limite.bottom==442){
while(d<=640){
BitBlt(hdc,0,0,d,442,hdcMicro,0,0,SRCCOPY);
d+=2; } } else BitBlt(hdc,0,0,640,442,hdcMicro,0,0,SRCCOPY);}
else BitBlt(hdc,0,0,640,442,hdcMicro,0,0,SRCCOPY);

SelectObject(hdcMicro,hbmpMicro1);
DeleteDC(hdcMicro);

 Font1= CreateFont(   // para cambiar el texto de tamaño
    10,                          /* altura, en pixels */
    0,      /* usar la razón de aspecto para determinar la anchura */
    0,              /* angulo de la línea de texto, en .1 grados */
    0,               /* ngulo de cada car cter, en .1 grados */
    FW_BOLD,                     /* ¢ FW_LIGHT ¢ FW_NORMAL */
    FALSE,                    /* TRUE para que sea cursiva */
    FALSE,                   /* TRUE para que est, subrayada */
    FALSE,                    /* TRUE para que est, tachada */
    ANSI_CHARSET,             /* juego de caracteres ANSI */
    OUT_DEFAULT_PRECIS,          /* m,todo de correspondencia */
    CLIP_DEFAULT_PRECIS,            /* m,todo de recorte */
    DRAFT_QUALITY,                /* escalado activado */
    VARIABLE_PITCH | FF_SWISS,          /* inclinaci¢n y familia */
    "Helv");                  /* nombre de la tipografía */
 PrevFont1= SelectObject(hdc,Font1);          /* seleccionar la fuente */

 SetBkColor(hdc,RGB(255,255,0));
 SetTextColor(hdc,RGB(0,0,0));
 TextOut(hdc,327,137,"LINEA",5);
 TextOut(hdc,410,137,"DE CONTROL",10);
 TextOut(hdc,430,364,"PUERTA DE",9);
 TextOut(hdc,430,380,"SALIDA",6);
 TextOut(hdc,568,364,"PUERTA DE",9);
 TextOut(hdc,568,380,"ENTRADA",7);
 TextOut(hdc,337,113,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,385,344,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,406,230,"LINEA DE DATOS",14);

 TIEMPO(cuenta6);
     VER_CADENA_CON(hwnd,255,255,0,0,0,0,"EL TIPO DE ESTRUCTURA DE UNA
MICROCOMPUTADOR (µC), DEPENDE DEL NUMERO DE",
                                    "BITS QUE ESTA UTILIZE. ESTO SE EXPLICA EN LA
SECCION DE CONCEPTOS BASICOS",1,14,10,14,30,72,73);
 TIEMPO(cuenta6);
 VER_CADENA_CON(hwnd,191,191,191,0,0,0,  "LA µC CUENTA CON 3 LINEAS O BUSES,QUE
TRANSPORTAN TODA LA INFORMACION Y",
                                    "SEÑALES IMPLICADAS EN LA OPERACION DEL
SISTEMA Y UN BUS MAS DE ENERGIA.",1,14,10,14,30,71,71);
 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,0,255,255,0,0,0,   "UN BUS ES UNA O MAS RUTAS ELECTRICAS
SOBRE LAS",
                                    "CUALES SE CONDUCE INFORMACION O
CORRIENTE",1,14,10,14,30,46,41);
 TIEMPO(cuenta6);
  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LOS BUSES DE LA µC SON: LINEA O BUS DE
DIRECCIONES (UNIDIRECCIONAL),LINEA",
```

```
                                        "O BUS DE DATOS (BIDIRECCIONAL), LINEA O BUS DE
CONTROL Y BUS DE ENERGIA.",1,14,10,14,30,73,72);
SetTextColor(hdc,RGB(0,0,0));
for(c=0;c<=7;c++){
MessageBeep(0);
 SetBkColor(hdc,RGB(0,0,0));
 TextOut(hdc,327,137,"LINEA",5);
 TextOut(hdc,410,137,"DE CONTROL",10);
 TextOut(hdc,337,113,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,385,344,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,406,230,"LINEA DE DATOS",14);
 TIEMPO(cuenta5);

 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,327,137,"LINEA",5);
 TextOut(hdc,410,137,"DE CONTROL",10);
 TextOut(hdc,337,113,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,385,344,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,406,230,"LINEA DE DATOS",14);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,191,191,191,0,0,0,"BUS DE DIRECCIONES: LINEA UTILIZADA POR
EL BUS DE CONTROL PARA ENVIAR",
                                        "SEÑALES (DIRECCION) QUE SEÑALA UNA POSICION DE
MEMORIA REQUERIDA POR EL µP",1,14,10,14,30,69,74);
for(c=0;c<=6;c++){
MessageBeep(0);
 SetBkColor(hdc,RGB(0,0,0));
 TextOut(hdc,337,113,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,385,344,"LINEA DE DIRECCIONES",20);
 TIEMPO(cuenta5);

 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,337,113,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,385,344,"LINEA DE DIRECCIONES",20);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta6);
   VER_CADENA_CON(hwnd,0,255,255,0,0,0,   "BUS DE DATOS: LINEA UTILIZADA PARA
TRANSMITIR DATOS HACIA CIERTO NUMERO DE",
                                        "LOCALIZACIONES EN LA COMPUTADORA (UTILIZADA
EN LAS OPERQACIONES R/W)",1,14,10,14,30,74,68);
for(c=0;c<=6;c++){
MessageBeep(0);
 SetBkColor(hdc,RGB(0,0,0));
 TextOut(hdc,406,230,"LINEA DE DATOS",14);
 TIEMPO(cuenta5);

 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,406,230,"LINEA DE DATOS",14);
 TIEMPO(cuenta5);
 }

TIEMPO(cuenta6);
```

```
  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"BUS DE CONTROL: UTILIZADO PARA ENVIAR
SEÑALES A LA ALU Y A LA MEMORIA",
                                    "INTERNA A FIN DE EJECUTAR INSTRUCCIONES DE
PROGRAMA DE COMPUTADORA. ",1,14,10,14,30,69,67);
for(c=0;c<=6;c++){
MessageBeep(0);
 SetBkColor(hdc,RGB(0,0,0));
 TextOut(hdc,327,137,"LINEA",5);
 TextOut(hdc,410,137,"DE CONTROL",10);
 TIEMPO(cuenta5);
 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,327,137,"LINEA",5);
 TextOut(hdc,410,137,"DE CONTROL",10);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,191,191,191,0,0,0,"LAS SEÑALES DE LINEA DE CONTROL
ESPECIFICAN LA LOCALIZACION DE MEMORIA",
                                    "QUE INTERESA AL PROCESADOR.",1,14,10,14,30,70,27);
 TIEMPO(cuenta6);
  VER_CADENA_CON(hwnd,0,255,255,0,0,0,  "TAMBIEN INDICA MEDIANTE SEÑALES EN LA
LINEA DE CONTROL SI SE  DESEA LEER O",
                                    "ESCRIBIR  A  PARTIR  DE  UNA  POSICION  DE
MEMORIA",1,14,10,14,30,74,44);
 TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,255,255,0,0,0,0,"EL  BUS  DE  CONTROL  TAMBIEN  LLEVA
SEÑALES DE RELOJ QUE SINCRONIZAN",
                                    "EL, PROCESADOR Y LA MEMORIA.",1,20,10,20,30,64,28);
 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,191,191,191,0,0,0,"LAS SEÑALES DE DITRIBUCION: EN LA LINEA
DE CONTROL LAS SEÑALES DE CRONOMETRO",
                                    "SON LAS MAS IMPORTANTES, ESTAS GENERAN LOS
INTERVALOS DE TIEMPO DURANTE LOS",1,14,10,14,30,76,75);
 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,191,191,191,0,0,0,"CUALES  TODAS  LAS  OPERACIONES  DEL
SISTEMA SE LLEVAN A CABO.",
                                    "",1,14,10,14,30,58,0);
 for(c=0;c<=4;c++){
 MessageBeep(0);
 Brocha= CreateSolidBrush(RGB(255,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,27,242,76,342);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 Brocha= CreateSolidBrush(RGB(127,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,27,242,76,342);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta6);
 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LAS PUERTAS DE ENTRADA / SALIDA.",
                                    "",1,20,10,20,30,32,0);
for(c=0,c<=6,c++){
```

```
MessageBeep(0);
 SetBkColor(hdc,RGB(0,0,0));
 TextOut(hdc,430,364,"PUERTA DE",9);
 TextOut(hdc,430,380,"SALIDA",6);
 TextOut(hdc,575,364,"PUERTA DE",9);
 TextOut(hdc,575,380,"ENTRADA",7);
 TIEMPO(cuenta5);

 SetBkColor(hdc,RGB(255,255,0));
 TextOut(hdc,430,364,"PUERTA DE",9);
 TextOut(hdc,430,380,"SALIDA",6);
 TextOut(hdc,575,364,"PUERTA DE",9);
 TextOut(hdc,575,380,"ENTRADA",7);
 TIEMPO(cuenta5);
 }

 TIEMPO(cuenta6);
    VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LAS PUERTAS DE ENTRADA / SALIDA:
DURANTE LA EJECUCION DE UN PROGRAMA, LA UCP",
                                    "CONSTANTEMENTE LEE DE, O ESCRIBE EN LA
MEMORIA. EL PROGRAMA PUEDE SOLICITAR",1,20,10,20,30,76,75);
 TIEMPO(cuenta6);
VER_CADENA_CON(hwnd,255,255,0,0,0,0,"A LA UCP LEER DE UNO DE LOS DISPOSITIVOS
DE ENTRADA O BIEN ESCRIBIR EN UNO",
                                "DE LOS DISPOSITIVOS DE SALIDA.",1,14,10,14,30,74,30);

TIEMPO(cuenta6);
TIEMPO(cuenta6);
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);

SelectObject(hdc,PrevFont1);
DeleteObject(Font1);
ReleaseDC(hwnd,hdc);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_w=5;  j=7;
}  // TERMINA FUNCION MICROCOMPUTADORA 2



//-------------------------------------------------------------------------------------------------
---
//-------------------------------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "FUNCION.H"
#include "VARIOS.H"
#include "CONTROL.H"
#include "PREGUNTA.H"
#include "MODULOS.H"
#include "MICRO.H"
#include "TESIS.H"
#include "PREG_FUN.H"

#include <stdlib.h>
```

```
#include <stdio.h>

#define cuenta 40
#define cuenta1 5
#define cuenta2 70
#define cuenta3 10
#define cuenta4 100
#define cuenta5 20
#define cuenta6 200
#define cuenta7 150

extern HANDLE hInst;
extern HCURSOR Reloj_Arena;
extern int var_z,var_w,j;
extern int mensaje;// toma el valor devuelto por messagebox
extern int var_mouse;
extern int indicador; //  se utiliza para saber que variable SE NIEGA CUANDO ENTRAN LAS
EVALUACIONES
extern int numero_pregunta;
extern int con;  // se utiliza en CHECA_EVALUACION para que pida la respuesta de la sig. pregunta
extern int preg;
extern int cantidad_preguntas;
extern char respuesta;
extern char pregunta[5];
extern RECT limite;
extern float factor_ancho;
extern float factor_alto;
extern BOOL salir_evaluacion;
extern BOOL evaluacion;
extern BOOL imagen;
extern BOOL ejecucion;
extern BOOL ejecucion_1;
extern BOOL ejecucion_1_1;
extern BOOL seleccion;
extern BOOL boton_selec;
extern BOOL micro;
extern BOOL puntos;
extern HFONT Font,PrevFont,Fuente,PreFuente;
extern HDC hdcMemory,hdcMemory1,hdcCuadro,hdcSeleccion; // contextos de visualizacion de pag.
oculta
extern                                                                    HBITMAP
hbmpOld,hmbCuadro,hmbCuadro1,hbmpSeleccion,hbmpSeleccion1,hbmpMyBitmap1,hbmpOld1;


//*****************************************************************************
****
//    FUNCION APLICACION

 void APLICACION(HWND hwnd,int z){

int d=0,b;
HDC hdc;
ejecucion=FALSE;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
```

```
SetCursor(Reloj_Arena);

hbmpMyBitmap1 = LoadBitmap(hInst, "MyBitmap1");
hdcMemory1 = CreateCompatibleDC(hdc);
hbmpOld1 = SelectObject(hdcMemory1, hbmpMyBitmap1);

PatBlt(hdc,0,0,640,442,BLACKNESS);
if(limite.right==624){
 if(limite.bottom==442){
while(d<=640){
BitBlt(hdc,0,0,d,442,hdcMemory1,0,0,SRCCOPY);
d+=2; }
 } else BitBlt(hdc,0,0,640,442,hdcMemory1,0,0,SRCCOPY);}
else BitBlt(hdc,0,0,640,442,hdcMemory1,0,0,SRCCOPY);

SelectObject(hdcMemory1,hbmpOld1);
DeleteDC(hdcMemory1);

MessageBeep(-1);
TIEMPO(cuenta4);
ejecucion=TRUE;
CONTROL_GENERAL(hwnd,hdc,z); // CONTROLA LAS LLAMADAS A FUNCIONES

ReleaseDC(hwnd, hdc);

 return;
 } // TERMINA APLICACION


//*****************************************************************************
****
// FUNCION CHECA APLI


 void CHECA_APLI(HWND hwnd,int posicion_x,int posicion_y){

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
        if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
        if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto *435)){
         BOTONES(hwnd,2);
         TIEMPO(cuenta5);
         BOTONES(hwnd,1);
          mensaje= MessageBox(GetFocus(),"DESEA CANCELAR","PAUSA",MB_OKCANCEL |
MB_ICONINFORMATION);
        if(mensaje==IDOK){
        zResetFunc(hwnd);
        ejecucion=FALSE;
        seleccion=TRUE;
        SELECCION(hwnd); }}
        else    if(posicion_y>(factor_alto *349) && posicion_y<(factor_alto *390)){
                BOTONES(hwnd,4);
                 TIEMPO(cuenta5);
                BOTONES(hwnd,3);
```

```
                CONTROL_GENERAL(hwnd,hdc,var_z);}
                else if(posicion_y>(factor_alto *304) && posicion_y<(factor_alto *345)){
                    BOTONES(hwnd,6);
                        TIEMPO(cuenta5);
                    BOTONES(hwnd,5);
                    CONTROL_GENERAL(hwnd,hdc,var_z-1);}
                    }
    ReleaseDC(hwnd,hdc);

 }


//*****************************************************************************
****
//    FUNCION APLICACION 1

 void APLICACION_1(HWND hwnd){

 int punto=0,punto1=0,punto2;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
seleccion=FALSE;
ejecucion_1=TRUE;

 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);
 punto2=limite.right;

  if(puntos==TRUE){

    while(punto<=(limite.right/2)+2){

     PatBlt(hdc,0,0,punto1,limite.bottom,BLACKNESS);
     PatBlt(hdc,punto2,0,limite.right,limite.bottom,BLACKNESS);
     punto+=2;
     punto1+=2;
     punto2-=2;
     }
     }

puntos=FALSE;
 Pluma= CreatePen(PS_SOLID,1,RGB(128,255,255));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(128,255,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,0,0,640,442);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);
```

```
 Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(0,127,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,246,48,382,291);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);


FLECHAS_DATOS(hdc,2,11,3);
FLECHAS_CONTROL(hdc,1,5,3);
BOTONES_1(hwnd,11);
BOTONES_1(hwnd,1);
BOTONES_1(hwnd,3);
BOTONES_1(hwnd,5);
BOTONES_1(hwnd,7);
BOTONES_1(hwnd,9);
BOTONES(hwnd,1);

 SetTextColor(hdc,RGB(127,0,0));
 SetBkColor(hdc,RGB(191,191,191));
 TextOut(hdc,230,15,"SELECCIONE UN MODULO",20);

 ULTIMA_IMAGEN(hwnd);

ReleaseDC(hwnd,hdc);
                }


//**********************************************************************************
****
//   FUNCION CHECA APLI 1

 void CHECA_APLI_1(HWND hwnd,int posicion_x,int posicion_y){

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite right, limite bottom);

        if(posicion_x>(factor_ancho *248) && posicion_x<(factor_ancho *380)){
         if(posicion_y>(factor_alto *50) && posicion_y<(factor_alto *70)){
          BOTONES_1(hwnd,12);
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,11);
          mensaje= MessageBox(GetFocus(),"DESEA INFORMACION DE LA UNIDAD CENTRAL
DE PROCESAMIENTO (CPU O UCP)","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
         if(mensaje==IDOK){
          ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          MessageBeep(-1);
          CONTROL_MODULOS(hwnd ,hdc,1,1);}}
         else if(posicion_y>(factor_alto *86) && posicion_y<(factor_alto *159)){
          BOTONES_1(hwnd,2);
```

```
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,1);
                mensaje=    MessageBox(GetFocus(),"DESEA    INFORMACION    DE    LA
ALU","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
          ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          MessageBeep(-1);
          CONTROL_MODULOS(hwnd ,hdc,1,2);}}
          else   if(posicion_y>(factor_alto *215) && posicion_y<(factor_alto *289)){
           BOTONES_1(hwnd,4);
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,3);
                    mensaje=    MessageBox(GetFocus(),"DESEA    INFORMACION    DE
CONTROL","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
          ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          MessageBeep(-1);
          CONTROL_MODULOS(hwnd,hdc,1,3);}}
          else   if(posicion_y>(factor_alto *352) && posicion_y<(factor_alto *425)){
           BOTONES_1(hwnd,6);
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,5);
                mensaje=    MessageBox(GetFocus(),"DESEA    INFORMACION    DE    LA
MEMORIA","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
       ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          MessageBeep(-1);
          CONTROL_MODULOS( hwnd, hdc,1,4);}}}   else

        if(posicion_x>(factor_ancho *75) && posicion_x<(factor_ancho *154)){
         if(posicion_y>(factor_alto *189) && posicion_y<(factor_alto *310)){
          BOTONES_1(hwnd,8);
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,7);
                mensaje=    MessageBox(GetFocus(),"DESEA    INFORMACION    DE    LA
ENTRADA","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
          ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          CONTROL_MODULOS(hwnd,hdc,1,5);
          MessageBeep(-1);}}}   else

        if(posicion_x>(factor_ancho *484) && posicion_x<(factor_ancho *563)){
         if(posicion_y>(factor_alto *190) && posicion_y<(factor_alto *311)){
          BOTONES_1(hwnd,10);
          TIEMPO(cuenta5);
          BOTONES_1(hwnd,9);
                mensaje=    MessageBox(GetFocus(),"DESEA    INFORMACION    DE    LA
SALIDA","HELP",MB_OKCANCEL | MB_ICONINFORMATION);
          if(mensaje==IDOK){
          ejecucion_1=FALSE;
       ejecucion_1_1=TRUE;
          CONTROL_MODULOS(hwnd,hdc,1,6);
      MessageBeep(-1);}}} else
    if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
```

```
        if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto *435)){
         BOTONES(hwnd,2);
         TIEMPO(cuenta5);
         BOTONES(hwnd,1);
           mensaje=  MessageBox(GetFocus(),"DESEA  CANCELAR","PAUSA",MB_OKCANCEL |
MB_ICONINFORMATION);
         if(mensaje==IDOK){
         zResetFunc(hwnd);
         ejecucion_1=FALSE;
      seleccion=TRUE;
         SELECCION(hwnd); }}}

ReleaseDC(hwnd,hdc);

}



//*****************************************************************************
****
//  FUNCION  CHECA MODULOS

 void CHECA_MOD(HWND hwnd,int posicion_x,int posicion_y){
HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
        if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
        if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto *435)){
         BOTONES(hwnd,2);
         TIEMPO(cuenta5);
         BOTONES(hwnd,1);
           mensaje=   MessageBox(GetFocus(),"DESEA   SALIR","PAUSA",MB_OKCANCEL  |
MB_ICONINFORMATION);
         if(mensaje==IDOK){
         zResetFunc(hwnd);
         ejecucion_1_1=FALSE;
         if(micro==TRUE){
         seleccion=TRUE;
         micro=FALSE;
         SELECCION(hwnd);}
      else
        {ejecucion_1=TRUE;
         APLICACION_1(hwnd); }}}
         else   if(posicion_y>(factor_alto *349) && posicion_y<(factor_alto *390)){
                 BOTONES(hwnd,4);
                  TIEMPO(cuenta5);
                 BOTONES(hwnd,3);
                 CONTROL_MODULOS(hwnd,hdc,var_w,j);}
                 else if(posicion_y>(factor_alto *304) && posicion_y<(factor_alto *345)){
                     BOTONES(hwnd,6);
                         TIEMPO(cuenta5);
                     BOTONES(hwnd,5);
                     CONTROL_MODULOS(hwnd,hdc,var_w-1,j);}
                     }
```

```
    ReleaseDC(hwnd,hdc);
  }

//*****************************************************************************
****
//   FUNCION AYUDA

 void AYUDA(HWND hwnd){

HBRUSH Brocha, Brocha1;
HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 640,442);
SetViewportExt(hdc, limite.right, limite.bottom);

  Brocha= CreateSolidBrush(RGB(255,255,255));
  Brocha1= SelectObject(hdc,Brocha);
  Rectangle(hdc,0,0,640,442);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  ReleaseDC(hwnd,hdc);
  LECTOR_AYUDA(hwnd);
  }

//*****************************************************************************
****
//   FUNCION SELECCION

 void SELECCION(HWND hwnd){

int r,d1=0,d2=320;
HBRUSH Brocha, Brocha1;
HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
seleccion=TRUE;

if(boton_selec==FALSE){
hbmpSeleccion = LoadBitmap(hInst, "MyBitmap3");
hdcSeleccion = CreateCompatibleDC(hdc);
hbmpSeleccion1 = SelectObject(hdcSeleccion, hbmpSeleccion);}

if(imagen==TRUE){

for(r=0;r<=255;r=r+3){
Brocha= CreateSolidBrush(RGB(255-r,255,255));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,0,0,640,442);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
}
```

```
for(r=0;r<=255;r=r+3){
Brocha= CreateSolidBrush(RGB(0,255-r,255));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,0,0,640,442);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
}

for(r=0;r<=255;r=r+3){
Brocha= CreateSolidBrush(RGB(0,0,255-r));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,0,0,640,442);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
}

TIEMPO(cuenta2);
PatBlt(hdc,0,0,640,442,BLACKNESS);
if(limite.right==624){
 if(limite.bottom==442){
while(d1<=320){
BitBlt(hdc,320,0,d1,442,hdcSeleccion,320,0,SRCCOPY);
BitBlt(hdc,d2,0,d1,442,hdcSeleccion,d2,0,SRCCOPY);
d1+=2;
d2-=2; } } else BitBlt(hdc,0,0,640,442,hdcSeleccion,0,0,SRCCOPY);}
else BitBlt(hdc,0,0,640,442,hdcSeleccion,0,0,SRCCOPY);
} else BitBlt(hdc,0,0,640,442,hdcSeleccion,0,0,SRCCOPY); //  de la decisión
boton_selec=TRUE;
BOTON_SELECCION(hdc,1,33,161,198,307);
BOTON_SELECCION(hdc,1,238,161,404,307);
BOTON_SELECCION(hdc,1,442,161,604,307);
BOTONES(hwnd,1);

imagen=FALSE;
ULTIMA_IMAGEN(hwnd);
ReleaseDC(hwnd,hdc);
                                    }


//*****************************************************************************
****
//    FUNCION CHECA SELECCION

 void CHECA_SELEC(HWND hwnd,int posicion_x,int posicion_y){

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);

          if(posicion_y>(factor_alto *161) && posicion_y<(factor_ancho *307)){
            if(posicion_x>(factor_ancho *33) && posicion_x<(factor_ancho *198)){
            BOTON_SELECCION(hdc,4,33,161,198,307);
             TIEMPO(cuenta5);
            BOTON_SELECCION(hdc,1,33,161,198,307);
```

```
            seleccion=FALSE;
            ejecucion_1_1=TRUE;
            micro=TRUE;
            CONTROL_MODULOS(hwnd,hdc,1,7);}
            else if(posicion_x>(factor_ancho *238) && posicion_x<(factor_ancho *404)){
            BOTON_SELECCION(hdc,2,238,161,404,307);
             TIEMPO(cuenta5);
            BOTON_SELECCION(hdc,1,238,161,404,307);
          seleccion=FALSE;
            ejecucion_1=TRUE;
          APLICACION_1(hwnd);}
            else if(posicion_x>(factor_ancho *442) && posicion_x<(factor_ancho *604)){
            BOTON_SELECCION(hdc,3,442,161,604,307);
             TIEMPO(cuenta5);
            BOTON_SELECCION(hdc,1,442,161,604,307);
          seleccion=FALSE;
            ejecucion=TRUE;
            var_z=1;
            APLICACION(hwnd,var_z);}}
           else
           if(posicion_x>(factor_ancho *555) && posicion_x<(factor_ancho *623)){
           if(posicion_y>(factor_alto *394) && posicion_y<(factor_alto *435)){
            BOTONES(hwnd,2);
             TIEMPO(cuenta5);
            BOTONES(hwnd,1);
              mensaje=  MessageBox(GetFocus(),"DESEA  CANCELAR","EXIT",MB_OKCANCEL |
MB_ICONINFORMATION);
            if(mensaje==IDOK){
          seleccion=FALSE;
          SelectObject(hdcMemory,hbmpOld);   /* ...deseleccionarlo */
          SelectObject(hdcCuadro,hmbCuadro1);
          SelectObject(hdcSeleccion,hbmpSeleccion1);

          DeleteObject(hmbCuadro);  /* ...borrar los mapas de bits ocultos... */
          DeleteDC(hdcMemory);          /* ...borrar el DC en memoria... */
          DeleteDC(hdcCuadro);
          DeleteDC(hdcSeleccion);

          PostQuitMessage(0);}}}

            ReleaseDC(hwnd,hdc);
            }


//*****************************************************************************
****
// FUNCION EVALUACION
 void EVALUACION(HWND hwnd){

  HBRUSH Brocha, Brocha1;
HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
  if(ejecucion_1_1==TRUE){ejecucion_1_1=FALSE; indicador=1;}
```

```
if(ejecucion==TRUE){ejecucion=FALSE; indicador=2 ;}
evaluacion=TRUE;
salir_evaluacion=FALSE; //  se niega para que al entrar a checa evaluacion no tenga ningun efecto
con=1;  // se utiliza en CHECA_EVALUACION para que pida la respuesta de la sig. pregunta
preg=0; // indica en el arreglo el numero de pregunta
numero_pregunta=1; // indica cuando se llego a la ultima pregunta

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,0,0,640,442);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
Brocha= CreateSolidBrush(RGB(0,127,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,80,370,410,442);
Select(Object(hdc,Brocha1);
DeleteObject(Brocha);

BOTONES_EVALUACION(hwnd,1);
BOTONES_EVALUACION(hwnd,3);
BOTONES_EVALUACION(hwnd,5);
BOTONES_EVALUACION(hwnd,7);
BOTONES_EVALUACION(hwnd,9);
SetBkColor(hdc,RGB(0,0,0));
SetTextColor(hdc,RGB(0,255,0));
TextOut(hdc,263,10,"EVALUACION",10);
SetBkMode(hdc,TRANSPARENT);
SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,170,375,"RESPONDA PREGUNTA 1",19);
ReleaseDC(hwnd,hdc);
var_w=3;  j=5;
}


//*****************************************************************************
****
//   FUNCION CHECA evaluacion

void CHECA_EVALUACION(HWND hwnd,int posicion_x,int posicion_y){

HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;

HDC hdc;
GetClientRect(hwnd, &limite);
hdc=GetDC(hwnd);
SetMapMode(hdc, MM_ANISOTROPIC);
SetWindowExt(hdc, 624,442);
SetViewportExt(hdc, limite.right, limite.bottom);
        if(posicion_y>(factor_alto *395) && posicion_y<(factor_alto *435)){
         if(posicion_x>(factor_ancho *100) && posicion_x<(factor_ancho *150)){
          BOTONES_EVALUACION(hwnd,2);
          TIEMPO(cuenta5);
          BOTONES_EVALUACION(hwnd,1);
           respuesta='a'; }
        else if(posicion_x>(factor_ancho *160) && posicion_x<(factor_ancho *210)){
         BOTONES_EVALUACION(hwnd,4);
         TIEMPO(cuenta5);
```

```
    BOTONES_EVALUACION(hwnd,3);
    respuesta='b'; }
    else   if(posicion_x>(factor_ancho    *220)   &&    posicion_x<(factor_ancho    *270)){
    BOTONES_EVALUACION(hwnd,6);
    BOTONES_EVALUACION(hwnd,6);
    TIEMPO(cuenta5);
    BOTONES_EVALUACION(hwnd,5);
    respuesta='c';}
    else   if(posicion_x>(factor_ancho *280) && posicion_x<(factor_ancho *330)){
    BOTONES_EVALUACION(hwnd,8);
    TIEMPO(cuenta5);
    BOTONES_EVALUACION(hwnd,7);
    respuesta='d';}
    else if(posicion_x>(factor_ancho *340) && posicion_x<(factor_ancho *400)){
    BOTONES_EVALUACION(hwnd,10);
    TIEMPO(cuenta5);
    BOTONES_EVALUACION(hwnd,9);
    salir_evaluacion=FALSE;
    switch(indicador){
     case 1: ejecucion_1_1=TRUE; break;
     case 2: ejecucion=TRUE; break;
  }
    evaluacion=FALSE;
    PatBlt(hdc,0,0,645,445,BLACKNESS);
    Brocha= CreateSolidBrush(RGB(127,0,0));
BrochaI= SelectObject(hdc,Brocha);
Rectangle(hdc,120,120,520,320);
SelectObject(hdc,BrochaI);
DeleteObject(Brocha);
SetTextColor(hdc,RGB(191,191,191));
SetBkColor(hdc,RGB(0,0,0));
    TextOut(hdc,140,150,"SI ELIGE ATRAS REGRESARA A LA EVALUACION.",41);
    TextOut(hdc,140,180,"SI ELIGE AVANCE PASARA A LA SIGIENTE SESION",43);
    TextOut(hdc,140,210,"CORRESPONDIENTE A ESTE DISPOSITIVO.",35);
    TextOut(hdc,140,240,"SI ELIGE SALIR REGRESA AL ULTIMO MENU.",38);
BOTONES(hwnd,1);
BOTONES(hwnd,3);
   BOTONES(hwnd,5); }}
  SetBkMode(hdc,TRANSPARENT);
  if(cantidad_preguntas>=numero_pregunta){
if(salir_evaluacion==TRUE){
    if(pregunta[preg]==respuesta){
      MessageBeep(0);
      TextOut(hdc,370,375,"BIEN",4),}
      else {MessageBeep(-1);
            MessageBeep(-1);
            MessageBeep(-1);
            MessageBeep(-1);
            MessageBeep(-1);
            MessageBeep(-1);
            TextOut(hdc,370,375,"MAL",3);
            con--;
     preg--;
       numero_pregunta--;
       }

          TIEMPO(cuenta4);
    Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
```

```
            Pluma1= SelectObject(hdc,Pluma);
                  Brocha= CreateSolidBrush(RGB(0,127,0));
            Brocha1= SelectObject(hdc,Brocha);
            Rectangle(hdc,100,373,408,393);
            SelectObject(hdc,Brocha1);
            DeleteObject(Brocha);
                  SelectObject(hdc,Pluma1);
            DeleteObject(Pluma1);

             if(cantidad_preguntas-1>=numero_pregunta){
              switch(con){
              case 0:
                   TextOut(hdc,170,375,"RESPONDA PREGUNTA 1",19);
               break;
              case 1:
                   TextOut(hdc,170,375,"RESPONDA PREGUNTA 2",19);
               break;
              case 2:
                   TextOut(hdc,170,375,"RESPONDA PREGUNTA 3",19);
               break;
              case 3:
                   TextOut(hdc,170,375,"RESPONDA PREGUNTA 4",19);
               break;
              case 4:
                   TextOut(hdc,170,375,"RESPONDA PREGUNTA 5",19);
               break;
              } } else TextOut(hdc,170,375,"FIN DE LA EVALUACION.",21);
              con++;
              preg++;
              numero_pregunta++;} }

salir_evaluacion=TRUE;
ULTIMA_IMAGEN(hwnd);
ReleaseDC(hwnd,hdc);
}



//*****************************************************************************
****
//  FUNCION CONTROL GENERAL

 void CONTROL_GENERAL(HWND hwnd,HDC hdc,int w){

switch(w){
case 1:
MODULO_1(hwnd,hdc); break;
case 2:
MODULO_2(hwnd,hdc); break;
case 3:
EVALUACION(hwnd);
EVALUACION_MODULO_1_2(hwnd); break;
case 4:
MODULO_3(hwnd,hdc); break;
case 5:
MODULO_4(hwnd,hdc); break;
case 6:
EVALUACION(hwnd);
```

```
EVALUACION_MODULO_3_4(hwnd); break;
case 7:
EVALUACION(hwnd);
EVALUACION_MODULO_3_4_1(hwnd); break;
case 8:
MODULO_5(hwnd,hdc); break;
case 9:
MODULO_6(hwnd,hdc); break;
case 10:
EVALUACION(hwnd);
EVALUACION_MODULO_5_6(hwnd); break;
case 11:
MODULO_7(hwnd,hdc); break;
case 12:
MODULO_8(hwnd,hdc); break;
case 13:
EVALUACION(hwnd);
EVALUACION_MODULO_7_8(hwnd); break;
case 14:
MODULO_9(hwnd,hdc); break;
case 15:
MODULO_10(hwnd,hdc); break;
case 16:
EVALUACION(hwnd);
EVALUACION_MODULO_9_10(hwnd); break;
  }
  } // FIN DE LA FUNCION



//**********************************************************************************
****
// FUNCION CONTROL MODULOS

void CONTROL_MODULOS(HWND hwnd,HDC hdc,int w,int j){

switch(j){
case 1:
 switch(w){
  case 1:
   MOD_CPU_1(hwnd,hdc); break;
  case 2:
   EVALUACION(hwnd);
   EVALUACION_CPU_1(hwnd); break;
  case 3:
   MOD_CPU_2(hwnd,hdc); break;
  case 4:
   EVALUACION(hwnd);
   EVALUACION_CPU_2(hwnd); break;
   } break;
case 2:
 switch(w){
  case 1:
   MOD_ALU(hwnd,hdc); break;
  case 2:
   EVALUACION(hwnd);
   EVALUACION_ALU_1(hwnd); break;
  case 3:
```

```
    EVALUACION(hwnd);
    EVALUACION_ALU_2(hwnd); break;
    } break;
  case 3:
   switch(w){
   case 1:
    MOD_CONTROL(hwnd,hdc); break;
   case 2:
    EVALUACION(hwnd);
    EVALUACION_CONTROL(hwnd); break;
    } break;
  case 4:
   switch(w){
   case 1:
    MOD_MEMORIA_1(hwnd,hdc); break;
   case 2:
    EVALUACION(hwnd);
    EVALUACION_MEMORIA_1(hwnd); break;
   case 3:
    MOD_MEMORIA_2(hwnd,hdc); break;
   case 4:
    EVALUACION(hwnd);
    EVALUACION_MEMORIA_2(hwnd); break;
   case 5:
    MOD_MEMORIA_3(hwnd,hdc); break;
   case 6:
    EVALUACION(hwnd);
    EVALUACION_MEMORIA_3(hwnd); break;
   case 7:
    EVALUACION(hwnd);
    EVALUACION_MEMORIA_4(hwnd); break;
    } break;
  case 5:
   switch(w){
   case 1:
    MOD_ENTRADA(hwnd,hdc); break;
   case 2:
    EVALUACION(hwnd);
    EVALUACION_ENTRADA(hwnd); break;
    } break;
  case 6:
   switch(w){
   case 1:
    MOD_SALIDA(hwnd,hdc); break;
   case 2:
    EVALUACION(hwnd);
    EVALUACION_SALIDA(hwnd); break;
    } break;
  case 7:
   switch(w){
   case 1:
    MICROCOMPUTADORA_1(hwnd); break;
   case 2:
    EVALUACION(hwnd);
    EVALUACION_MICRO_1(hwnd); break;
   case 3:
    EVALUACION(hwnd);
    EVALUACION_MICRO_2(hwnd); break;
```

```
  case 4:
   MICROCOMPUTADORA_2(hwnd); break;
  case 5:
   EVALUACION(hwnd);
   EVALUACION_MICRO_3(hwnd); break;
  case 6:
   EVALUACION(hwnd);
   EVALUACION_MICRO_4(hwnd); break;
   } break;

}}


//-------------------------------------------------------------------------------------------------
---
//-------------------------------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "VARIOS.H"
#include "TESIS.H"
extern int cantidad_preguntas;
extern int var_z,var_w,j;
extern char pregunta[5],respuesta;
extern RECT limite;
extern HFONT Fuente,PreFuente;


//**********************************************************************************************
****
// FUNCION EVALUACION MEMORIA 1

 void EVALUACION_MEMORIA_1(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) LA FUNCION DE LA UNIDAD DE MEMORIA ES:",41);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) ENVIAR SEÑALES DE CONTROL.",29);
  TextOut(hdc,330,50,"b) ALMACENAR DIGITOS BINARIOS.",30);
  TextOut(hdc,20,70,"c) RECUPERAR INFORMACION DE ENTRADA.",36);
  TextOut(hdc,330,70,"d) LEER DATOS O INFORMACION.",28);
  pregunta[0]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) SIRVE COMO ALMACENAMIENTO INTERMEDIO O FINAL DE
OPER.ARITMETICAS:",68);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) EL MICROPROCESADOR.",22);
  TextOut(hdc,315,120,"b) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,20,140,"c) LA UNIDAD DE MEMORIA.",25);
  TextOut(hdc,330,140,"d) LA ALU O UAL.",16);
  pregunta[1]='c';
```

```
cantidad_preguntas=2;
ULTIMA_IMAGEN(hwnd);
var_w=3; j=4;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//  FUNCION EVALUACION MEMORIA 2

void EVALUACION_MEMORIA_2(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) LA PRINCIPAL DIFERENCIA ENTRE LOS SISTEMAS DIGITALES Y LOS
ANALOGICOS ES:",76);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) EL ALMACENAMIENTO MACIVO DE INFORMACION.",43);
  TextOut(hdc,390,50,"b) LA VELOCIDAD.",17);
  TextOut(hdc,20,70,"c) EL TAMAÑO.",13);
  TextOut(hdc,290,70,"d) LA FACIL ADAPTACION A DISTINTAS SITUACIONES.",47);
 pregunta[0]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) UN SISTEMA DE COMPUTACION UTILIZA NORMALMENTE:",49);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) 2 MEORIAS DE ALTA VELOCIDAD.",31);
  TextOut(hdc,330,120,"b) 2 MEMORIAS DE ALMACENAMIENTO EN MASA.",40);
  TextOut(hdc,20,140,"c) 1 MEMORIA DE ALTA VELOCIDAD Y UNA MAS LENTA.",48);
  TextOut(hdc,400,140,"d) NINGUNA MEMORIA.",19);
 pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LA RAM ES:",13);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) MEMORIA DE ALTA VELOCIDAD.",29);
  TextOut(hdc,330,190,"b) MEMORIA EN MASA.",19);
  TextOut(hdc,20,210,"c) NO ES UNA MEMORIA.",21);
  TextOut(hdc,330,210,"d) DISPOSITIVO DE ENTRADA.",26);
 pregunta[2]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) TIENE LA CAPACIDAD DE ALMACENAR MILLONES DE BITS SIN
ENERGIA:",64);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) MEMORIA DE ALTA VELOCIDAD.",29);
  TextOut(hdc,350,260,"b) UNIDAD DE SALIDA.",20);
  TextOut(hdc,20,280,"c) UNIDAD DE CONTROL.",21);
  TextOut(hdc,330,280,"d) MEMORIA DE ALMACENAMIENTO EN MASA.",37);
 pregunta[3]='a';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_w=5; j=4;
```

```
ReleaseDC(hwnd,hdc);
}

//**********************************************************************************
****
// FUNCION EVALUACION MEMORIA 3

void EVALUACION_MEMORIA_3(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) EN CUALQUIER SISTEMA LOS PRINCIPIOS DE OPERACION DE LA
MEMORIA:",66);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) SON DIFERENTES.",18);
  TextOut(hdc,370,50,"b) SON LOS MISMOS.",18);
  TextOut(hdc,20,70,"c) -------",10);
  TextOut(hdc,310,70,"d) -------",10);
 pregunta[0]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) UNA MEMORIA DE 32 X 4 ALMACENA:",33);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) 32 PALABRAS DE 4 BITS CADA UNA.",34);
  TextOut(hdc,330,120,"b) 4 PALABRAS DE 32 BITS CADA UNA.",34);
  TextOut(hdc,20,140,"c) 128 PALABRAS.",16);
  TextOut(hdc,400,140,"d) 128 BITS.",12);
 pregunta[1]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) PARA ALMACENAR Y RECUPERAR DE LA MEMORIA PALABRAS DE 4
BITS REQUERIMOS:",74);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) 2 LINEAS DE ENTRADA DE DATOS Y 2 DE SALIDA DE DATOS.",55);
  TextOut(hdc,450,190,"b) 8 LINEAS DE ENTRADA.",23);
  TextOut(hdc,20,210,"c) 8 LINEAS DE SALIDA.",22);
  TextOut(hdc,230,210,"d) 4 LINEAS DE ENTRADA DE DATOS Y 4 DE SALIDA DE DATOS.",55);
 pregunta[2]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) EN UNA OPERACION DE ESCRITURA SE INVOLUCRA:",46);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) A LAS LINEAS DE SALIDA DE DATOS.",36);
  TextOut(hdc,350,260,"b) A LAS LINEAS DE ENT/SAL.",27);
  TextOut(hdc,20,280,"c) A LAS LINEA DE ENTRADA DE DATOS.",37);
  TextOut(hdc,330,280,"d) NO SE INVOLUCRAN LAS LINEAS.",31);
 pregunta[3]='c';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_w=7, j=4;
ReleaseDC(hwnd,hdc);
}
```

```
//*****************************************************************************
****
//  FUNCION EVALUACION MEMORIA 4

 void EVALUACION_MEMORIA_4(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) EN UNA OPERACION DE LECTURA SE INVOLUCRA:",44);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) A LAS LINEAS DE ENTRADA DE DIRECCION.",40);
  TextOut(hdc,370,50,"b) NINGUNA LINEA.",17);
  TextOut(hdc,20,70,"c) A TODAS LAS LINEAS.",22);
  TextOut(hdc,310,70,"d) A LAS LINEAS DE SALIDA DE DATOS.",35);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) PARA UNA MEMORIA DE 32 LOCALIDADES REQUERIMOS DE:",52);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) DE 32 BITS PARA ACCESARLAS.",30);
  TextOut(hdc,330,120,"b) DE 5 BITS PARA ACCESARLAS.",29);
  TextOut(hdc,20,140,"c) DE 16 BITS PARA ACCESARLAS.",30);
  TextOut(hdc,400,140,"d) NO REQUERIMOS BITS.",22);
 pregunta[1]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) PARA ACCESAR ALGUNA DE LAS 32 LOCALIDADES DE MEMORIA
UTILIZAMOS A:",70);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) LAS LINEAS DE ENTRADA DE DATOS.",34);
  TextOut(hdc,420,190,"b) LAS LINEAS DE SALIDA DE DATOS.",33);
  TextOut(hdc,20,210,"c) NIGUNA LINEA.",16);
  TextOut(hdc,230,210,"d) LA ENTRADA DE DIRECCIONES.",29);
 pregunta[2]='d';

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_w=8; j=4;
ReleaseDC(hwnd hdc).
}

//*****************************************************************************
****
//  FUNCION EVALUACION MODULO UNIDAD DE CONTROL

 void EVALUACION_CONTROL(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1)  SE  ENCARGAN  DE  DECODIFICAR  LA  PALABRA  DE
INSTRUCCION:",56);
```

```
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LOS REGISTROS.",17);
  TextOut(hdc,370,50,"b) LAS LINEA DE ENTRADA.",24);
  TextOut(hdc,20,70,"c) LOS CIRCUITOS LOGICOS.",25);
  TextOut(hdc,310,70,"d) LAS LINEAS DE SALIDA.",23);
 pregunta[0]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) BUSCA Y TRAE UNA INSTRUCCION DE LA MEMORIA:",46);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) LA UNIDAD DE SALIDA.",23);
  TextOut(hdc,330,120,"b) LA UNIDAD DE CONTROL.",24);
  TextOut(hdc,20,140,"c) LA MEMORIA.",14);
  TextOut(hdc,400,140,"d) LA UNIDAD ARIT. /LOGICA.",27);
 pregunta[1]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) PARA BUSCAR Y TRAER UNA INSTRUCCION DE LA MEMORIA
REQUERIMOS DE:",67);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) UNA INSTRUCCION Y UN COMANDO DE LECTURA.",43);
  TextOut(hdc,420,190,"b) UNA DIRECCION.",17);
  TextOut(hdc,20,210,"c) UN COMANDO DE LECTURA.",25);
  TextOut(hdc,230,210,"d) NO SE REQUIERE NADA",22);
 pregunta[2]='a';

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_w=3; j=3;
ReleaseDC(hwnd,hdc);
}


//**********************************************************************************
****
//  FUNCION EVALUACION UNIDAD DE ENTRADA

 void EVALUACION_ENTRADA(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) CUAL ES LA FUNCION DE LA UNIDAD DE ENTRADA.",46);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) ENVIAR DATOS AL EXTERIOR.",28);
  TextOut(hdc,330,50,"b) GRABAR DATOS DE LA MEMORIA.",30);
  TextOut(hdc,20,70,"c) INTRODUCIR DATOS DEL EXTERIOR.",33);
  TextOut(hdc,330,70,"d) REALIZAR OPERACIONES ARITMETICAS.",36);
 pregunta[0]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) LA UNIDAD DE ENTRADA ES:",27);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) EXCLISIVAMENTE UN TECLADO.",29);
  TextOut(hdc,330,120,"b) ES UN DISPOSITIVO INTERIOR DE LA PC.",39);
  TextOut(hdc,20,140,"c) DISPOSITIVO QUE PROPORCIONA ENTRADA.",38);
```

```
  TextOut(hdc,330,140,"d) ES UNA CABLE QUE TRANSMITE DATOS.",36);
  pregunta[1]='c';

 SetTextColor(hdc,RGB(0,0,0));
 TextOut(hdc,10,170,"3) COMO LLEGA LA ENTRADA A LA MAQUINA (PC):",43);
   SetTextColor(hdc,RGB(127,0,0));
   TextOut(hdc,20,190,"a) COMO SEÑALES ATRAVEZ DE UN BUS",33);
   TextOut(hdc,330,190,"b) EN PALABRAS DEL LENGUAJE COTIDIANO.",38);
   TextOut(hdc,20,210,"c) EN CODIGO ARABIGO.",21);
   TextOut(hdc,330,210,"d) COMO UNA SECUENCIA DE SIMBOLOS.",34);
  pregunta[2]='a';

 cantidad_preguntas=3;
 ULTIMA_IMAGEN(hwnd);
 var_w=3;  j=5;
 ReleaseDC(hwnd,hdc);
 }


 //**********************************************************************************
 ****
 //  FUNCION EVALUACION UNIDAD DE SALIDA

 void EVALUACION_SALIDA(HWND hwnd){

 HDC hdc;
 hdc=GetDC(hwnd);
 SetBkMode(hdc,TRANSPARENT);

 SetTextColor(hdc,RGB(0,0,0));
 TextOut(hdc,10,30,"1) QUIEN CONTROLA A LA UNIDAD DE SALIDA.",40);
   SetTextColor(hdc,RGB(127,0,0));
   TextOut(hdc,20,50,"a) LA INIDAD DE MEMORIA.",24);
   TextOut(hdc,330,50,"b) LA UNIDAD DE CONTROL.",24);
   TextOut(hdc,20,70,"c) LA UNIDAD DE ENTRADA.",24);
   TextOut(hdc,330,70,"d) LA UNIDAD ARITMETICO/LOGICA.",31);
  pregunta[0]='b';

 SetTextColor(hdc,RGB(0,0,0));
 TextOut(hdc,10,100,"2) CUAL ES LA FUNCION DE LA UNIDAD DE SALIDA.",45);
   SetTextColor(hdc,RGB(127,0,0));
   TextOut(hdc,20,120,"a) ENVIAR DATOS AL EXTERIOR.",28);
   TextOut(hdc,330,120,"b) GRABAR DATOS DE LA MEMORIA.",30);
   TextOut(hdc,20,140,"c) INTRODUCIR DATOS DEL EXTERIOR.",33);
   TextOut(hdc,330,140,"d) REALIZAR OPERACIONES ARITMETICAS.",36);
  pregunta[1]='a';


 SetTextColor(hdc,RGB(0,0,0));
 TextOut(hdc,10,170,"3) LA UNIDAD DE SALIDA ES UNICAMENTE:",37);
   SetTextColor(hdc,RGB(127,0,0));
   TextOut(hdc,20,190,"a) LA IMPRESORA.",16);
   TextOut(hdc,330,190,"b) LA PANTALLA.",15);
   TextOut(hdc,20,210,"c) ALGUN DISCO.",15);
   TextOut(hdc,280,210,"d) CUALQUIER DISPOSITIVO QUE TRANSMITA SALIDA.",46);
  pregunta[2]='d';

 cantidad_preguntas=3;
```

```
ULTIMA_IMAGEN(hwnd);
var_w=3; j=6;
ReleaseDC(hwnd,hdc);
}



//*****************************************************************************
****
// FUNCION EVALUACION CPU 1

void EVALUACION_CPU_1(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) LA UNIDAD CENTRAL DE PROCESAMIENTO (CPU) ESTA
CONSTITUIDO POR:",65);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LA UNIDAD DE ENTRADA. Y SALIDA.",34);
  TextOut(hdc,330,50,"b) POR LA ALU Y LA U.DE MEMORIA.",32);
  TextOut(hdc,20,70,"c) POR LA U. DE ENTRADA Y LA ALU.",33);
  TextOut(hdc,330,70,"d) POR LA U. DE CONTROL Y LA ALU.",33);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) AL INTEGRADO QUE CONTIENE AL CPU SE LE CONOCE COMO:",54);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) MICROPROCESADOR.",19);
  TextOut(hdc,330,120,"b) MICROCOMPUTADORA.",20);
  TextOut(hdc,20,140,"c) CPU.",7);
  TextOut(hdc,330,140,"d) CEREBRO.",11);
 pregunta[1]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LAS CARACTERISTICAS DE UNA MICROCOMPUTADORA ESTAN
DADAS POR:",63);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) LA UNIDAD ARITMETICO / LOGICA.",33);
  TextOut(hdc,330,190,"b) EL MICRPPROCESADOR.",22);
  TextOut(hdc,20,210,"c) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,330,210,"d) EL SISTEMA OPERATIVO.",24);
 pregunta[2]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) DE LAS SIGUIENTES FUNCIONES CUAL NO REALIZA EL
MICROPROCESADOR:",66);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) OFRECER SEÑALES DE DISTRIB. Y CONTROL.",41);
  TextOut(hdc,350,260,"b) TRANSFERENCIA DE DATOS.",26);
  TextOut(hdc,20,280,"c) GRABAR DATOS.",16);
  TextOut(hdc,330,280,"d) DECODIFICACION DE INSTRUCCIONES.",24);
 pregunta[3]='c';

cantidad_preguntas=4;
```

```
ULTIMA_IMAGEN(hwnd);
var_w=3; j=1;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
// FUNCION EVALUACION CPU 2

void EVALUACION_CPU_2(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) EL MICROPROCESADOR LO PODEMOS DIVIDIR EN 3 AREAS QUE
SON:",60);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) U.ENTRADA, U.SALIDA, U.DE CONTROL.",37);
  TextOut(hdc,330,50,"b) ALU, CPU O UCP ,U. DE MEMORIA.",33);
  TextOut(hdc,20,70,"c) SECCION. REGISTROS, ALU ,CPU.",32);
  TextOut(hdc,330,70,"d) ALU, S.REGISTROS, S.CONTROL Y TEMP..",39);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) BUSCA, TRAE Y DECODIFICA INSTRUCCIONES DE LA
MEMORIA:",56);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) EL MICROPROCESADOR.",22);
  TextOut(hdc,330,120,"b) SECCION DE REGISTROS.",24);
  TextOut(hdc,20,140,"c) SECCION DE CONTROL Y TEMPORALIZACION.",40);
  TextOut(hdc,400,140,"d) ALU.",7);
 pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) SU FUNCION ES LA TRANSFERENCIA DE INFORMACION ENTRE
REGISTROS:",65);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) LA UNIDAD ARITMETICO / LOGICA.",33);
  TextOut(hdc,330,190,"b) EL MICR0PPROCESADOR.",23);
  TextOut(hdc,20,210,"c) SECCION DE REGISTROS.",24);
  TextOut(hdc,330,210,"d) SECCION DE CONT. Y TEMP.",27);
 pregunta[2]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) SU FUNCION ES REALIZAR OPERACIONES ARITMETICO /
LOGICAS:",59);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) LA UNIDAD ARITMETICO/LOGICA (ALU).",37);
  TextOut(hdc,350,260,"b) UNIDAD DE SALIDA.",20);
  TextOut(hdc,20,280,"c) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,330,280,"d) SECCION DE REGISROS.",23);
 pregunta[3]='a';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
```

```
var_w=5; j=1;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//  FUNCION EVALUACION UNIDAD ARITMETICO / LOGICA 1 (ALU 1)

 void EVALUACION_ALU_1(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) LA FUNCION DE LA UNIDAD ARITMETICO /LOGICA (ALU) ES:",55);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) REALIZAR OPERACIONES CON DATOS BINARIOS.",43);
  TextOut(hdc,380,50,"b) GRABAR DATOS.",16);
  TextOut(hdc,20,70,"c) MANDAR INSTRUCCIONES.",24);
  TextOut(hdc,330,70,"d) RECIBIR Y MANDAR DATOS O INFORMACION.",40);
 pregunta[0]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) QUIEN MANDA LAS INSTRUCCIONES A LA ALU:",42);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) EL MICROPROCESADOR.",22);
  TextOut(hdc,330,120,"b) LA UNIDAD DE ENTRADA.",24);
  TextOut(hdc,20,140,"c) LA UNIDAD DE CONTROL.",24);
  TextOut(hdc,330,140,"d) LA UNIDAD DE SALIDA.",23);
 pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LA ALU CONTIENE POR LO MENOS DOR REGISTROS, LOS CUALES
SON:",62);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) REGISTRO DE ENTRADA Y SALIDA.",32);
  TextOut(hdc,330,190,"b) REGISTRO ACUMULADOR Y REGISTRO B",35);
  TextOut(hdc,20,210,"c) REGISTRO DE INSTRUCCION Y R.B",32);
  TextOut(hdc,330,210,"d) REGISTRO DE ENCABEZADO Y R.B.",32);
 pregunta[2]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,260,"OPRIMA AVANCE PARA LAS SIGUIENTES PREGUNTAS",43);

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_w=3; j=2;
ReleaseDC(hwnd,hdc);
}

//*****************************************************************************
****
//  FUNCION EVALUACION ALU 2

 void EVALUACION_ALU_2(HWND hwnd){
```

```
HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,'TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) LOS CIRCUITOS LOGICOS SE ENCUENTRAN CONTENIDOS EN:",53);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) LA UNIDAD DE ENTRADA.",25);
  TextOut(hdc,330,50,"b) LA UNIDAD DE MEMORIA.",24);
  TextOut(hdc,20,70,"c) LA SECCION REGISTROS.",25);
  TextOut(hdc,330,70,"d) LA UNIDAD ARITMETICO / LOGICA (ALU).",39);
 pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2)  LOS  CIRCUITOS  LOGICOS  SON  UNA  SECCION  ESPECIAL
ENCARGADA DE:",63);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) OPERACIONES DE LECTURA.",26);
  TextOut(hdc,345,120,"b) OPERACIONES ARITMETICAS Y LOGICAS.",37);
  TextOut(hdc,20,140,"c) MANDAR SEÑALES DE CONTROL.",29);
  TextOut(hdc,345,140,"d) OPEACIONES DE ESCRITURA.",27);
 pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) EL RESULTADO DE LAS OPERACIONES DE LA ALU SE MANDA
A:",56);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) EL REGISTRO ACUMULADOR O A LA MEMORIA.",41);
  TextOut(hdc,350,190,"b) AL CPU.",10);
  TextOut(hdc,20,210,"c) UNIDAD DE SALIDA.",20);
  TextOut(hdc,330,210,"d) UNIDAD DE ENTRADA.",21);
 pregunta[2]='a';

cantidad_preguntas=3;
ULTIMA_IMAGEN(hwnd);
var_w=4;j=2;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//  FUNCION EVALUACION MICROCOMPUTADORA I

void EVALUACION_MICRO_I(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) QUE UNIDADES CONTIENE EL CPU:",32);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) ALU Y U. CONTROL.",20);
  TextOut(hdc,290,50,"b) UIDADES ENT/SAL.",20);
  TextOut(hdc,20,70,"c) ALU Y U. MEMORIA.",20);
  TextOut(hdc,290,70,"d) U. CONTROL Y U. SALIDA.",27);
 pregunta[0]='a';
```

```
SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) EL MICROPROCESADOR ES:",25);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) EL CPU O UCP.",16);
  TextOut(hdc,330,120,"b) LA ALU.",10);
  TextOut(hdc,20,140,"c) LA ROM.",10);
  TextOut(hdc,330,140,"d) LA RAM.",10);
 pregunta[1]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LA RAM Y LA ROM SE ENCUENTRAN EN:",36);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) EL MICROPROCESADOR.",22);
  TextOut(hdc,330,190,"b) EN LA UNID.ARIT./LOGICA.",27);
  TextOut(hdc,20,210,"c) LA UNIDAD DE MEMORIA.",24);
  TextOut(hdc,330,210,"d) LA UNIDAD DE ENTRADA.",24);
 pregunta[2]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) ALMACENA PROGRAMAS Y DATOS QUE VARIAN DURANTE LA
OPERACION DEL PROGRAMA:",76);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) UNIDAD ARITMETICO / LOGICA.",30);
  TextOut(hdc,330,260,"b) LA RAM.",10);
  TextOut(hdc,20,280,"c) UNIDAD DE CONTROL.",21);
  TextOut(hdc,330,280,"d) LA ROM.",10);
 pregunta[3]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,310,"5) ALMACENA PROGRAMAS Y DATOS QUE NO VARIAN:",44);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,330,"a) UNIDAD ARITMETICO / LOGICA.",30);
  TextOut(hdc,330,330,"b) LA RAM.",10);
  TextOut(hdc,20,350,"c) UNIDAD DE CONTROL.",21);
  TextOut(hdc,330,350,"d) LA ROM.",10);
 pregunta[4]='d';

cantidad_preguntas=5;
ULTIMA_IMAGEN(hwnd);
var_w=3; j=7;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//  FUNCION EVALUACION MICROCOMPUTADORA 2

 void EVALUACION_MICRO_2(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) MEMORIA DE ACCESO ALEATORIO.",31);
  SetTextColor(hdc,RGB(127,0,0));
```

```
  TextOut(hdc,20,50,"a) MEMORIA RAM.",15);
  TextOut(hdc,290,50,"b) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,20,70,"c) MEMORIA ROM.",15);
  TextOut(hdc,290,70,"d) ALU.",7);
 pregunta[0]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) MEMORIA DE SOLO LECTURA:",27);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,330,120,"b) MEMORIA ROM.",15);
  TextOut(hdc,20,140,"c) MEMORIA RAM.",15);
  TextOut(hdc,330,140,"d) BUFFER",9);
 pregunta[1]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) CONTIENE CIRCUITOS SINCRONIZADORES:",13);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) UNIDAD ARITMETICO / LOGICA.",30);
  TextOut(hdc,330,190,"b) UNIDAD DE CONTROL.",21);
  TextOut(hdc,20,210,"c) UNIDADES DE ENTRADA / SALIDA.",32);
  TextOut(hdc,330,210,"d) UNIDAD DE MEMORIA.",21);
 pregunta[2]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) SE UTILIZAN PARA LA ADECUADA COMUNICACION ENTRE
DISPOSITIVOS DE",66);
TextOut(hdc,10,260,"ENTRADA / SALIDA Y LA COMPUTADORA:",34);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,280,"a) LOS REGISTROS ",17);
  TextOut(hdc,330,280,"b) LOS CIRCUITOS SINCRONIZADORES.",33);
  TextOut(hdc,20,300,"c) LOS ACUMULADORES.",20);
  TextOut(hdc,330,300,"d) LOS DISPOSITIVIS DE MEMORIA.",31);
 pregunta[3]='b';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_w=4; j=7;
ReleaseDC(hwnd,hdc);
}


//*****************************************************************************
****
//  FUNCION EVALUACION MICROCOMPUTADORA 3

 void EVALUACION_MICRO_3(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) RUTA ELECTRICA QUE CONDUCE INFORMACION O CORRIENTE:",54);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) REGISTRO ",12);
  TextOut(hdc,290,50,"b) ACUMULADOR.",14);
  TextOut(hdc,20,70,"c) PERIFERICO.",14);
```

```
  TextOut(hdc,290,70,"d) BUS",6);
  pregunta[0]='d';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) CUANTOS BUSES DIFERENTES CONTIENE LA COMPUTADORA:",52);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) 5",4);
  TextOut(hdc,330,120,"b) 3",4);
  TextOut(hdc,20,140,"c) 4",4);
  TextOut(hdc,330,140,"d) 2",4);
  pregunta[1]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) LO UTILIZA EL BUS DE CONTROL PARA ENVIAR DIRECCIONES AL
µP.",62);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) BUS DE DIRECCIONES.",22);
  TextOut(hdc,330,190,"b) BUS DE DATOS.",16);
  TextOut(hdc,20,210,"c) BUS DE ENERGIA.",18);
  TextOut(hdc,330,210,"d) -----",8);
  pregunta[2]='a';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4)  TRANSMITE  DATOS  HACIA  CIERTAS  DIRECCIONES  DE
MEMORIA:",56);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) BUS DE DATOS.",16);
  TextOut(hdc,330,260,"b) BUS DE DIRECCIONES.",22);
  TextOut(hdc,20,280,"c) BUS DE CONTROL.",18);
  TextOut(hdc,330,280,"d) BUS DE ENERGIA.",18);
  pregunta[3]='a';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_w=6; j=7;
ReleaseDC(hwnd,hdc);
}


//**********************************************************************************
****
//  FUNCION EVALUACION MICROCOMPUTADORA 4

 void EVALUACION_MICRO_4(HWND hwnd){

HDC hdc;
hdc=GetDC(hwnd);
SetBkMode(hdc,TRANSPARENT);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,30,"1) ENVIA SEÑALES AL ALU Y A LA MEMORIA PARA EJECUTAR UN
PROGRAMA.",65);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,50,"a) BUS DE DIRECCIONES.",22);
  TextOut(hdc,290,50,"b) BUS DE DATOS.",16);
  TextOut(hdc,20,70,"c) BUS DE ENERGIA.",18);
  TextOut(hdc,290,70,"d) BUS DE CONTROL.",18);
  pregunta[0]='d';
```

```
SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,100,"2) ESPECIFICAN UNA DIRECCION AL MICROPROCESADOR (µP):",53);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,120,"a) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,330,120,"b) SEÑALES DE LINEA DE CONTROL.",31);
  TextOut(hdc,20,140,"c) SEÑALES DE DISTRIBUCION.",27);
  TextOut(hdc,330,140,"d) BUS DE DATOS",15);
 pregunta[1]='b';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,170,"3) GENERA LOS INTERVALOS DE TIEMPO ENTRE OPERACIONES DEL
SISTEMA.",65);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,190,"a) SEÑALES DE LINEA DE CONTROL.",31);
  TextOut(hdc,330,190,"b) UNIDAD DE MEMORIA.",21);
  TextOut(hdc,20,210,"c) SEÑALES DE DISTRIBUCION.",27);
  TextOut(hdc,330,210,"d) UNIDAD DE ENTRADA.",21);
 pregunta[2]='c';

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,10,240,"4) SE UTILIZAN PARA LEER O ESCRIBIR EN LOS DISPOSITIVOS DE
ENTR./SALIDA.",72);
  SetTextColor(hdc,RGB(127,0,0));
  TextOut(hdc,20,260,"a) PUERTAS DE ENTR./SALIDA.",27);
  TextOut(hdc,330,260,"b) LOS CIRCUITOS SINCRONIZADORES.",33);
  TextOut(hdc,20,280,"c) LOS ACUMULADORES.",20);
  TextOut(hdc,330,280,"d) BUS DE CONTROL.",18);
 pregunta[3]='a';

cantidad_preguntas=4;
ULTIMA_IMAGEN(hwnd);
var_w=7; j=7;
ReleaseDC(hwnd,hdc);
}




//-----------------------------------------------------------------------------
---
//-----------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "VARIOS.H"

#define cuenta6 200
#define cuenta7 150
#define cuenta 40
#define cuenta4 100
extern HANDLE hInst;
extern HCURSOR Reloj_Arena;
extern HCURSOR Cursor_Cruz;
extern RECT limite;
extern int X=13, Y=17;
extern int p;
extern int var_z;
```

```
HDC hdcMemory1; // contextos de visualizacion de pag. oculta
HBITMAP hbmpMyBitmap1,hbmpOld1;
extern HDC hdcMemory; // contextos de visualizacion de pag. oculta
extern HBITMAP hbmpMyBitmap,hbmpOld;


//**********************************************************************************
****
//   FUNCION FLECHAS_DATOS    (dibuja las flechas de flujo de datos)

 void FLECHAS_DATOS(HDC hdc,int k,int y,int g){


  HBRUSH Brocha, Brocha1;
  HPEN Pluma, Pluma1;
  POINT puntos1[7];
  POINT puntos2[7];
  POINT puntos3[7];
  POINT puntos4[7];
  POINT puntos5[9];
  POINT puntos6[9];
  POINT puntos7[11];
  POINT puntos8[9];
  POINT puntos9[11];
  POINT puntos10[9];

  puntos1[0].x = 580-X;  puntos1[0].y = 225+Y;
  puntos1[1].x = 605-X;  puntos1[1].y = 225+Y;
  puntos1[2].x = 605-X;  puntos1[2].y = 215+Y;
  puntos1[3].x = 635-X;  puntos1[3].y = 240+Y;
  puntos1[4].x = 605-X;  puntos1[4].y = 265+Y;
  puntos1[5].x = 605-X;  puntos1[5].y = 255+Y;
  puntos1[6].x = 580-X;  puntos1[6].y = 255+Y;

  puntos2[0].x = 25-X;  puntos2[0].y = 225+Y;
  puntos2[1].x = 50-X;  puntos2[1].y = 225+Y;
  puntos2[2].x = 50-X;  puntos2[2] y = 215+Y;
  puntos2[3].x = 80-X;  puntos2[3].y = 240+Y;
  puntos2[4].x = 50-X;  puntos2[4].y = 265+Y;
  puntos2[5].x = 50-X;  puntos2[5].y = 255+Y;
  puntos2[6].x = 25-X;  puntos2[6].y = 255+Y;

  puntos3[0].x = 305-X;  puntos3[0].y = 275+Y;
  puntos3[1].x = 305-X;  puntos3[1].y = 300+Y;
  puntos3[2].x = 315-X;  puntos3[2].y = 300+Y;
  puntos3[3].x = 290-X;  puntos3[3].y = 330+Y;
  puntos3[4].x = 265-X;  puntos3[4].y = 300+Y;
  puntos3[5].x = 275-X;  puntos3[5].y = 300+Y;
  puntos3[6].x = 275-X;  puntos3[6].y = 275+Y;

  puntos4[0].x = 380-X;  puntos4[0].y = 330+Y;
  puntos4[1].x = 380-X;  puntos4[1].y = 305+Y;
  puntos4[2].x = 390-X;  puntos4[2].y = 305+Y;
  puntos4[3].x = 365-X;  puntos4[3].y = 275+Y;
  puntos4[4].x = 340-X;  puntos4[4].y = 305+Y;
  puntos4[5].x = 350-X;  puntos4[5].y = 305+Y;
  puntos4[6].x = 350-X;  puntos4[6].y = 330+Y;
```

```
puntos5[0].x = 400-X;  puntos5[0].y = 380+Y;
puntos5[1].x = 520-X;  puntos5[1].y = 380+Y;
puntos5[2].x = 520-X;  puntos5[2].y = 330+Y;
puntos5[3].x = 510-X;  puntos5[3].y = 330+Y;
puntos5[4].x = 535-X;  puntos5[4].y = 300+Y;
puntos5[5].x = 560-X;  puntos5[5].y = 330+Y;
puntos5[6].x = 550-X;  puntos5[6].y = 330+Y;
puntos5[7].x = 550-X;  puntos5[7].y = 410+Y;
puntos5[8].x = 400-X;  puntos5[8].y = 410+Y;

puntos6[0].x = 145-X;  puntos6[0].y = 300+Y;
puntos6[1].x = 145-X;  puntos6[1].y = 375+Y;
puntos6[2].x = 220-X;  puntos6[2].y = 375+Y;
puntos6[3].x = 220-X;  puntos6[3].y = 365+Y;
puntos6[4].x = 250-X;  puntos6[4].y = 390+Y;
puntos6[5].x = 220-X;  puntos6[5].y = 415+Y;
puntos6[6].x = 220-X;  puntos6[6].y = 405+Y;
puntos6[7].x = 115-X;  puntos6[7].y = 405+Y;
puntos6[8].x = 115-X;  puntos6[8].y = 300+Y;

puntos7[0].x = 400-X;  puntos7[0].y = 110+Y;
puntos7[1].x = 470-X;  puntos7[1].y = 110+Y;
puntos7[2].x = 470-X;  puntos7[2].y = 370+Y;
puntos7[3].x = 430-X;  puntos7[3].y = 370+Y;
puntos7[4].x = 430-X;  puntos7[4].y = 380+Y;
puntos7[5].x = 400-X;  puntos7[5].y = 355+Y;
puntos7[6].x = 430-X;  puntos7[6].y = 330+Y;
puntos7[7].x = 430-X;  puntos7[7].y = 340+Y;
puntos7[8].x = 440-X;  puntos7[8].y = 340+Y;
puntos7[9].x = 440-X;  puntos7[9].y = 140+Y;
puntos7[10].x = 400-X;  puntos7[10].y = 140+Y;

puntos8[0].x = 400-X;  puntos8[0].y = 70+Y;
puntos8[1].x = 550-X;  puntos8[1].y = 70+Y;
puntos8[2].x = 550-X;  puntos8[2].y = 140+Y;
puntos8[3].x = 560-X;  puntos8[3].y = 140+Y;
puntos8[4].x = 535-X;  puntos8[4].y = 170+Y;
puntos8[5].x = 510-X;  puntos8[5].y = 140+Y;
puntos8[6].x = 520-X;  puntos8[6].y = 140+Y;
puntos8[7].x = 520-X;  puntos8[7].y = 100+Y;
puntos8[8].x = 400-X;  puntos8[8].y = 100+Y;

puntos9[0].x = 250-X;  puntos9[0].y = 365+Y;
puntos9[1].x = 178-X;  puntos9[1].y = 365+Y;
puntos9[2].x = 178-X;  puntos9[2].y = 115+Y;
puntos9[3].x = 220-X;  puntos9[3].y = 115+Y;
puntos9[4].x = 220-X;  puntos9[4].y = 105+Y;
puntos9[5].x = 250-X;  puntos9[5].y = 130+Y;
puntos9[6].x = 220-X;  puntos9[6].y = 155+Y;
puntos9[7].x = 220-X;  puntos9[7].y = 145+Y;
puntos9[8].x = 208-X;  puntos9[8].y = 145+Y;
puntos9[9].x = 208-X;  puntos9[9].y = 335+Y;
puntos9[10].x = 250-X; puntos9[10].y = 335+Y;

puntos10[0].x = 115-X;  puntos10[0].y = 165+Y;
puntos10[1].x = 115-X;  puntos10[1].y = 65+Y;
puntos10[2].x = 220-X;  puntos10[2].y = 65+Y;
puntos10[3].x = 220-X;  puntos10[3].y = 55+Y;
```

```
 puntos10[4].x = 250-X;  puntos10[4].y = 80+Y;
 puntos10[5].x = 220-X;  puntos10[5].y = 105+Y;
 puntos10[6].x = 220-X;  puntos10[6].y = 95+Y;
 puntos10[7].x = 145-X;  puntos10[7].y = 95+Y;
 puntos10[8].x = 145-X;  puntos10[8].y = 165+Y;

switch(k){

 case 1:
  Brocha= CreateSolidBrush(RGB(0,0,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(255,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

 case 2:
  Brocha= CreateSolidBrush(RGB(0,0,127));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

 case 3:
  Brocha= CreateSolidBrush(RGB(127,127,127));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

  }

switch(y){

 case 1:
  Polygon(hdc,puntos1,7);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 2:
  Polygon(hdc,puntos2,7);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 3:
  Polygon(hdc,puntos3,7);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 4:
```

```
  Polygon(hdc,puntos4,7);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 5:
  Polygon(hdc,puntos5,9);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 6:
  Polygon(hdc,puntos6,9);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 7:
  Polygon(hdc,puntos7,11);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 8:
  Polygon(hdc,puntos8,9);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 9:
  Polygon(hdc,puntos9,11);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 10:
  Polygon(hdc,puntos10,9);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;
 case 11:
  Polygon(hdc,puntos1,7);
  Polygon(hdc,puntos2,7);
```

```
 Polygon(hdc,puntos3,7);
 Polygon(hdc,puntos4,7);
 Polygon(hdc,puntos5,9);
 Polygon(hdc,puntos6,9);
 Polygon(hdc,puntos7,11);
 Polygon(hdc,puntos8,9);
 Polygon(hdc,puntos9,11);
 Polygon(hdc,puntos10,9);
 switch(g){
  case 1: MessageBeep(0); break;
  case 2: MessageBeep(-1); break;
  case 3: break;
        }
 break;

case 12:
 Polygon(hdc,puntos9,11);
 Polygon(hdc,puntos10,9);
 switch(g){
  case 1: MessageBeep(0); break;
  case 2: MessageBeep(-1); break;
  case 3: break;
        }
 break;

case 13:
 Polygon(hdc,puntos7,11);
 Polygon(hdc,puntos8,9);
 switch(g){
  case 1: MessageBeep(0); break;
  case 2: MessageBeep(-1); break;
  case 3: break;
        }
 break;

case 14:
 Polygon(hdc,puntos5,9),
 Polygon(hdc,puntos9,11);
 switch(g){
  case 1: MessageBeep(0); break;
  case 2: MessageBeep(-1); break;
  case 3: break;
        }
 break;

case 15:
 Polygon(hdc,puntos6,9);
 Polygon(hdc,puntos7,11);
 switch(g){
  case 1: MessageBeep(0); break;
  case 2: MessageBeep(-1); break;
  case 3: break;
        }
 break;

case 16:
 Polygon(hdc,puntos6,9);
 Polygon(hdc,puntos10,9);
```

```
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;

 case 17:
  Polygon(hdc,puntos5,9);
  Polygon(hdc,puntos8,9);
  switch(g){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
          }
  break;

  }
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

}

//*****************************************************************************
****
// FUNCION FLECHAS_CONTROL(DIBUJA LA FLECHAS DE FLUJO DE CONTROL)

void FLECHAS_CONTROL(HDC hdc,int f,int z,int j){

 HBRUSH Brocha, Brocha1;
 HPEN Pluma, Pluma1;

 POINT flecha1[7];
 POINT flecha2[7];
 POINT flecha3[7];
 POINT flecha4[7];

 flecha1[0].x = 310;  flecha1[0].y = 293;
 flecha1[1].x = 320;  flecha1[1].y = 293;
 flecha1[2].x = 320;  flecha1[2].y = 333;
 flecha1[3].x = 325;  flecha1[3].y = 333;
 flecha1[4].x = 315;  flecha1[4].y = 348;
 flecha1[5].x = 305;  flecha1[5].y = 333;
 flecha1[6].x = 310;  flecha1[6].y = 333;

 flecha2[0].x = 315;  flecha2[0].y = 163;
 flecha2[1].x = 325;  flecha2[1].y = 178;
 flecha2[2].x = 320;  flecha2[2].y = 178;
 flecha2[3].x = 320;  flecha2[3].y = 211;
 flecha2[4].x = 310;  flecha2[4].y = 211;
 flecha2[5].x = 310;  flecha2[5].y = 178;
 flecha2[6].x = 305;  flecha2[6].y = 178;

 flecha3[0].x = 243;  flecha3[0].y = 247;
 flecha3[1].x = 173;  flecha3[1].y = 247;
 flecha3[2].x = 173;  flecha3[2].y = 242;
```

```
flecha3[3].x = 158;  flecha3[3].y = 252;
flecha3[4].x = 173;  flecha3[4].y = 262;
flecha3[5].x = 173;  flecha3[5].y = 257;
flecha3[6].x = 243;  flecha3[6].y = 257;

flecha4[0].x = 384;  flecha4[0].y = 247;
flecha4[1].x = 464;  flecha4[1].y = 247;
flecha4[2].x = 464;  flecha4[2].y = 242;
flecha4[3].x = 479;  flecha4[3].y = 252;
flecha4[4].x = 464;  flecha4[4].y = 262;
flecha4[5].x = 464;  flecha4[5].y = 257;
flecha4[6].x = 384;  flecha4[6].y = 257;

switch(f){

 case 1:
  Brocha= CreateSolidBrush(RGB(127,0,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;
 case 2:
  Brocha= CreateSolidBrush(RGB(0,255,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;
 case 3:
  Brocha= CreateSolidBrush(RGB(255,255,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  break;

 }

switch(z){

 case 1:
  Polygon(hdc,flecha1,7);
  switch(j){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
         }
  break;
 case 2:
  Polygon(hdc,flecha2,7);
  switch(j){
   case 1: MessageBeep(0); break;
   case 2: MessageBeep(-1); break;
   case 3: break;
      }
  break;
 case 3:
  Polygon(hdc,flecha3,7);
  switch(j){
   case 1: MessageBeep(0); break;
```

```
    case 2: MessageBeep(-1); break;
    case 3: break;
          }
   break;
  case 4:
   Polygon(hdc,flecha4,7);
   switch(j){
    case 1: MessageBeep(0); break;
    case 2: MessageBeep(-1); break;
    case 3: break;
          }
   break;
  case 5:
   Polygon(hdc,flecha1,7);
   Polygon(hdc,flecha2,7);
   Polygon(hdc,flecha3,7);
   Polygon(hdc,flecha4,7);
   switch(j){
    case 1: MessageBeep(0); break;
    case 2: MessageBeep(-1); break;
    case 3: break;
          }
   break;

 }
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

}


//*****************************************************************************
****
// FUNCION MODULO I

 void MODULO_I(HWND hwnd,HDC hdc){

HBRUSH Brocha, Brocha1;
HPEN Pluma, Pluma1;

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

Brocha= CreateSolidBrush(RGB(0,0,255));
Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,40,4,430,57);
Rectangle(hdc,440,2,620,82);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
```

```
SetTextColor(hdc,RGB(191,191,191));
SetBkColor(hdc,RGB(0,0,0));
TextOut(hdc,70,15,"TODA COMPUTADORA ESTA COMPUESTA POR CINCO",41);
TextOut(hdc,70,35,"ELEMENTO BASICOS, QUE SON : ",28);
MessageBeep(0);

TIEMPO(cuenta6);
SetTextColor(hdc,RGB(255,255,0));
SetBkColor(hdc,RGB(0,0,0));

TextOut(hdc,445,5,"LA UNIDAD ARIT.LOGICA",21);
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,1,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,1,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

TextOut(hdc,445,18,"LA UNIDAD DE CONTROL",20);
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,2,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,2,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

TextOut(hdc,445,33,"LA UNIDAD DE MEMORIA",20);
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,3,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,3,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

TextOut(hdc,445,49,"LA UNIDAD DE ENTRADA",20);
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,4,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,2,4,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

TextOut(hdc,445,65,"LA UNIDAD DE SALIDA",19);
TIEMPO(cuenta4);
for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,5,1);
```

```
TIEMPO(cuenta);
INT_MODULOS(hdc,2,5,2);
TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);

var_z=2;
} //TERMINA MODULO 1


//*****************************************************************************
****
// FUNCION MODULO 2
void MODULO_2(HWND hwnd, HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,255,0,0,191,191,191,"LOS MODULOS DE UNIDAD ARITMETICA
LOGICA Y EL DE UNIDAD DE CONTROL",
              "JUNTOS CONFORMAN LA UNIDAD CENTRAL DE PROCESAMIENTO ( CPU
)",1,50,10,50,30,65,59);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,2,7,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,3,7,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,255,255,255,0,0,0,"LAS LINEAS GRUESAS REPRESENTAN EL
FLUJO",
        "DE LOS DATOS O INFORMACION",1,50,10,50,30,39,26);

TIEMPO(cuenta);

for(p=0,p<=4;p++){
FLECHAS_DATOS(hdc,1,11,1);
TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,11,2);
TIEMPO(cuenta);
}
TIEMPO(cuenta);
```

```
VER_CADENA_CON(hwnd,127,0,0,191,191,191,"LAS LINEAS DELGADAS REPRESENTAN EL
FLUJO",
        "DE LAS SEÑALES DE CONTROL",1,50,10,50,30,40,25);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_CONTROL(hdc,2,5,1);
TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,1,5,2);
TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);

var_r=3;
} // TERMINA MODULO 2



//*****************************************************************************
****
//   FUNCION MODULO 3

void MODULO_3(HWND hwnd, HDC hdc){

SetCursor(Reloj_Arena);

MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA UNIDAD ARITMETICA LOGICA (UAL O ALU)
ES EL AREA EN LA QUE SE",
        "REALIZAN LAS OPERACIONES ARITMETICAS Y LOGICAS CON
DATOS",1,30,10,30,30,63,56);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,1,1);
TIEMPO(cuenta);
INT_MODULOS(hdc,3,1,2);
TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,127,0,0,0,0,"EL TIPO DE OPERACION QUE SE REALIZARA SE
DETERMINA",
        "POR MEDIO DE SEÑALES DE LA UNIDAD DE CONTROL",1,50,10,50,30,50,44);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
```

```
FLECHAS_CONTROL(hdc,1,2,1);
 TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,2,2,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LOS DATOS QUE SERAN UTILIZADOS POR LA
UAL, PUEDEN PROVENIR",
                    "DE LA UNIDAD DE MEMORIA O DE LA UNIDAD DE
ENTRADA",1,50,10,50,30,58,49);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,12,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,12,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=5;
} //  TERMINA MODULO 3


//*****************************************************************************
****
//    FUNCION MODULO 4

void MODULO_4(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);

VER_CADENA_CON(hwnd,0,0,0,0,127,0,"LOS RESULTADOS DE OPERACIONES REALIZADAS
EN LA UAL PUEDEN TRANSFERIRSE",
          "A LA UNIDAD DE MEMORIA PARA SER ALMACENADAS O A LA UNIDAD DE
SALIDA",1,18,10,18,30,70,67);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,13,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,13,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,0,0,255,255,0,"LA UNIDAD DE MEMORIA ALMACENAN GRUPOS
DE DIGITOS BINARIOS (PALABRA)",
        "QUE PUEDEN REPRESENTAR INSTRUCCIONES (PROGRAMA) QUE LA MAQUINA
EJECUTA",1,14,10,14,30,67,70);
```

```
TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,3,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,3,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,127,0,0,191,191,191,"ASI COMO LOS DATOS QUE SERAN
OPERADOS POR EL PROGRAMA",
        "",1,40,10,40,30,53,0);

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA MEMORIA SIRVE TAMBIEN COMO
ALMACENAMIENTO DE RESULTADOS INTERMEDIOS",
                "Y FINALES DE OPERACIONES ARITMETICAS REALIZADAS EN LA
UAL",1,14,10,14,30,70,57);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,7,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,7,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=6;
} //  TERMINA MODULO 4


//**********************************************************************************
****
// FUNCION MODULO 5

 void MODULO_5(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,0,127,0,0,0,0,"LA OPERACION DE LA MEMORIA ES CONTROLADA
POR MEDIO DE SEÑALES DE LA",
                "UNIDAD DE CONTROL QUE INDICA UNA OPERACION DE LECTURA O
ESCRITURA",1,50,10,50,30,67,65);

TIEMPO(cuenta);
```

```
for(p=0;p<=4;p++){
FLECHAS_CONTROL(hdc,1,1,1);
 TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,2,1,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,127,0,0,0,0,"UNA LOCALIDAD DADA EN LA MEMORIA SE
ACCESA POR LA UNIDAD",
                    "DE CONTROL, OFRECIENDO EL CODIGO DE DIRECCION
ADECUADO",1,50,10,50,30,56,54);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,3,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,3,2);
 TIEMPO(cuenta);
}

BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=9;
} // TERMINA MODULO 5



//*****************************************************************************
****
// FUNCION MODULO 6


 void MODULO_6(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"PUEDE ESCRIBIRSE INFORMACION EN LA
MEMORIA PROVENIENTE DE LA",
                    "UNIDAD ARITMETICO / LOGICA O DE LA UNIDAD DE
ENTRADA",1,23,10,23,30,60,52);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,15,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,15,2);
 TIEMPO(cuenta);
}
```

```
TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,0,0,255,255,0,"TAMBIEN PUEDE LEERSE INFORMACION DE LA
MEMORIA EN LA UAL",
        "O EN LA UNIDAD DE SALIDA",1,23,10,23,30,56,24);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,14,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,14,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=10;
}// TERMINA MODULO 6


//*****************************************************************************
****
// FUNCION MODULO 7


void MODULO_7(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA UNIDAD DE ENTRADA ESTA FORMADA POR
TODOS LOS DISPOSITIVOS QUE SE",
        "USAN PARA INTODUCIR INFORMACION Y DATOS EXTERNOS EN LA
MAQUINA",1,20,10,20,30,67,62);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,4,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,4,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,0,0,255,255,0,"ESTOS DATOS SE PUEDEN COLOCAR EN LA
UNIDAD DE",
        "MEMORIA O EN LA UNIDAD ARITMETICO / LOGICA",1,23,10,23,30,45,42);


TIEMPO(cuenta);
```

```
for(p=0;p<=4;p++){
FLECHAS_DATOS(hdc,3,16,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,16,2);
 TIEMPO(cuenta);
}

VER_CADENA_CON(hwnd,0,127,0,255,255,255,"LA UNIDAD DE CONTROL ES QUIEN
DETERMINA HACIA DONDE SE ENVIA",
                "LA INFORMACION DE ENTRADA POR MEDIO DE SEÑALES DE
CONTROL",1,50,10,50,30,60,57);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_CONTROL(hdc,1,3,1);
 TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,2,3,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=12;
}// TERMINA MODULO 7



//*************************************************************************
****
// FUNCION MODULO 8


void MODULO_8(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA UNIDAD DE SALIDA ESTA FORMADA POR
LOS DISPOSITIVOS QUE SE USAN",
          "PARA TRANSFERIR DATOS E INFORMACION DE LA COMPUTADORA AL
EXTERIOR",1,20,10,20,30,65,65);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,5,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,5,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);
```

```
VER_CADENA_CON(hwnd,0,0,0,255,255,0,"LOS DISPOSITIVOS DE SALIDA SON DIRIGIDOS
POR",
        "LA UNIDAD DE CONTROL",1,50,10,50,30,44,20);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
FLECHAS_CONTROL(hdc,1,4,1);
 TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,2,4,2);
 TIEMPO(cuenta);
}

VER_CADENA_CON(hwnd,0,127,0,255,255,255,"LOS DISPOSITIVOS DE SALIDA PUEDEN
RECIBIR DATOS DE LA UNIDAD",
        "DE MEMORIA O DE LA UNIDAD ARITMETICO / LOGICA",1,50,10,50,30,60,45);

TIEMPO(cuenta);

for(p=0,p<=4;p++){
FLECHAS_DATOS(hdc,3,17,1);
 TIEMPO(cuenta);
FLECHAS_DATOS(hdc,2,17,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=13;
}//  TERMINA MODULO 8



//**********************************************************************************
****
// FUNCION MODULO 9


 void MODULO_9(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"EL ASPECTO MAS IMPORTANTE DE LAS
UNIDADES E/S ES LA SINCRONIZACION",
        "",1,20,10,20,30,66,0);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,8,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,8,2);
```

```
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,0,0,255,255,0,"ESTA LA PODEMOS DEFINIR COMO LA UNION DE
DISPOSITIVOS DISIMILARES",
                "PARA QUE PUEDAN FUNCIONAR DE MANERA COMPATIBLE Y
COORDINADA",1,50,10,50,30,65,59);

  TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,0,0,255,255,0,"ES DECIR, ES LA SINCRONIZACION DE LA
TRANSMISION DE INFORMACION",
          "DIGITAL ENTRE LA COMPUTADORA  Y DISPOSITIVOS EXTERNOS DE
ENT/SAL.",1,50,10,50,30,63,64);

BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_z=15;
}//  TERMINA MODULO 9



//*****************************************************************************
****
// FUNCION MODULO 10


void MODULO_10(HWND hwnd,HDC hdc){

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,1,5,3);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"EN ESTE MODULO SE ELIGE LA SIGUIENTE
INSTRUCCION, INTERPRETA SU",
      "DIRECCION (LOCALIZACION) Y PARTES OPERATIVAS",1,20,10,20,30,63,44);

TIEMPO(cuenta);

for(p=0;p<=4;p++){
INT_MODULOS(hdc,1,2,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,2,2);
 TIEMPO(cuenta);
}

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"ESTA  INSTRUCCION  ES  EJECUTADA
APLICANDO LAS SEÑALES ADECUADAS A LA UAL",
          " Y A OTRAS PARTES DEL SISTEMA DE ACUERDO A SU
INTERPRETACION.",1,20,10,20,30,70,64);

for(p=0;p<=4;p++){
```

```
FLECHAS_CONTROL(hdc,1,5,1);
 TIEMPO(cuenta);
FLECHAS_CONTROL(hdc,2,5,2);
 TIEMPO(cuenta);
}
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);

var_z=16;
}//  TERMINA MODULO 10




//-----------------------------------------------------------------------------
---
//-----------------------------------------------------------------------------
---

#include "WINDOWS.H"
#include "FUNCION.H"
#include "VARIOS.H"
#include "STRING.H"
#include "STDIO.H"

#define cuenta 40
#define cuenta1 5
#define cuenta2 70
#define cuenta3 10
#define cuenta4 100
#define cuenta6 200

 static void CADENA_BITS(HDC);
 static void CADENA_BITS_1(HWND,HDC);

extern HANDLE hInst;
extern HCURSOR Reloj_Arena;
extern HCURSOR Cursor_Cruz;

extern HDC hdcU_Ent_Sal; // contextos de visualizacion de pag. oculta
extern HBITMAP hbmpU_Ent_Sal,hbmpU_Ent_Sal1;

extern int p,r,q;
extern int var_z,var_w,j;
extern RECT limite;
extern HFONT Fuente,PreFuente,Font,PrevFont;



//*****************************************************************************
****
// FUNCION CADENA BITS
```

```
static void CADENA_BITS(HDC hdc){

int t,v,l,w,f,e,g,h,i;
char cadena[16]="1001 0101 1100  ";
char *s;

  HBRUSH Brocha, Brocha1;
  HPEN Pluma, Pluma1;

  Brocha= CreateSolidBrush(RGB(127,127,127));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);

s=cadena;
f=70,e=0;
w=11;g=0;
i=0,h=0;
v=0;

Rectangle(hdc,270,330,300,360);  //para crear el cuadrado pequeño (gris oscuro)
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  SelectObject(hdc,Pluma1);
  DeleteObject(Pluma1);


while(e<=15){
  SetTextColor(hdc,RGB(255,255,255));
  SetBkMode(hdc,TRANSPARENT);
  f=f+(12*e);
  h=w-e;
  i=v;
 while(g<=h){
  s=&cadena[g];
  TextOut(hdc,f,272,s,1);
  f=f+12;
  g++;
    }
 while(i<15){
   s=&cadena[14-i];
   TextOut(hdc,283,340,s,1);
   i=15;}
   i=v;
 while(i<16){
   s=&cadena[15-i];
   TextOut(hdc,335,340,s,1);
   MessageBeep(0);
   TIEMPO(cuenta);
   i=16;}

  Brocha= CreateSolidBrush(RGB(0,127,0));
  Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma1= SelectObject(hdc,Pluma);
  Rectangle(hdc,60,265,240,293);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
```

```
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

Brocha= CreateSolidBrush(RGB(0,0,127));
Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
Brocha1= SelectObject(hdc,Brocha);
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,320,245,470,360);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

Brocha= CreateSolidBrush(RGB(127,127,127));
Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
Brocha1= SelectObject(hdc,Brocha);
Pluma1= SelectObject(hdc,Pluma);
Rectangle(hdc,270,330,300,360);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

  v++;
  e++;
  g=0;
  f=70;
  }

 SetTextColor(hdc,RGB(191,191,191));
 v=400; g=0; q=0;w=0;i=310;
 while(w!=3){
 MessageBeep(0);
 TIEMPO(cuenta);
 while(q!=5){
       s=&cadena[g];
       TextOut(hdc,v,i,s,1);
       v=v+12;
       g++;
       q++;
       }
  w++;
  v=400;
  i=i+14;
  q=0;
  }

}



//*****************************************************************************
****
// FUNCION CADENA BITS 1

static void CADENA_BITS_1(HWND hwnd,HDC hdc){

int t,v,z,w,f,e,g,h,i,n;
```

```
char cadena[10]="100101    ";
char cadena1[22]="110100              ";
char *s,*d;

HBRUSH Brocha, Brocha1;
HPEN Pluma1, Pluma;

POINT puntos7[11];
POINT flecha4[7];
POINT puntos5[9];

puntos5[0].x = 180;  puntos5[0].y = 370;
puntos5[1].x = 180;  puntos5[1].y = 400;
puntos5[2].x = 75;  puntos5[2].y = 400;
puntos5[3].x = 75;  puntos5[3].y = 340;
puntos5[4].x = 65;  puntos5[4].y = 340;
puntos5[5].x = 90;  puntos5[5].y = 310;
puntos5[6].x = 115;  puntos5[6].y = 340;
puntos5[7].x = 105;  puntos5[7].y = 340;
puntos5[8].x = 105;  puntos5[8].y = 370;

flecha4[0].x = 170;  flecha4[0].y = 200;
flecha4[1].x = 170;  flecha4[1].y = 170;
flecha4[2].x = 300;  flecha4[2].y = 170;
flecha4[3].x = 300;  flecha4[3].y = 160;
flecha4[4].x = 330,  flecha4[4].y = 185;
flecha4[5].x = 300,  flecha4[5].y = 210,
flecha4[6].x = 300;  flecha4[6].y = 200;

puntos7[0].x = 470;  puntos7[0].y = 170;
puntos7[1].x = 520;  puntos7[1].y = 170;
puntos7[2].x = 520;  puntos7[2].y = 400;
puntos7[3].x = 400;  puntos7[3].y = 400;
puntos7[4].x = 400;  puntos7[4].y = 410;
puntos7[5].x = 370;  puntos7[5].y = 385;
puntos7[6].x = 400;  puntos7[6].y = 360;
puntos7[7].x = 400;  puntos7[7].y = 370;
puntos7[8].x = 490;  puntos7[8].y = 370;
puntos7[9].x = 490;  puntos7[9].y = 200;
puntos7[10].x = 470;  puntos7[10].y = 200;

SetCursor(Cursor_Cruz);
Pluma= CreatePen(PS_SOLID,1,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= CreateSolidBrush(RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);

  Polygon(hdc,puntos7,11);
  Polygon(hdc,flecha4,7);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= CreateSolidBrush(RGB(0,127,0));
```

```
Brocha1= SelectObject(hdc,Brocha);


VER_CADENA_SIN(hwnd,23,10,"LA U. DE CONTROL BUSCA Y TRAE UNA INSTRUCCION DE
LA MEMORIA ENVIANDO",68,0,0,255);
VER_CADENA_SIN(hwnd,23,30,"UNA DIRECCION Y UN COMANDO DE LECTURA A LA
UNIDAD DE MEMORIA",60,0,0,255);

SetTextColor(hdc,RGB(127,0,0));
SetBkMode(hdc,TRANSPARENT);
TextOut(hdc,215,150,"DIRECCION",9);
SetTextColor(hdc,RGB(0,0,0));

s=cadena;
f=55;e=0;
w=6;g=0;
i=0;h=0;
v=0;
n=290;

  TIEMPO(cuenta);

while(el=7){
  f=f+(10*e);
  h=w-e;
 while(g!=h){
  s=&cadena[g];
  TextOut(hdc,f,180,s,1);
  f=f+12;
  g++;
     }
   s=&cadena[6-i];
   TextOut(hdc,n,180,s,1);
   MessageBeep(0);
    TIEMPO(cuenta);
   Rectangle(hdc,42,155,135,195);

    i++;
    e++;
    g=0;
    n-=12;
    f=50;
   }
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);


 Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(127,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 SetTextColor(hdc,RGB(191,191,191));
 SetBkMode(hdc,TRANSPARENT);

g=6;  f=416; e=278;
while(g!=0){
```

```
  s=&cadena[g-1];
  MessageBeep(0);
  Rectangle(hdc,e,173,290,197);
  TextOut(hdc,f,180,s,1);
   TIEMPO(cuenta);
   f=f-12;
   e-=12;
   g--;
     }
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);


 Pluma= CreatePen(PS_SOLID,1,RGB(0,127,127));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,170,140,297,165);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 SetTextColor(hdc,RGB(127,0,0));
 TextOut(hdc,170,145,"COMANDO DE LECTURA",18);

 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);

  TIEMPO(cuenta);
 MessageBeep(0);
 TextOut(hdc,55,180,"0 1 1 0 0 1",11);
  TIEMPO(cuenta);
 Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(0,127,0));
 Brocha1= SelectObject(hdc,Brocha);
 MessageBeep(0);
 Rectangle(hdc,42,155,135,195);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 TextOut(hdc,210,180,"0 1 1 0 0 1",11);
 SetTextColor(hdc,RGB(191,191,191));
 SetBkMode(hdc,TRANSPARENT);
  TIEMPO(cuenta);
 Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(127,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,180,173,310,197);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
```

```
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

MessageBeep(0);
TextOut(hdc,355,200,"0 1 1 0 0 1",11);

 TIEMPO(cuenta);


VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA PALABRA DE INSTRUCCION ALMACENADA
EN LA LOCALIDAD DE LA MEMORIA",
            "SE TRANSFIERE DESPUES A LA UNIDAD DE CONTROL",1,20,10,20,30,66,44);

d=cadena1;
f=360,e=0;
w=5;g=0;
i=0;h=0;
z=380;v=0;
n=240;


 Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(255,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Polygon(hdc,puntos5,9);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 Brocha= CreateSolidBrush(RGB(255,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,230,360,330,410);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 SetTextColor(hdc,RGB(0,255,0));
 TextOut(hdc,190,330,"PALABRA DE INSTRUCCION",22);

 TIEMPO(cuenta);

while(e<=18){
 i=v;
 h=w-e;
 f=f+(10*e);
 Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(0,0,127));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,340,150,440,300);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 SetTextColor(hdc,RGB(255,255,0));
```

```
while(g<=h){
d=&cadena1[g];
TextOut(hdc,f,180,d,1);
f=f+12;
g++;
    }
while(i<7){
 d=&cadena1[6-i];
 TextOut(hdc,505,180,d,1);
 i=7; }
 i=v;
while(i<8){
 d=&cadena1[7-i];
 TextOut(hdc,505,247,d,1);
 i=8; }
 i=v;
while(i<9){
 d=&cadena1[8-i];
 TextOut(hdc,505,318,d,1);
 i=9; }
 i=v;
while(i<10){
 d=&cadena1[9-i];
 TextOut(hdc,505,380,d,1);
 i=10; }
 i=v;
while(i<11){
 d=&cadena1[10-i];
 TextOut(hdc,475,380,d,1);
 i=11; }
 i=v;
while(i<12){
 d=&cadena1[11-i];
 TextOut(hdc,435,380,d,1);
 i=12; }
 i=v;
while(i<13){
 d=&cadena1[12-i];
 TextOut(hdc,395,380,d,1);
 i=13; }
 i=v;
 SetTextColor(hdc,RGB(0,0,0));
while(i<14){
 d=&cadena1[13-i];
 TextOut(hdc,z,380,d,1);
 i=14; }
 z-=10;
 i=v;
while(i<15){
 d=&cadena1[14-i];
 TextOut(hdc,170,380,d,1);
 i=15; }
 i=v;
while(i<16){
 d=&cadena1[15-i];
 TextOut(hdc,130,380,d,1);
 i=16; }
 i=v;
```

```
  while(i<17){
   d=&cadena1[16-i];
   TextOut(hdc,90,380,d,1);
   i=17; }
   i=v;
  while(i<18){
   d=&cadena1[17-i];
   TextOut(hdc,90,330,d,1);
   i=18; }
   i=v;
  while(i<19){
   d=&cadena1[18-i];
   MessageBeep(0);
   TextOut(hdc,n,245,d,1);
   i=19; }
   n-=10;

   TIEMPO(cuenta);

  Pluma= CreatePen(PS_SOLID,1,RGB(127,127,127));
  Pluma1= SelectObject(hdc,Pluma);
  Brocha= CreateSolidBrush(RGB(127,127,127));
  Brocha1= SelectObject(hdc,Brocha);
  Rectangle(hdc,493,173,517,398);
  Rectangle(hdc,390,373,490,398);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  SelectObject(hdc,Pluma1);
  DeleteObject(Pluma1);

  Pluma= CreatePen(PS_SOLID,1,RGB(255,0,255));
  Pluma1= SelectObject(hdc,Pluma);
  Brocha= CreateSolidBrush(RGB(255,0,255));
  Brocha1= SelectObject(hdc,Brocha);
  Rectangle(hdc,80,373,178,398);
  Rectangle(hdc,80,325,100,398);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  SelectObject(hdc,Pluma1);
  DeleteObject(Pluma1);

    c++;
    g=0;
    v++;
    f=360;
   }

VER_CADENA_CON(hwnd,0,255,0,0,0,0,"ESTA PALABRA DE INSTRUCCION (CODIGO
BINARIO), ES DECODIFICADA POR CIRC.",
                "LOGICOS DE LA U. DE CONTROL PARA DETERMINAR LA INSTRUCION
SOLICITADA",1,12,10,12,30,71,68);

TIEMPO(cuenta);

}
```

```
//**********************************************************************************
****
//  FUNCION MODULO MEMORIA 1

void MOD_MEMORIA_1(HWND hwnd,HDC hdc){

 HBRUSH Brocha, Brocha1;
 POINT flecha4[7];  //para crear la flecha

 flecha4[0].x = 190;  flecha4[0].y = 335;
 flecha4[1].x = 230;  flecha4[1].y = 335;
 flecha4[2].x = 230;  flecha4[2].y = 330;
 flecha4[3].x = 250;  flecha4[3].y = 345;
 flecha4[4].x = 230;  flecha4[4].y = 360;
 flecha4[5].x = 230;  flecha4[5].y = 355;
 flecha4[6].x = 190;  flecha4[6].y = 355;

 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);
 SetCursor(Reloj_Arena);
 Rectangle(hdc,0,0,645,445);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);


 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,60,265,240,293,0,127,0,2);
 Brocha= CreateSolidBrush(RGB(255,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Polygon(hdc,flecha4,7);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,320,245,470,360,0,0,127,2);
 SetTextColor(hdc,RGB(0,0,0));
 SetBkColor(hdc,RGB(255,255,0));


 TextOut(hdc,350,180,"UNIDAD DE MEMORIA",17);
 TextOut(hdc,70,338,"DIGITO BINARIO",14);
 TextOut(hdc,80,200,"DATOS O INSTRUCCIONES",21);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

VER_CADENA_CON(hwnd,191,191,191,127,0,0,"LA MEMORIA ALMACENA GRUPOS DE
DIGITOS (PALABRA) BINARIOS QUE PUEDEN RE",
        "PRESENTAR INSTRUCCIONES (PROGRAMA) O DATOS QUE LA MAQUINA
EJECUTARA",1,23,10,23,30,70,67);

TIEMPO(cuenta);

CADENA_BITS(hdc);

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,127,0,0,255,255,0,"LA MEMORIA SIRVE TAMBIEN COMO
ALMACENAMIENTO DE RESULTADOS INTERMEDIOS",
        "Y FINALES DE OPERACIONES ARITMETICAS",1,23,10,23,30,70,36);
```

```
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);

SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_w=2;  j=4;
} //   TERMINA MODULO MEMORIA 1


//**********************************************************************************
****
/   FUNCION MODULO MEMORIA 2

 void MOD_MEMORIA_2(HWND hwnd,HDC hdc){

 int r;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;

POINT puntos[4];
POINT puntos1[4];
POINT flechas3[10];

  flechas3[0].x = 440;  flechas3[0].y = 235;
  flechas3[1].x = 465;  flechas3[1].y = 265;
  flechas3[2].x = 455;  flechas3[2].y = 265;
  flechas3[3].x = 455;  flechas3[3].y = 315;
  flechas3[4].x = 465;  flechas3[4].y = 315;
  flechas3[5].x = 440;  flechas3[5].y = 345;
  flechas3[6].x = 415;  flechas3[6].y = 315;
  flechas3[7].x = 425;  flechas3[7].y = 315;
  flechas3[8].x = 425;  flechas3[8].y = 265;
  flechas3[9].x = 415;  flechas3[9].y = 265;

puntos[0].x=40;  puntos[0].y=100;
puntos[1].x=20;  puntos[1].y=80;
puntos[2].x=520;  puntos[2].y=80;
puntos[3].x=540;  puntos[3].y=100;

puntos1[0].x=40;  puntos1[0].y=100;
puntos1[1].x=20;  puntos1[1].y=80;
puntos1[2].x=20;  puntos1[2].y=280;
puntos1[3].x=40;  puntos1[3].y=300;

  Brocha= CreateSolidBrush(RGB(127,127,127));
  Brocha1= SelectObject(hdc,Brocha);

  Rectangle(hdc,0,0,645,445);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);

  SetCursor(Reloj_Arena);
  Brocha= CreateSolidBrush(RGB(127,0,0));
  Brocha1= SelectObject(hdc,Brocha);

  SetBkMode(hdc,TRANSPARENT);
```

```
Rectangle(hdc,40,100,540,300);
Polygon(hdc,puntos,4);
Polygon(hdc,puntos1,4);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= CreateSolidBrush(RGB(255,255,0));
Brocha1 = SelectObject(hdc,Brocha);
Polygon(hdc,flechas3,10);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Pluma= CreatePen(PS_SOLID,3,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,140,165);
LineTo(hdc,140,140);
LineTo(hdc,440,140);
LineTo(hdc,440,165);
MoveTo(hdc,135,160);
LineTo(hdc,140,165);
LineTo(hdc,145,160);

MoveTo(hdc,140,235);
LineTo(hdc,140,260);
LineTo(hdc,400,260);
LineTo(hdc,400,235);
MoveTo(hdc,395,240);
LineTo(hdc,400,235);
LineTo(hdc,405,240);

MoveTo(hdc,195,190);
LineTo(hdc,235,190);
MoveTo(hdc,230,185);
LineTo(hdc,235,190);
LineTo(hdc,230,195);

MoveTo(hdc,195,210);
LineTo(hdc,235,210);
MoveTo(hdc,200,205);
LineTo(hdc,195,210);
LineTo(hdc,200,215);

MoveTo(hdc,345,190);
LineTo(hdc,385,190);
MoveTo(hdc,380,185);
LineTo(hdc,385,190);
LineTo(hdc,380,195);

MoveTo(hdc,345,210);
LineTo(hdc,385,210);
MoveTo(hdc,350,205);
LineTo(hdc,345,210);
LineTo(hdc,350,215);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
```

```
Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);
Brocha= CreateSolidBrush(RGB(0,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,41,101,152,120);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
SetTextColor(hdc,RGB(255,255,255));
TextOut(hdc,45,103,"COMPUTADORA",11);

Brocha= CreateSolidBrush(RGB(255,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,90,170,190,230);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetTextColor(hdc,RGB(0,0,0));
TextOut(hdc,114,192,"UNIDAD",6);
TextOut(hdc,103,204,"ARITMETICA",10);

Brocha= CreateSolidBrush(RGB(0,0,127));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,170,340,230);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetTextColor(hdc,RGB(191,191,191));
TextOut(hdc,255,192,"UNIDAD DE",9);
TextOut(hdc,260,204,"CONTROL",7);

Brocha= CreateSolidBrush(RGB(0,127,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,390,170,490,230);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetTextColor(hdc,RGB(0,0,255));
TextOut(hdc,398,175,"MEMORIA",7);
TextOut(hdc,398,187,"INTERNA",7);
TextOut(hdc,398,199,"( NUCLEO O",10);
TextOut(hdc,400,211,"MOS )",5);

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,372,350,508,435);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetTextColor(hdc,RGB(0,0,255));
TextOut(hdc,378,360,"ALMACENAMIENTO",14);
TextOut(hdc,378,375,"EN MASA EXTERNO",15);
TextOut(hdc,378,390,"(CINTA, DISCO,",14);
TextOut(hdc,378,405,"MBM)",4);

TIEMPO(cuenta);
```

```
VER_CADENA_CON(hwnd,0,255,0,0,0,0,"LA PRINCIPAL DIFERENCIA ENTRE LOS SISTEMAS
DIGITALES Y LOS ANALOGICOS",
                    "ES LA CAPACIDAD PARA ALMACENAR GRANDES CANTIDADES DE
INFRORMACION",1,12,10,12,30,69,65);
TIEMPO(cuenta);

VER_CADENA_CON(hwnd,0,255,0,0,0,0,"DIGITAL POR PERIODOS DE TIEMPO CORTOS O
LARGOS",
                    "",1,12,10,12,30,46,0);

TIEMPO(cuenta);

  VER_CADENA_CON(hwnd,0,127,0,0,0,0,"UN SISTEMA DE COMPUTACION NORMALMENTE
UTILIZA MEMORIA INTERNA DE ALTA",
                                    "DE ALTA VELOCIDAD (MEMORIA RAM) Y MEMORIA EN
MASA MAS LENTA.",1,15,10,15,30,69,61);

 TIEMPO(cuenta);
  VER_CADENA_CON(hwnd,0,127,0,0,0,0,"EN LA MEMORIA INTERNA LA VELOCIDAD DE
OPERACION ES BASICA EL PROGRAMA",
                    "Y CUALQUIER INFORMACION QUE ESTE UTILIZA SE ENCUENTRA
ALMACENADA AQUI",1,12,10,12,30,70,69);

 for(r=0;r<=3;r++){
 MessageBeep(-1);
 Brocha= CreateSolidBrush(RGB(0,255,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,390,170,490,230);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 TIEMPO(cuenta3);
 Brocha= CreateSolidBrush(RGB(0,127,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,390,170,490,230);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 SetTextColor(hdc,RGB(0,0,255));
 TextOut(hdc,398,175,"MEMORIA",7);
 TextOut(hdc,398,187,"INTERNA",7);
 TextOut(hdc,398,199,"( NUCLEO O",10);
 TextOut(hdc,400,211,"MOS )",5);
 TIEMPO(cuenta3);
 }

TIEMPO(cuenta);
      VER_CADENA_CON(hwnd,0,127,0,0,0,0,"ESTE TIPO DE MEMORIA REQUIERE
FORZOSAMENTE ENERGIA ELECTRICA",
                    "PARA CONCERBAR LA INFORMACION Y DATOS QUE
CONTENGA",1,12,10,12,30,60,50);

TIEMPO(cuenta);


  VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA MEMORIA INTERNA NO SE PUEDE UTILIZAR
COMO MEMORIA DE ALMACENAMIENTO",
```

```
                    "EN MASA,DEBIDO AL COSTO POR ALMACENAMIENTO DE BIT (MAS
ESPACIO EN MEMORIA)",1,12,10,12,30,70,74);

 TIEMPO(cuenta);
  VER_CADENA_CON(hwnd,255,0,0,0,0,0,"EL ALMACENAMIENTO EN MASA SE REFIERE A
LA MEMORIA QUE ES EXTERNA A",
                                    "LA COMPUTADORA CENTRAL",1,12,10,12,30,66,23);
 TIEMPO(cuenta);

 for(r=0;r<=3;r++){
 MessageBeep(-1);
 Brocha= CreateSolidBrush(RGB(0,255,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,372,350,508,435);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 TIEMPO(cuenta3);
 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,372,350,508,435);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 SetTextColor(hdc,RGB(0,0,255));
 TextOut(hdc,378,360,"ALMACENAMIENTO",14);
 TextOut(hdc,378,375,"EN MASA EXTERNO",15);
 TextOut(hdc,378,390,"(CINTA, DISCO,",14);
 TextOut(hdc,378,405,"MBM)",4);
 TIEMPO(cuenta3);
 }

 TIEMPO(cuenta);
   VER_CADENA_CON(hwnd,0,255,0,0,0,0,"ESTA MEMORIA TIENE LA CAPACIDAD DE
ALMACENAR MILLONES DE BITS DE",
                    "DATOS SIN NECESIDAD DE ENERGIA ELECTRICA. ESTA MEMORIA
ES MUY LENTA",1,12,10,12,30,64,67);

 TIEMPO(cuenta);
  VER_CADENA_CON(hwnd,0,255,0,0,0,0,"SE USA PARA ALMACENAR INFORMACION QUE
NO SERA UTILIZADA FRECUENTEMENTE",
                    "ESTA INFORMACION SE TRANSFIERE A LA MEMORIA
INTERNA CUANDO SE REQUIERA",1,12,10,12,30,70,70);

 for(r=0;r<=3;r++){
 MessageBeep(-1);
 Brocha= CreateSolidBrush(RGB(0,127,0));
 Brocha1= SelectObject(hdc,Brocha);
 Pluma= CreatePen(PS_SOLID,1,RGB(0,127,0));
 Pluma1= SelectObject(hdc,Pluma);
 Polygon(hdc,flechas3,10);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 TIEMPO(cuenta3);
 Brocha= CreateSolidBrush(RGB(255,255,0));
```

```
 Brocha1= SelectObject(hdc,Brocha);
 Pluma= CreatePen(PS_SOLID,1,RGB(255,255,0));
 Pluma1= SelectObject(hdc,Pluma);
 Polygon(hdc,flechas3,10);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 TIEMPO(cuenta3);
 }

 TIEMPO(cuenta);
    VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA CINTA Y DISCO MAGNETICO SON
DISPOSITIVOS DE ALMACENAMIENTO EN MASA",
                   "QUE SOM MENOS COSTOSOS POR BIT DE ALMACENAMIENTO QUE
LA MEMORIA INTERNA",1,12,10,12,30,69,71);
 TIEMPO(cuenta);
  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"OTRA CATEGORIA EN MEMORIA EN MASA ES
LA MEMORIA DE BURBUJA MAGNETICA (MBM)",
                   ",DISPOSITIVO SEMICONDUCTOR QUE UTILIZA
PRINCIPIOS MAGNETICOS AL ALMACENAR",1,12,10,12,30,74,73);

 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 var_w=4; j=4;

} //   TERMINA MODULO MEMORIA 2



//*****************************************************************************
****
//   FUNCION MODULO MEMORIA 3

void MOD_MEMORIA_3(HWND hwnd,HDC hdc){

int i,j,c1,c2,c3,r,c4,c5,c6;
HBRUSH Brocha, Brocha1;
HPEN Pluma1, Pluma;

 Brocha= CreateSolidBrush(RGB(127,127,127));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,0,0,645,445);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 SetCursor(Reloj_Arena);
 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,200,130,300,230,0,0,255,1);
 SetBkMode(hdc,TRANSPARENT);
 SetTextColor(hdc,RGB(191,191,191));
 TextOut(hdc,205,170,"MEMORIA DE",10);
 TextOut(hdc,235,185,"32 X 4",6);
```

```
 TIEMPO(cuenta);
 VER_CADENA_CON(hwnd,0,255,0,0,0,0,"AUNQUE CADA TIPO DE MEMORIA ES DIFERENTE
EN SU OPERACION INTERNA, CIERTOS",
                    "PRINCIPIOS BASICOS DE OPERACION SON LOS MISMOS
PARA CUALQUIER SISTEMA",1,12,10,12,30,73,69);
 TIEMPO(cuenta4);
   VER_CADENA_CON(hwnd,255,255,0,0,0,0,"TODO SISTEMA DE MEMORIA REQUIERE
VARIOS TIPOS DIFERENTES DE LINEAS DE",
                    "ENTRADA Y SALIDA.",1,12,10,12,30,69,17);
 TIEMPO(cuenta);

 SetTextColor(hdc,RGB(0,0,255));
 TextOut(hdc,145,138,"A4",2);
 TextOut(hdc,145,155,"A3",2);
 TextOut(hdc,145,172,"A2",2);
 TextOut(hdc,145,189,"A1",2);
 TextOut(hdc,145,206,"A0",2);
 TextOut(hdc,216,75,"I3",2);
 TextOut(hdc,236,75,"I2",2);
 TextOut(hdc,256,75,"I1",2);
 TextOut(hdc,276,75,"I0",2);
 TextOut(hdc,212,270,"O3",2);
 TextOut(hdc,232,270,"O2",2);
 TextOut(hdc,252,270,"O1",2);
 TextOut(hdc,272,270,"O0",2);
 TextOut(hdc,340,157,"R/W-",4);
 TextOut(hdc,340,191,"M.E",3);
 SetTextColor(hdc,RGB(127,0,0));
 TextOut(hdc,185,61,"ENTRADA DE DATOS",16);
 TextOut(hdc,30,165,"ENTRADA DE",10);
 TextOut(hdc,30,180,"DIRECCIONES",11);
 TextOut(hdc,193,288,"SALIDA DE DATOS",15);
 TextOut(hdc,379,157,"COMANDO DE LECTURA / ESCRITURA",30);
 TextOut(hdc,379,191,"MEMORIA HABILITADA (ENABLE)",27);

 Pluma= CreatePen(PS_SOLID,3,RGB(255,255,0));
 Pluma1= SelectObject(hdc,Pluma);
 i=0; j=0; c1=220; c2=215; c3=225; c4=146; c5=141; c6=151;
 MessageBeep(-1);
 for(r=0;r<=3;r++){
 MoveTo(hdc,195,c4+j);
 LineTo(hdc,165,c4+j);
 MoveTo(hdc,190,c5+j);
 LineTo(hdc,195,c4+j);
 LineTo(hdc,190,c6+j);
 MoveTo(hdc,c1+i,125);
 LineTo(hdc,c1+i,95);
 MoveTo(hdc,c2+i,120);
 LineTo(hdc,c1+i,125);
 LineTo(hdc,c3+i,120);
 MoveTo(hdc,c1+i,265);
 LineTo(hdc,c1+i,235);
 MoveTo(hdc,c2+i,260);
 LineTo(hdc,c1+i,265);
 LineTo(hdc,c3+i,260);
 i+=20; j+=17; }
 MoveTo(hdc,195,c4+j);
```

```
LineTo(hdc,165,c4+j);
MoveTo(hdc,190,c5+j);
LineTo(hdc,195,c4+j);
LineTo(hdc,190,c6+j);

i=0;
for(r=0;r<=1;r++){
MoveTo(hdc,305,164+i);
LineTo(hdc,335,164+i);
MoveTo(hdc,310,159+i);
LineTo(hdc,305,164+i);
LineTo(hdc,310,169+i);
i+=34; }

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

TIEMPO(cuenta);
 VER_CADENA_CON(hwnd,255,0,0,0,0,0,"CON ESTAS ENTRADAS Y SALIDAS QUE DEBE DE
CONTENER CUALQUIER SISTEMA DE",
                                   "DE MEMORIA, DEBE DE REALIZAR LAS SIGUIENTES
FUNCIONES :",1,12,10,12,30,70,55);

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Pluma= CreatePen(PS_SOLID,2,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);
MessageBeep(-1);
Rectangle(hdc,17,310,567,440);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
TIEMPO(cuenta4);
 VER_CADENA_SIN(hwnd,25,315, "SELECCIONAR LA DIRECCION EN LA MEMORIA QUE
ESTA SIENDO ACCESADA",63,0,0,255);
     VER_CADENA_SIN(hwnd,25,330, "PARA UNA OPERACION DE LECTURA O
ESCRITURA.",42,0,0,255);
 VER_CADENA_SIN(hwnd,25,360, "SELECCIONAR UNA OPERACION DE LECTURA O BIEN
DE ESCRITURA PARA SER",65,0,0,255);
 VER_CADENA_SIN(hwnd,25,375, "EFECTUADA.",10,0,0,255);
 VER_CADENA_SIN(hwnd,25,405, "PROPORCIONAR LOS DATOS DE ENTRADA PARA SER
ALMACENADOS EN LA",60,0,0,255);
     VER_CADENA_SIN(hwnd,25,420, "MEMORIA DURANTE UNA OPERACION DE
LECTURA.",41,0,0,255);
TIEMPO(cuenta4);
Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);
Pluma= CreatePen(PS_SOLID,2,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma),
MessageBeep(-1);
Rectangle(hdc,17,310,567,440);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma);
 VER_CADENA_SIN(hwnd,25,315, "CONTENER LOS DATOS DE SALIDA QUE VIENEN DE LA
MEMORIA DURANTE",61,0,0,255);
```

```
 VER_CADENA_SIN(hwnd,25,330, "UNA OPERACION DE LECTURA.",25,0,0,255);
  VER_CADENA_SIN(hwnd,25,360, "ACTIVAR O DESACTIVAR LA MEMORIA DE MANERA
QUE RESPONDA O NO A LAS",65,0,0,255);
   VER_CADENA_SIN(hwnd,25,375, "ENTRADAS DE DIRECCION Y AL COMANDO DE
LECTURA / ESCRITURA.",58,0,0,255);

 TIEMPO(cuenta4);
 Brocha= CreateSolidBrush(RGB(0,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Pluma= CreatePen(PS_SOLID,1,RGB(255,255,0));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,17,310,567,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA FIGURA REPRESENTA UNA MEMORIA DEC
32 X 4 QUE ALMACENA 32 PALABRAS DE",
                 "4 BITS, DE AHI QUE SEAN 4 LINEAS DE ENTRADA DE DATOS Y 4 DE
SALIDA.",1,12,10,12,30,71,67);

 TIEMPO(cuenta2);
 Pluma= CreatePen(PS_SOLID,3,RGB(255,0,0));
 Pluma1= SelectObject(hdc,Pluma);
 i=0; j=0; c1=220; c2=215; c3=225;
 MessageBeep(-1);
 for(r=0;r<=3;r++){
 MoveTo(hdc,c1+i,125);
 LineTo(hdc,c1+i,95);
 MoveTo(hdc,c2+i,120);
 LineTo(hdc,c1+i,125);
 LineTo(hdc,c3+i,120);
 MoveTo(hdc,c1+i,265);
 LineTo(hdc,c1+i,235);
 MoveTo(hdc,c2+i,260);
 LineTo(hdc,c1+i,265);
 LineTo(hdc,c3+i,260);
 i+=20; j+=17; }

 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 TIEMPO(cuenta4);
  VER_CADENA_SIN(hwnd,25,313, "DURANTE UNA OPERACION DE ESCRITURA LOS DATOS
QUE SE ALMACENARAN",63,255,0,0);
  VER_CADENA_SIN(hwnd,25,328, "EN LA MEMORIA TIENEN QUE SER APLICADOS A LAS
LINEAS DE ENTRADA DE",65,255,0,0);
  VER_CADENA_SIN(hwnd,25,343, "DATOS. DURANTE UNA OPERACION DE LECTURA LA
PALABRA QUE ES LEIDA DE",66,255,0,0);
  VER_CADENA_SIN(hwnd,25,358, "LA MEMORIA SE PRESENTA EN LAS LINEAS DE SALIDA
DE DATOS",55,255,0,0);
   VER_CADENA_SIN(hwnd,25,378, "ESTA MEMORIA TIENE 32 LOCALIZACIONES DE
ALMACENAMIENTO DISTINTAS,",65,255,0,0);
  VER_CADENA_SIN(hwnd,25,393, "QUE VAN DE 00000(base 2) a 11111(base 2), DE 0 A 31 EN
DECIMAL.",63,255,0,0);
   VER_CADENA_SIN(hwnd,25,408, "POR LO TANTO UTILIZAMOS 5 ENTRADAS DE
DIRECCION PARA ESPECIFICAR",64,255,0,0);
```

```
      VER_CADENA_SIN(hwnd,25,423,  "UNA  DE  LAS  32  LOCALIZACIONES  DE
DIRECCION.",42,255,0,0);
 TIEMPO(cuenta2);
 Pluma= CreatePen(PS_SOLID,3,RGB(0,255,255));
 Pluma1= SelectObject(hdc,Pluma);
 i=0; j=0; c1=220; c2=215; c3=225; c4=146; c5=141; c6=151;
 MessageBeep(-1);
 for(r=0;r<=3;r++){
 MoveTo(hdc,195,c4+j);
 LineTo(hdc,165,c4+j);
 MoveTo(hdc,190,c5+j);
 LineTo(hdc,195,c4+j);
 LineTo(hdc,190,c6+j);
 j+=17; }
 MoveTo(hdc,195,c4+j);
 LineTo(hdc,165,c4+j);
 MoveTo(hdc,190,c5+j);
 LineTo(hdc,195,c4+j);
 LineTo(hdc,190,c6+j);

 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 TIEMPO(cuenta4);
 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);
 Pluma= CreatePen(PS_SOLID,2,RGB(0,0,255));
 Pluma1= SelectObject(hdc,Pluma);
 Rectangle(hdc,17,310,567,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

  VER_CADENA_SIN(hwnd,25,313, "PARA ACCESAR UNA DE LAS DIRECCIONES DE LA
MEMORIA EN UNA OPERACION",66,127,0,0);
  VER_CADENA_SIN(hwnd,25,328, "EL CODIGO DE DIRECCION DE 5 BITS SE APLICA A LA
ENTRADA DE DIRCCION",67,127,0,0);
    VER_CADENA_SIN(hwnd,25,343, "LA LINEA DE ENTRADA R/W DETERMINA QUE
OPERACION DE LA MEMORIA SE VA",67,127,0,0);
 VER_CADENA_SIN(hwnd,25,358, "A EJECUTAR.",11,127,0,0);
 TIEMPO(cuenta2);
 Pluma= CreatePen(PS_SOLID,3,RGB(0,255,0));
 Pluma1= SelectObject(hdc,Pluma);
 MessageBeep(-1);
 MoveTo(hdc,305,164);
 LineTo(hdc,335,164);
 MoveTo(hdc,310,159);
 LineTo(hdc,305,164);
 LineTo(hdc,310,169);
 TIEMPO(cuenta2);

  VER_CADENA_SIN(hwnd,25,378, "MUCHOS SISTEMAS DE MEMORIA TIENEN ALGUN
MEDIO PARA DESACTIVAR POR",65,127,0,0);
 VER_CADENA_SIN(hwnd,25,393, "COMPLETO TODA O PARTE DE LA MEMORIA, Y ASI NO
RESPONDA A ENTRADAS",65,127,0,0);
 VER_CADENA_SIN(hwnd,25,408, "ESTE TIPO DE ENTRADA ES DE UTILIDAD CUANDO SE
COMBINAN VARIOS",61,127,0,0);
```

```
 VER_CADENA_SIN(hwnd,25,423, "MODULOS DE MEMORIA PARA FORMAR UNA MEMORIA
MAYOR.",49,127,0,0);
 TIEMPO(cuenta2);
 i=34;
 MessageBeep(-1);
 MoveTo(hdc,305,164+i);
 LineTo(hdc,335,164+i);
 MoveTo(hdc,310,159+i);
 LineTo(hdc,305,164+i);
 LineTo(hdc,310,169+i);

 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma);

 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 SelectObject(hdc,PreFuente);
 DeleteObject(Fuente);

 var_w=6; j=4;

} //  TERMINA MODULO MEMORIA 3


//**********************************************************************************
****
// FUNCION DE MODULO UNIDAD DE CONTROL

 void MOD_CONTROL(HWND hwnd,HDC hdc){

int x;
 HBRUSH Brocha, Brocha1;

 Brocha= CreateSolidBrush(RGB(0,127,127));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,0,0,645,445);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 SetCursor(Reloj_Arena);

 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,340,150,440,300,0,0,127,2);
 CREA_RECTANGULO_BIDIMENSIONAL(hwnd,40,150,140,300,0,127,0,2);

 SetTextColor(hdc,RGB(127,0,0));
 SetBkColor(hdc,RGB(191,191,191));

 TextOut(hdc,30,90,"UNIDAD DE CONTROL",17);
 TextOut(hdc,330,90,"UNIDAD DE MEMORIA",17);

 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,45,225,135,270);
```

```
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);

  SetBkColor(hdc,RGB(0,127,0));
  TextOut(hdc,55,205,"C.LOGICOS",9);

CADENA_BITS_1(hwnd,hdc);

SetCursor(Reloj_Arena);
MODULOS(hdc);
FLECHAS_DATOS(hdc,1,11,3);
FLECHAS_CONTROL(hdc,2,5,3);

VER_CADENA_CON(hwnd,0,255,0,0,0,0,"LOS CIRCUITOS  LOGICOS DE LA UNIDAD DE
CONTROL SON LOS ENCARGADOS",
                    "DE DECODIFICAR LA PALABRA DE INSTRUCCION (DIGITOS
BINARIOS)",1,12,10,12,30,66,59);

TIEMPO(cuenta);

for(p=0;p<=3;p++){
INT_MODULOS(hdc,1,2,1);
 TIEMPO(cuenta);
INT_MODULOS(hdc,3,2,2);
 TIEMPO(cuenta);
}

TIEMPO(cuenta);

VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA  UNIDAD  DE  CONTROL  UTILIZA  ESTA
INFORMACION PARA GENERAR",
                    "LAS  SEÑALES  NECESARIAS  PARA  EJECUTAR  LA
INSTRUCCION",1,12,10,12,30,58,51);

TIEMPO(cuenta);

for(p=0;p<=2;p++){
 MessageBeep(0);
for(x=1;x<=4;x++){
FLECHAS_CONTROL(hdc,2,x,3);
 TIEMPO(cuenta3);
FLECHAS_CONTROL(hdc,1,x,3);
 TIEMPO(cuenta3);
} }

BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);

var_w=2;  j=3;

}// TERMINA MODULO CONTROL
```

```
//**********************************************************************************
****
// FUNCION DE MODULO UNIDAD DE ENTRADA

 void MOD_ENTRADA(HWND hwnd,HDC hdc){

int y=0,a=0;
char *s;
HPEN Pluma1, Pluma;
HBRUSH Brocha, Brocha1;

  Brocha= CreateSolidBrush(RGB(127,127,127));
  Brocha1= SelectObject(hdc,Brocha);
  Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
  Pluma1= SelectObject(hdc,Pluma);

  Rectangle(hdc,0,0,645,445);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);


SetCursor(Reloj_Arena);
hbmpU_Ent_Sal = LoadBitmap(hInst, "MyBitmap4");
hdcU_Ent_Sal = CreateCompatibleDC(hdc);
hbmpU_Ent_Sal1 = SelectObject(hdcU_Ent_Sal,hbmpU_Ent_Sal);

while(a<=160){
BitBlt(hdc,185,185,271,a,hdcU_Ent_Sal,8,123,SRCCOPY);
a+=2; }

VER_CADENA_SIN(hwnd,50,375,"LA INFORMACION DE ENTRADA ES AQUELLA QUE SE
RECIBE DEL EXTERIOR",63,128,0,0);
VER_CADENA_SIN(hwnd,50,395,"LA ENTRADA LLEGA EN FORMA DE SEÑALES
PROVENIENTES DE DIFERENTES",63,128,0,0);
VER_CADENA_SIN(hwnd,50,415,"DISPOSITIVOS",12,128,0,0);

BitBlt(hdc,50,70,95,101,hdcU_Ent_Sal,34,309,SRCCOPY);
BitBlt(hdc,50,220,95,101,hdcU_Ent_Sal,162,310,SRCCOPY);
BitBlt(hdc,300,20,95,101,hdcU_Ent_Sal,280,310,SRCCOPY);
BitBlt(hdc,530,220,95,101,hdcU_Ent_Sal,401,309,SRCCOPY);
BitBlt(hdc,530,70,95,101,hdcU_Ent_Sal,520,309,SRCCOPY);


TIEMPO(cuenta);

  Brocha= CreateSolidBrush(RGB(0,0,255));
  Brocha1= SelectObject(hdc,Brocha);

MessageBeep(0);
while(y<=5){
//**************************************************
Pluma= CreatePen(PS_SOLID,3,RGB(0,0,255));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,347,158);
LineTo(hdc,347,125),
```

```
Ellipse(hdc,344,155,350,161);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,347,158);
LineTo(hdc,347,125);
Ellipse(hdc,344,155,350,161);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta1);
//*********************************************
Pluma= CreatePen(PS_SOLID,3,RGB(0,0,255));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,269,173);
LineTo(hdc,150,113);
Ellipse(hdc,266,170,272,176);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,269,173);
LineTo(hdc,150,113);
Ellipse(hdc,266,170,272,176);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta1);
//*********************************************
Pluma= CreatePen(PS_SOLID,3,RGB(0,0,255));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,259,232);
LineTo(hdc,154,287);
Ellipse(hdc,256,229,262,235);
SelectObject(hdc,Pluma1),
DeleteObject(Pluma1);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,259,232);
LineTo(hdc,154,287);
Ellipse(hdc,256,229,262,235);
SelectObject(hdc,Pluma1),
DeleteObject(Pluma1);
TIEMPO(cuenta1);
//*********************************************
Pluma= CreatePen(PS_SOLID,3,RGB(0,0,255));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,393,216);
LineTo(hdc,520,277);
Ellipse(hdc,390,213,396,219);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,393,216);
```

```
LineTo(hdc,520,277);
Ellipse(hdc,390,213,396,219);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta1);
//**********************************************
Pluma= CreatePen(PS_SOLID,3,RGB(0,0,255));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,388,181);
LineTo(hdc,510,113);
Ellipse(hdc,385,178,391,185);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,3,RGB(127,127,127));
Pluma1= SelectObject(hdc,Pluma);
MoveTo(hdc,388,181);
LineTo(hdc,510,113);
Ellipse(hdc,385,178,391,185);
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
TIEMPO(cuenta1);

y++;
}
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TIEMPO(cuenta3);
Pluma= CreatePen(PS_SOLID,4,RGB(128,0,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,347,158);
LineTo(hdc,347,125);
Ellipse(hdc,344,155,350,161);
MoveTo(hdc,269,173);
LineTo(hdc,150,113);
Ellipse(hdc,266,170,272,176);
MoveTo(hdc,259,232);
LineTo(hdc,154,287);
Ellipse(hdc,256,229,262,235);
MoveTo(hdc,393,216);
LineTo(hdc,520,277);
Ellipse(hdc,390,213,396,219);
MoveTo(hdc,388,181);
LineTo(hdc,510,113);
Ellipse(hdc,385,178,391,185);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);
SelectObject(hdcU_Ent_Sal,hbmpU_Ent_Sal1);
DeleteDC(hdcU_Ent_Sal);
BOTONES(hwnd,1);
BOTONES(hwnd,3);
BOTONES(hwnd,5);
SetCursor(Cursor_Cruz);
ULTIMA_IMAGEN(hwnd);
var_w=2; j=5;
```

```
}    // TERMINA MODULO UNIDAD DE ENTRADA

// MODULO UNIDAD DE SALIDA

void MOD_SALIDA(HWND hwnd, HDC hdc){

 int f=0,a=0;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma, Pluma1;

  POINT flecha4[7];

 flecha4[0].x = 380;  flecha4[0].y = 250;
 flecha4[1].x = 380;  flecha4[1].y = 220;
 flecha4[2].x = 460;  flecha4[2].y = 220;
 flecha4[3].x = 460;  flecha4[3].y = 210;
 flecha4[4].x = 490;  flecha4[4].y = 235;
 flecha4[5].x = 460;  flecha4[5].y = 260;
 flecha4[6].x = 460;  flecha4[6].y = 250;

 Brocha= CreateSolidBrush(RGB(0,64,128));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,0,0,645,445);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

SetCursor(Reloj_Arena);
hbmpU_Ent_Sal = LoadBitmap(hInst, "MyBitmap4");
hdcU_Ent_Sal = CreateCompatibleDC(hdc);
hbmpU_Ent_Sal1 = SelectObject(hdcU_Ent_Sal,hbmpU_Ent_Sal);

while(a<=162){
BitBlt(hdc,100,150,272,a,hdcU_Ent_Sal,290,124,SRCCOPY);
a+=2; }

VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA UNIDAD DE SALIDA ESTA COMPUESTA POR
DIVERSOS DISPOSITIVOS",
                    "QUE PUEDEN TRANSFERIR DATOS O INFORMACION AL
EXTERIOR",1,12,10,12,30,60,53);


 Brocha=CreateSolidBrush(RGB(0,0,127));
 Pluma= CreatePen(PS_SOLID,3,RGB(0,0,0));
 Brocha1= SelectObject(hdc,Brocha);
 Pluma1= SelectObject(hdc,Pluma);

 Polygon(hdc,flecha4,7);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);


while(f!=5){

TIEMPO(cuenta);
BitBlt(hdc,500,185,99,113,hdcU_Ent_Sal,4,3,SRCCOPY);
```

```
TIEMPO(cuenta);
BitBlt(hdc,500,185,99,113,hdcU_Ent_Sal,112,4,SRCCOPY);

TIEMPO(cuenta);
BitBlt(hdc,500,185,99,113,hdcU_Ent_Sal,221,4,SRCCOPY);

TIEMPO(cuenta);
BitBlt(hdc,500,185,99,113,hdcU_Ent_Sal,331,5,SRCCOPY);

f++; }

VER_CADENA_SIN(hwnd,50,375,"LA UNIDAD DE SALIDA ES CONTROLADA POR MEDIO DE
SEÑALES DE LA" ,60,0,255,0);
VER_CADENA_SIN(hwnd,50,395,"UNIDAD DE CONTROL",17,0,255,0);
 SelectObject(hdcU_Ent_Sal,hbmpU_Ent_Sal1);
 DeleteDC(hdcU_Ent_Sal);
 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 var_w=2; j=6;
}       // TERMINA MODULO UNIDAD SE SALIDA


//**********************************************************************************
****
// FUNCION MODULO CPU 1

 void MOD_CPU_1(HWND hwnd ,HDC hdc){

int a;
HBRUSH Brocha, Brocha1;

POINT flecha1[7];
POINT flecha2[7];

  flecha1[0].x = 145;  flecha1[0].y = 163;
  flecha1[1].x = 175;  flecha1[1].y = 138;
  flecha1[2].x = 175;  flecha1[2].y = 148;
  flecha1[3].x = 220;  flecha1[3].y = 148;
  flecha1[4].x = 220;  flecha1[4].y = 178;
  flecha1[5].x = 175;  flecha1[5].y = 178;
  flecha1[6].x = 175;  flecha1[6].y = 188;

  flecha2[0].x = 145;  flecha2[0].y = 278;
  flecha2[1].x = 175;  flecha2[1].y = 253;
  flecha2[2].x = 175;  flecha2[2].y = 263;
  flecha2[3].x = 220;  flecha2[3].y = 263;
  flecha2[4].x = 220;  flecha2[4].y = 293;
  flecha2[5].x = 175;  flecha2[5].y = 293;
  flecha2[6].x = 175;  flecha2[6].y = 303;

  Brocha= CreateSolidBrush(RGB(0,127,0));
  Brocha1= SelectObject(hdc,Brocha);

  Rectangle(hdc,0,0,645,445);
```

```
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetCursor(Reloj_Arena);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,40,100,140,350,0,0,127,2);

SetTextColor(hdc,RGB(0,0,255));
SetBkColor(hdc,RGB(191,191,191));

TextOut(hdc,33,360,"MICROPROCESADOR",15);
TextOut(hdc,190,385,"UNIDAD DE CONTROL",17);
TextOut(hdc,190,410,"UNIDAD ARITMETICO/LOGICA",24);

SetTextColor(hdc,RGB(0,0,0));
SetBkMode(hdc,TRANSPARENT);

Brocha= CreateSolidBrush(RGB(0,255,0));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,60,120,120,215);
Polygon(hdc,flecha1,7);
TextOut(hdc,80,160,"U.C",3);
Ellipse(hdc,170,385,180,395);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= CreateSolidBrush(RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,60,235,120,330);
Polygon(hdc,flecha2,7);
TextOut(hdc,80,275,"ALU",3);
Ellipse(hdc,170,410,180,420);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

TIEMPO(cuenta4);

VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA UAL Y LA U. DE CONTROL SE ENCUENTRAN
COMBINADAS EN UNA UNIDAD",
                "DENOMINADA UNIDAD CENTRAL DE PROCESAMIENTO (CPU O
UCP)",1,35,10,35,30,64,54);

MessageBeep(0);
for(r=0;r<=1;r++){
Brocha= CreateSolidBrush(RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,60,120,120,215);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TextOut(hdc,80,160,"U C",3);
TIEMPO(cuenta2);
Brocha= CreateSolidBrush(RGB(0,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,60,120,120,215);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TextOut(hdc,80,160,"U.C",3);
```

```
TIEMPO(cuenta3);
Brocha= CreateSolidBrush(RGB(0,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,60,235,120,330);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TextOut(hdc,80,275,"ALU",3);
TIEMPO(cuenta2);
Brocha= CreateSolidBrush(RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,60,235,120,330);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TextOut(hdc,80,275,"ALU",3);
TIEMPO(cuenta3);
}
TIEMPO(cuenta4);

Brocha= CreateSolidBrush(RGB(191,191,191));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,235,80,620,300);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

  VER_CADENA_SIN(hwnd,240,100,"EL CPU ES REALMENTE EL CEREBRO DE LA PC,
ES",43,127,0,0);
  VER_CADENA_SIN(hwnd,240,120,"DECIR DE LA COMPUTADORA. NORMALMENTE EL
CPU",43,127,0,0);
    VER_CADENA_SIN(hwnd,240,140,"SE ENCUENTRA CONTENIDO EN UN SOLO
INTEGRADO",43,127,0,0);
 VER_CADENA_SIN(hwnd,240,160,"LSI LLAMADO MICROPROCESADOR.",28,127,0,0);
    VER_CADENA_SIN(hwnd,240,200,"SUS CARACTERISTICAS DETERMINAN LAS
CAPACIDADES",46,0,0,127);
  VER_CADENA_SIN(hwnd,240,220,"DE LA MICROCOMPUTADORA (PC),COMO ES LA
MEMORIA",46,0,0,127);
  VER_CADENA_SIN(hwnd,240,240,", LA VELOCIDAD Y EL TIPO DE SINCRONIZACION
E/S",46,0,0,127);
 VER_CADENA_SIN(hwnd,240,260,"QUE DEBE UTILIZARSE",19,0,0,127);

TIEMPO(cuenta6);
Brocha= CreateSolidBrush(RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,235,80,620,345);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

  VER_CADENA_SIN(hwnd,240,100,"EL MICROPROCESADOR REALIZA VARIAS FUNCIONES
:",45,127,0,0);
  VER_CADENA_SIN(hwnd,240,140,"OFRECER SEÑALES DE DISTRIBUCION Y CONTROL
PARA",46,255,255,0);
    VER_CADENA_SIN(hwnd,240,160,"TODOS LOS ELEMENTOS DE LA
MICROCOMPUTADORA.",43,255,255,0);
   VER_CADENA_SIN(hwnd,240,180,"BUSCAR Y TRAER INSTRUC. Y DATOS DE LA
MEMORIA",45,0,0,255);
   VER_CADENA_SIN(hwnd,240,200,"TRANSFERENCIA DE DATOS HACIA Y DESDE
DISPOSI_",45,255,255,0);
```

```
 VER_CADENA_SIN(hwnd,240,220,"TIVOS DE ENTRADA/SALIDA ",24,255,255,0);
 VER_CADENA_SIN(hwnd,240,240,"DECODIFICACION DE INSTRUCCIONES.",32,0,0,255);
    VER_CADENA_SIN(hwnd,240,260,"REALIZACION DE OPERACIONES ARITMET. Y
LOGICAS",45,255,255,0);
 VER_CADENA_SIN(hwnd,240,280,"SOLICITAIDAS POR LAS INSTRUCCIONES",33,255,255,0);
  VER_CADENA_SIN(hwnd,240,300,"RESPUESTA A SEÑALES DE CONTROL GENERADAS
EN",43,0,0,255);
 VER_CADENA_SIN(hwnd,240,320,"E/S COMO RESET E INTERRUPT.",27,0,0,255);

 TIEMPO(cuenta4);
 BOTONES(hwnd,5);
 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 var_w=2; j=1;

} //  TERMINA FUNCION MODULO CPU 1


//*****************************************************************************
//  FUNCION MODULO CPU 2

 void MOD_CPU_2(HWND hwnd ,HDC hdc){

 int w,q=70;
 HBRUSH Brocha, Brocha1;
 HPEN Pluma1, Pluma;

POINT flechas1[7];
POINT flechas2[10];
POINT flechas3[10];

 flechas1[0].x = 250;  flechas1[0].y = 110;
 flechas1[1].x = 330+q;  flechas1[1].y = 110;
 flechas1[2].x = 330+q;  flechas1[2].y = 100;
 flechas1[3].x = 360+q;  flechas1[3].y = 125;
 flechas1[4].x = 330+q;  flechas1[4].y = 150;
 flechas1[5].x = 330+q;  flechas1[5].y = 140;
 flechas1[6].x = 250;  flechas1[6].y = 140;

 flechas2[0].x = 250;  flechas2[0].y = 205;
 flechas2[1].x = 280;  flechas2[1].y = 180;
 flechas2[2].x = 280;  flechas2[2].y = 190;
 flechas2[3].x = 330+q;  flechas2[3].y = 190;
 flechas2[4].x = 330+q;  flechas2[4].y = 180;
 flechas2[5].x = 360+q;  flechas2[5].y = 205;
 flechas2[6].x = 330+q;  flechas2[6].y = 230;
 flechas2[7].x = 330+q;  flechas2[7].y = 220;
 flechas2[8].x = 280;  flechas2[8].y = 220;
 flechas2[9].x = 280;  flechas2[9].y = 230;

 flechas3[0].x = 250;  flechas3[0].y = 265;
 flechas3[1].x = 280;  flechas3[1].y = 240;
 flechas3[2].x = 280;  flechas3[2].y = 250;
flechas3[3].x = 330+q;  flechas3[3].y = 250;
flechas3[4].x = 330+q;  flechas3[4].y = 240;
```

```
flechas3[5].x = 360+q;  flechas3[5].y = 265;
flechas3[6].x = 330+q;  flechas3[6].y = 290;
flechas3[7].x = 330+q;  flechas3[7].y = 280;
flechas3[8].x = 280;  flechas3[8].y = 280;
flechas3[9].x = 280;  flechas3[9].y = 290;

Brocha= CreateSolidBrush(RGB(0,127,0));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,0,0,645,445);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetCursor(Reloj_Arena);

CREA_RECTANGULO_BIDIMENSIONAL(hwnd,40,100,240,300,127,0,0,1);

Pluma= CreatePen(PS_SOLID,1,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,140,100);
LineTo(hdc,140,230);
MoveTo(hdc,40,230);
LineTo(hdc,240,230);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

 Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));// estas linea de pluma son opcionales, ya que el color
por default de la pluma es negro
 Pluma1= SelectObject(hdc,Pluma);
 Brocha= CreateSolidBrush(RGB(255,255,0));
 Brocha1= SelectObject(hdc,Brocha);

  Polygon(hdc,flechas1,7);
  Polygon(hdc,flechas2,10);
  Polygon(hdc,flechas3,10);

 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);
 SelectObject(hdc,Pluma1);
 DeleteObject(Pluma1);

 SetTextColor(hdc,RGB(191,191,191));
 SetBkColor(hdc,RGB(0,0,255));
 TextOut(hdc,80,302,"MICROPROCESADOR",15);

 SetTextColor(hdc,RGB(0,255,0));
 SetBkMode(hdc,TRANSPARENT);
 TextOut(hdc,80,170,"UAL",3);
 TextOut(hdc,145,150,"SECCION DE",10);
 TextOut(hdc,145,165,"REGISTROS",9);
 TextOut(hdc,50,250,"SECCION CONTROL",15);
 TextOut(hdc,50,265,"Y TEMPORALIZACION",17);
 SetTextColor(hdc,RGB(127,0,0));
 TextOut(hdc,256,120,"LINEA DE DIRECCIONES",20);
 TextOut(hdc,280,200,"LINEA DE DATOS",14);
 TextOut(hdc,275,260,"LINEA DE CONTROL",16);
```

```
      VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA LOGICA DEL MICROPROCESADOR
PODEMOS DIVIDIRLA EN 3 AREAS:",
                  "",1,12,10,12,30,60,0);
 TIEMPO(cuenta4);

 Brocha= CreateSolidBrush(RGB(0,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,120,325,520,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

          VER_CADENA_CON(hwnd,0,127,0,0,0,0,"LA SECCION DE CONTROL Y
TEMPORALIZACION",
                  "",1,35,20,12,30,39,0);

 TIEMPO(cuenta4);

 for(w=0;w<=1;w++){
 MessageBeep(0);
 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);
 TextOut(hdc,50,250,"SECCION CONTROL",15);
 TextOut(hdc,50,265,"Y TEMPORALIZACION",17);
 TIEMPO(cuenta2);
 MessageBeep(0);
 SetTextColor(hdc,RGB(0,255,0));
 SetBkMode(hdc,TRANSPARENT);
 TextOut(hdc,50,250,"SECCION CONTROL",15);
 TextOut(hdc,50,265,"Y TEMPORALIZACION",17);
 TIEMPO(cuenta2);
 }

 TIEMPO(cuenta4);
       VER_CADENA_SIN(hwnd,136,340,"SU FUNCION PRINCIPAL CONSISTE EN
BUSCAR,TRAER",45,191,191,191);
   VER_CADENA_SIN(hwnd,136,360,"Y DECODIFICAR INSTRUCCIONES DE LA MEMORIA
DEL",45,191,191,191);
    VER_CADENA_SIN(hwnd,136,380,"PROGRAMA PARA GENERAR LAS SEÑALES DE
CONTROL",44,191,191,191);
     VER_CADENA_SIN(hwnd,136,400,"NECESARIAS QUE REQUIERE LA UAL Y LA
SECCION",43,191,191,191);
      VER_CADENA_SIN(hwnd,136,420,"DE REGISTROS PARA EJECUTAR ESTAS
INSTRUCCIONES",46,191,191,191);

 TIEMPO(cuenta4);

 Brocha= CreateSolidBrush(RGB(0,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,120,325,520,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA SECCION DE REGISTROS:",
                  "",1,35,20,12,30,25,0);

 TIEMPO(cuenta4);
```

```
for(w=0;w<=1;w++){
MessageBeep(0);
SetTextColor(hdc,RGB(0,0,0));
SetBkMode(hdc,TRANSPARENT);
TextOut(hdc,145,150,"SECCION DE",10);
TextOut(hdc,145,165,"REGISTROS",9);
TIEMPO(cuenta2);
MessageBeep(0);
SetTextColor(hdc,RGB(0,255,0));
SetBkMode(hdc,TRANSPARENT);
TextOut(hdc,145,150,"SECCION DE",10);
TextOut(hdc,145,165,"REGISTROS",9);
TIEMPO(cuenta2);
}

TIEMPO(cuenta4);
  VER_CADENA_SIN(hwnd,136,340,"DENTRO DEL MICROPROCESAADOR LA OPERACION
MAS",44,191,191,191);
  VER_CADENA_SIN(hwnd,136,360,"COMUN ES LA TRASNFERENCIA DE INFORMACION DE
UN",46,191,191,191);
    VER_CADENA_SIN(hwnd,136,380,"REGISTRO A OTRO. EL NUMERO Y TIPO DE
REGISTRO",45,191,191,191);
        VER_CADENA_SIN(hwnd,136,400,"DEPENDE DE LA ARQUITECTURA DEL
MICROPROC.",41,191,191,191);
    VER_CADENA_SIN(hwnd,136,420,"SIN EMBARGO LA FUNCION DE LOS DIFERENTES
TIPOS",46,191,191,191);
 TIEMPO(cuenta6);
 Brocha= CreateSolidBrush(RGB(0,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,120,325,520,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

        VER_CADENA_SIN(hwnd,136,340,"ES LA MISMA (ALMACENAR DATOS,
DIRECCIONES,",42,191,191,191);
  VER_CADENA_SIN(hwnd,136,360,"CODIGOS DE INSTRUCCION E INFORMACION ACERCA
DE",46,191,191,191);
    VER_CADENA_SIN(hwnd,136,380,"LAS DIREC. DE DIVERSAS OPERACIONES DEL
CPU",42,191,191,191);

 TIEMPO(cuenta4);

 Brocha= CreateSolidBrush(RGB(0,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,120,325,520,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

 VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA UNIDAD ARITMETICO / LOGICA:",
                "",1,35,20,12,30,30,0);

 TIEMPO(cuenta4);

 for(w=0;w<=1;w++){
 MessageBeep(0);
 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);
 TextOut(hdc,80,170,"UAL",3);
```

```
 TIEMPO(cuenta2);
 MessageBeep(0);
 SetTextColor(hdc,RGB(0,255,0));
 SetBkMode(hdc,TRANSPARENT);
 TextOut(hdc,80,170,"UAL",3);
 TIEMPO(cuenta2);
 }

 TIEMPO(cuenta4);
   VER_CADENA_SIN(hwnd,136,340,"LAS OPERACIONES DE LA ALU PUEDEN IMPLICAR
DOS",45,191,191,191);
  VER_CADENA_SIN(hwnd,136,360,"OPERANDOS, COMO EL ACUMULADOR Y LA PALABRA
DE",45,191,191,191);
   VER_CADENA_SIN(hwnd,136,380,"DATOS DE LA MEMORIA O BIEN, EL ACUMULADOR
Y",43,191,191,191);
 VER_CADENA_SIN(hwnd,136,400,"OTRO REGISTRO INTERNO DEL CPU.",30,191,191,191);
 TIEMPO(cuenta6);
 Brocha= CreateSolidBrush(RGB(0,0,255));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,120,325,520,440);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

  VER_CADENA_SIN(hwnd,136,340,"ALGUNAS OPERACIONES COMPRENDEN SOLAMENTE
UN",44,191,191,191);
    VER_CADENA_SIN(hwnd,136,360,"OPERANDO,COMO EL REGISTRO ACUMULADOR,
REGISTRO",46,191,191,191);
 VER_CADENA_SIN(hwnd,136,380,"O PALABRA DE LA MEMORIA",23,191,191,191);
   VER_CADENA_SIN(hwnd,136,400,"LAS FUNCIONES DE LA UAL LAS DETERMINAN
SEÑALES",46,191,191,191);
 VER_CADENA_SIN(hwnd,136,420,"DE LA UNIDAD DE CONTROL",23,191,191,191);
 TIEMPO(cuenta6);

 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);
 SelectObject(hdc,PreFuente);
 DeleteObject(Fuente);

 var_w=4; j=1;

} // TERMINA MODULO CPU 2


//  MODULO  UNIDAD ARITMETICA LOGICA

 void MOD_ALU(HWND hwnd, HDC hdc){

int t,v,z,w,f,e,g,h,i,n,r;

char cadena[11]="1001      ";
char cadena1[11]="1101      ";
char cadena2[11]="1011      ";
char cadena3[11]="11000     ";
char *s,*d;
```

```
HBRUSH Brocha, Brocha1;
HPEN Pluma1, Pluma;

Brocha= CreateSolidBrush(RGB(127,0,0));
Brocha1= SelectObject(hdc,Brocha);

Rectangle(hdc,0,0,645,445);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetCursor(Reloj_Arena);

CREA_RECTANGULO_BIDIMENSIONAL(hwnd,40,150,140,300,0,127,0,2);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,220,130,420,320,0,0,127,2);
CREA_RECTANGULO_BIDIMENSIONAL(hwnd,500,170,600,280,191,191,191,2);
//-----------------------------

Brocha= CreateSolidBrush(RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,150,400,180);
Ellipse(hdc,150,380,160,390);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= CreateSolidBrush(RGB(0,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,280,205,360,245);
Ellipse(hdc,150,400,160,410);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= CreateSolidBrush(RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,270,400,300);
Ellipse(hdc,150,420,160,430);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

SetTextColor(hdc,RGB(0,0,0));
SetBkColor(hdc,RGB(191,191,191));
TextOut(hdc,30,100,"UNIDAD DE MEMORIA",17);
TextOut(hdc,255,80,"UNIDAD ARIT./LOG.",17);
TextOut(hdc,493,120,"UNIDAD DE CONTROL",17);
TextOut(hdc,170,380,"ACUMULADOR",10);
TextOut(hdc,170,400,"CIRCUITOS LOGICOS",17);
TextOut(hdc,170,420,"REGISTRO B",10);

Pluma= CreatePen(PS_SOLID,3,RGB(255,255,0));
Pluma1= SelectObject(hdc,Pluma);

MoveTo(hdc,490,225);
LineTo(hdc,430,225);
MoveTo(hdc,435,230);
LineTo(hdc,430,225);
LineTo(hdc,435,220);

MoveTo(hdc,150,285);
LineTo(hdc,230,285);
```

```
MoveTo(hdc,225,280);
LineTo(hdc,230,285);
LineTo(hdc,225,290);

MoveTo(hdc,230,165);
LineTo(hdc,150,165);
MoveTo(hdc,155,160);
LineTo(hdc,150,165);
LineTo(hdc,155,170);

MoveTo(hdc,300,185);
LineTo(hdc,300,200);
MoveTo(hdc,295,195);
LineTo(hdc,300,200);
LineTo(hdc,305,195);

MoveTo(hdc,340,200);
LineTo(hdc,340,185);
MoveTo(hdc,335,190);
LineTo(hdc,340,185);
LineTo(hdc,345,190);

MoveTo(hdc,320,265);
LineTo(hdc,320,250);
MoveTo(hdc,315,255);
LineTo(hdc,320,250);
LineTo(hdc,325,255);

MoveTo(hdc,550,285);
LineTo(hdc,550,340);
LineTo(hdc,90,340);
LineTo(hdc,90,305);
MoveTo(hdc,85,310),
LineTo(hdc,90,305),
LineTo(hdc,95,310);

SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

  VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA FUNCION DE LA ALU ES ACEPTAR DATOS BINARIOS QUE ESTAN ALMACENADOS EN",
        "LA MEMORIA Y EJECUTAR OPERACIONES ARITMETICAS CON ESTOS DATOS DE ACUERDO ",1,12,10,12,30,71,72);
TIEMPO(cuenta6);

  VER_CADENA_CON(hwnd,0,127,0,0,0,0,"CON INSTRUCCIONES QUE PROVIENEN DE LA UNIDAD DE CONTROL",
        "",1,35,10,35,30,55,0);
TIEMPO(cuenta6);

   VER_CADENA_CON(hwnd,255,0,0,0,0,0,"LA UAL CONTIENE CUANDO MENOS DOS REGISTROS DE MULTIVIBRADORES BIESTABLES",
        "EL REGISTRO B Y EL REGISTRO ACUMULADOR",1,15,10,15,30,72,38);

Pluma= CreatePen(PS_SOLID,1,RGB(0,0,0));
Pluma1= SelectObject(hdc,Pluma);

for(r=0,r<=3;r++){
```

```
Brocha= CreateSolidBrush(RGB(0,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,150,400,180);
Ellipse(hdc,150,380,160,390);
Rectangle(hdc,240,270,400,300);
Ellipse(hdc,150,420,160,430);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TIEMPO(cuenta);
Brocha= CreateSolidBrush(RGB(255,255,255));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,150,400,180);
Ellipse(hdc,150,380,160,390);
Rectangle(hdc,240,270,400,300);
Ellipse(hdc,150,420,160,430);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);
TIEMPO(cuenta);
}
SelectObject(hdc,Pluma1);
DeleteObject(Pluma1);

Brocha= CreateSolidBrush(RGB(127,127,127));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,150,400,180);
Ellipse(hdc,150,380,160,390);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

Brocha= CreateSolidBrush(RGB(255,0,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,240,270,400,300);
Ellipse(hdc,150,420,160,430);
SelectObject(hdc,Brocha1);
DeleteObject(Brocha);

 VER_CADENA_CON(hwnd,255,255,0,0,0,0,"TAMBIEN CONTIENE LOGICA COMBINATORIA,
LA CUAL EFECTUA LAS OPERACIONES",
                        "ARITMETICAS SOBRE LOS NUMEROS DEL REGISTRO B Y DEL
ACUMULADOR",1,35,10,35,30,69,61);
 TIEMPO(cuenta6);

 VER_CADENA_CON(hwnd,191,191,191,0,0,0,"UN EJEMPLO DE ESTE FUNCIONAMIENTO ES
EL SIGUIENTE:",
                        "",1,35,10,35,30,50,0);
 TIEMPO(cuenta6);

VER_CADENA_CON(hwnd,255,255,0,0,0,0,"LA UNIDAD DE CONTROL RECIBE UNA
INSTRUCCION DE LA UNIDAD DE MEMORIA",
        "",1,35,10,35,30,67,0);


 SetTextColor(hdc,RGB(0,0,0));
 SetBkMode(hdc,TRANSPARENT);

 TIEMPO(cuenta4);
s=cadena;
g=0;f=530;
```

```
 while(g!=4){
  s=&cadena[g];
  TextOut(hdc,f,225,s,1);
  f=f+12;
  g++;
  MessageBeep(0);
  TIEMPO(cuenta4);
    }

VER_CADENA_CON(hwnd,255,0,0,0,0,0,"ESTA INSTRUCCION ESPECIFICA QUE UN NUMERO
ALMACENADO EN CIERTA LOCALIDAD",
              "DE LA MEMORIA (DIRECCION) SE SUMARA CON EL NUMERO QUE ESTA
ALMACENADO EN",1,35,10,35,30,72,72);
TIEMPO(cuenta6);

VER_CADENA_CON(hwnd,0,255,0,0,0,0,"ESE MOMENTO EN EL REGISTRO ACUMULADOR",
            "",1,35,10,35,30,37,0);

 TIEMPO(cuenta4);
 Brocha= CreateSolidBrush(RGB(191,191,191));
 Brocha1= SelectObject(hdc,Brocha);

 Rectangle(hdc,500,170,600,280);

 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

s=cadena1;
g=0;f=300;
 while(g!=4){
  s=&cadena1[g];
  TextOut(hdc,f,162,s,1);
  f=f+12;
  g++;
  MessageBeep(0);
  TIEMPO(cuenta4);
    }

TIEMPO(cuenta6);

VER_CADENA_CON(hwnd,191,191,191,0,0,0,"EL NUMERO QUE SE SUMARA SE TRANSFIERE
DE LA MEMORIA AL REGISTRO B",
            "",1,35,10,35,30,65,0);

TIEMPO(cuenta6);

 Brocha= CreateSolidBrush(RGB(0,127,0));
 Brocha1= SelectObject(hdc,Brocha);

s=cadena2;
f=70;e=0;
w=4;g=0;
i=0;h=0;
v=0;
n=350;

while(e!=5){
  f=f+(10*e);
```

```
  h=w-e;
 while(g!=h){
  s=&cadena2[g];
  TextOut(hdc,f,280,s,1);
  f=f+12;
  g++;
     }
   s=&cadena2[4-i];
   TextOut(hdc,n,282,s,1);
   MessageBeep(0);
    TIEMPO(cuenta4);
   Rectangle(hdc,40,150,140,300);
   i++;
   e++;
   g=0;
   n-=12;
   f=70;
   }
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

    VER_CADENA_CON(hwnd,0,255,0,0,0,0,"EL NUMERO CONTENIDO EN EL REGISTRO B Y
EL NUMERO EN EL REGISTRO ACUMULADOR",
            "SE SUMAN EN LOS CIRCUITOS LOGICOS",1,13,10,13,30,74,33);

 TIEMPO(cuenta4);

s=cadena1;
g=0;f=300;
 while(g!=4){
  s=&cadena1[g];
  TextOut(hdc,f,212,s,1);
  f=f+12;
  g++;
  MessageBeep(0);
  TIEMPO(cuenta4);
     }
  Brocha= CreateSolidBrush(RGB(127,127,127));
  Brocha1= SelectObject(hdc,Brocha);
  Rectangle(hdc,240,150,400,180);
  SelectObject(hdc,Brocha1);
  DeleteObject(Brocha);
  TIEMPO(cuenta4);

s=cadena2;
g=0;f=300;
 while(g!=4){
  s=&cadena2[g];
  TextOut(hdc,f,225,s,1);
  f=f+12;
  g++;
  MessageBeep(0);
  TIEMPO(cuenta4);
     }
  Brocha= CreateSolidBrush(RGB(255,0,0));
  Brocha1= SelectObject(hdc,Brocha);
  Rectangle(hdc,240,270,400,300);
  SelectObject(hdc,Brocha1);
```

```
 DeleteObject(Brocha);
 TIEMPO(cuenta6);

    VER_CADENA_CON(hwnd,0,255,0,0,0,0,"ESTA SUMA SE EFECTUA POR EL COMANDO
EMITIDO DESDE LA UNIDAD DE CONTROL.",
                   "LA SUMA RESULTANTE SE ENVIA ENTONCES AL ACUMULADOR Y
LUEGO A LA MEMORIA",1,13,10,13,30,71,71);

 TIEMPO(cuenta4);

 Brocha= CreateSolidBrush(RGB(0,255,0));
 Brocha1= SelectObject(hdc,Brocha);
 Rectangle(hdc,280,205,360,245);
 SelectObject(hdc,Brocha1);
 DeleteObject(Brocha);

d=cadena3;
f=290;e=0;
w=4;g=0;
i=0;h=0;
v=0;r=345;
n=120;

while(e<=6){
  i=v;
  h=w-e;
  f=f+(10*e);
 while(g<=h){
  d=&cadena3[g];
  TextOut(hdc,f,218,d,1);
  f=f+12;
  g++;
    }
 while(i<6){
   d=&cadena3[5-i];
   TextOut(hdc,r,162,d,1);
   i=6; }
   i=v;
   r-=10;
 while(i<7){
   d=&cadena3[6-i];
   TextOut(hdc,n,170,d,1);
   i=7; }
   i=v;
   n-=10;
   MessageBeep(0);
   TIEMPO(cuenta4);
   Rectangle(hdc,280,205,360,245);
    e++;
    g=0;
    v++;
    f=295;
   }

Brocha= CreateSolidBrush(RGB(0,255,0));
Brocha1= SelectObject(hdc,Brocha);
Rectangle(hdc,280,205,360,245);
SelectObject(hdc,Brocha1);
```

```
DeleteObject(Brocha);

 TIEMPO(cuenta4);
    VER_CADENA_CON(hwnd,255,255,0,0,0,0,"EL  NUEVO  NUMERO  QUE  ESTA  EN  EL
ACUMULADOR PUEDE PERMANECER AHI DE MANERA",
                                  "QUE SE PUEDA SUMAR OTRO NUMERO A EL O AL
FINAL SE TRANSFIERE A LA MEMORIA",1,13,10,13,30,72,73);
 BOTONES(hwnd,1);
 BOTONES(hwnd,3);
 BOTONES(hwnd,5);
 SetCursor(Cursor_Cruz);
 ULTIMA_IMAGEN(hwnd);

 var_w=2;j=2;

 } //  TERMINA MODULO UNIDAD ARITMETICA / LOGICA


ARCHIVOS *.H   DECLARACION DE FUNCIONES
**********************


void EVALUACION_MODULO_1_2(HWND);
void EVALUACION_MODULO_3_4(HWND);
void EVALUACION_MODULO_3_4_1(HWND);
void EVALUACION_MODULO_5_6(HWND);
void EVALUACION_MODULO_7_8(HWND);
void EVALUACION_MODULO_9_10(HWND);


**************************


 void TIEMPO(int);
 void VER_CADENA_CON(HWND,int,int,int,int,int,int,char *,char *,int,int,int,int,int,int,int,int);
 void VER_CADENA_SIN(HWND,int,int,char *,int,int,int,int);
 void CREA_RECTANGULO_BIDIMENSIONAL(HWND,int,int,int,int,int,int,int,int);
 void BOTONES(HWND,int);
 void CREA_BOTONES(HDC,int,int,int,int,char *,int,int);
 void ULTIMA_IMAGEN(HWND);
 void INT_MODULOS(HDC,int,int,int);
 void MODULOS(HDC);

 void zResetFunc(HWND);
 void BOTON_SELECCION(HDC,int,int,int,int,int);
 void BOTONES_EVALUACION(HWND, int);
 void BOTONES_1(HWND,int);


********************************


 void MICROCOMPUTADORA_1(HWND);
 void MICROCOMPUTADORA_2(HWND);


************************************
```

```
void APLICACION(HWND,int);
void APLICACION_1(HWND);
void SELECCION(HWND);
void CONTROL_GENERAL(HWND,HDC,int);
void CONTROL_MODULOS(HWND,HDC,int,int);
void CHECA_APLI(HWND,int,int);
void CHECA_APLI_1(HWND,int,int);
void CHECA_MOD(HWND,int,int);
void CHECA_SELEC(HWND,int,int);
void AYUDA(HWND);
void LECTOR_AYUDA(HWND);
void EVALUACION(HWND);
void CHECA_EVALUACION(HWND,int,int);

*****************************

void EVALUACION_ENTRADA(HWND);
void EVALUACION_SALIDA(HWND);
void EVALUACION_CPU_1(HWND);
void EVALUACION_CPU_2(HWND);
void EVALUACION_ALU_1(HWND);
void EVALUACION_ALU_2(HWND);
void EVALUACION_MEMORIA_1(HWND);
void EVALUACION_MEMORIA_2(HWND);
void EVALUACION_MEMORIA_3(HWND);
void EVALUACION_MEMORIA_4(HWND);
void EVALUACION_CONTROL(HWND);
void EVALUACION_MICRO_1(HWND);
void EVALUACION_MICRO_2(HWND);
void EVALUACION_MICRO_3(HWND);
void EVALUACION_MICRO_4(HWND);
void ULTIMA_IMAGEN(HDC);

*******************************************

void MODULO_1(HWND,HDC);
void MODULO_2(HWND,HDC);
void MODULO_3(HWND,HDC);
void MODULO_4(HWND,HDC);
void MODULO_5(HWND,HDC);
void MODULO_6(HWND,HDC);
void MODULO_7(HWND,HDC);
void MODULO_8(HWND,HDC);
void MODULO_9(HWND,HDC);
void MODULO_10(HWND,HDC);
void FLECHAS_DATOS(HDC,int,int,int);
void FLECHAS_CONTROL(HDC,int,int,int);

***********************************

void MOD_MEMORIA_1(HWND,HDC);
void MOD_MEMORIA_2(HWND,HDC);
```

```
  void MOD_MEMORIA_3(HWND,HDC);
  void MOD_CONTROL(HWND,HDC);
  void MOD_CPU_1(HWND,HDC);
  void MOD_CPU_2(HWND,HDC);
  void MOD_ALU(HWND,HDC);
  void MOD_ENTRADA(HWND,HDC);
  void MOD_SALIDA(HWND,HDC);


**********************************

ARCHIVO *.RC DE RECURSOS

//-------------------------------------------------------------------------
--
//-------------------------------------------------------------------------
--

#define zRCFILE
#include <WINDOWS.H>
#include "TESIS.H"

//PARA QUE SE CARGE SOLAMENTE CUANDO SE REQUIERA Y PA RA QUE SE DESCARGHE
CUANDO NO SE LE REQUIERA
MyBitmap  BITMAP LOADONCALL DISCARDABLE MODULOS.bmp
MyBitmap1 BITMAP LOADONCALL DISCARDABLE PRESENTA.bmp
MyBitmap2 BITMAP LOADONCALL DISCARDABLE MICRO.bmp
MyBitmap3 BITMAP LOADONCALL DISCARDABLE SELEC.bmp
MyBitmap4 BITMAP LOADONCALL DISCARDABLE ENT_SAL.bmp
Icono_Tesis ICON LOADONCALL DISCARDABLE TESIS ICO
// ELEMENTOS DEL MENU
TESIS MENU
 BEGIN
 POPUP "&File"
  BEGIN
   MENUITEM "Sali&r'       SALIR
  END
 POPUP "&Conceptos_Basicos"
  BEGIN
   MENUITEM  "&Bas&icos", TXT_CONCEPTOS_BASICOS
  END
 POPUP "&Demo"
  BEGIN
  MENUITEM  "&Demo_Gral",  TXT_DEMO_1
  MENUITEM  "&Demo_Mod",  TXT_DEMO_2
  MENUITEM  "&Demo_Micro",  TXT_DEMO_3
  END
 POPUP "\a&Ayuda"
  BEGIN
  MENUITEM  "&Ayuda...", TXT_AYUDA_PRINCIPAL
  END
 END


// CONTENIDO DE LAS CADENAS
STRINGTABLE
 BEGIN
  TEX_TITULO    "TUTORIAL"
```

```
  TEX_AYUDA     "AYUDA DE LA APLICACION"
  TEX_AYUDATEX  "PRESENTA CONCEPTOS GENERALES"
 END

••••••••••••••••••••••••••••••••••

#if !defined(zRCFILE)              /* si no es un archivo .RC ... */
 LONG FAR PASCAL zWndProc(HWND, unsigned, WORD, LONG);
 int PASCAL WinMain(HANDLE,HANDLE,LPSTR,int);
 HWND inicializa_ventana(HANDLE);
 BOOL inicializa_clase(HANDLE);
 void LECTOR_AYUDA(HWND);
 void TIPO_LETRA(HDC);
#endif
#define TXT_SALIR          1

#define TXT_CONCEPTOS_BASICOS 4
#define TXT_DEMO_1         5
#define TXT_DEMO_2         6
#define TXT_DEMO_3         7

#define TXT_AYUDA_PRINCIPAL  8

#define TEX_TITULO     1
#define TEX_ERROR      2
#define TEX_MOUSE      3
#define TEX_AYUDA      4
#define TEX_AYUDATEX   5

••••••••••••••••••••••••••••••••••••••

;Archivo TESIS.DEF de definicion del modulo

NAME        TESIS
DESCRIPTION 'MARCO ANTONIO AVILA MORALES, 1994'
EXETYPE     WINDOWS
STUB        'WINSTUB.EXE'
DATA        MOVEABLE MULTIPLE
CODE        MOVEABLE DISCARDABLE PRELOAD
HEAPSIZE    10240
STACKSIZE   10240
EXPORTS     zWndProc
```

APENDICE.

A Continuación se presenta una lista con todas las funciones del API de Windows (DLL):

GDI funciones (3.1)

AbortDoc Termina el trabajo de impresión.
AddFontResource Adiciona una fuente a la tabla de fuentes.
AnimatePalette Reemplaza la entrada de la paleta lógica.
Arc Dibuja un arc.
BitBlt Copia un bitmap entre contextos de dispositivos.
Chord Dibuja una cuerda.
CloseMetaFile Cierra un meta-archivo DC y obtiene el controlador.
CombineRgn Crea la región por combinar dos regiones.
CopyMetaFile Copia un meta-archivo.
CreateBitmap Crea un dispositivo dependiente del bitmap de memoria.

CreateBitmapIndirect Crea un bitmap usando la estructura BITMAP.
CreateBrushIndirect Crea una brocha con los atributos especificados.
CreateCompatibleBitmap Crea un bitmap compatible con el DC.
CreateCompatibleDC Crea un DC compatible con el DC especificado.
CreateDC Crea un contexto de dispositivo.
CreateDIBitmap Crea un controlador de bitmap con el DIB especificado.
CreateDIBPatternBrush Crea una brocha padre desde DIB.
CreateDiscardableBitmap Crea un bitmap descargable.
CreateEllipticRgn Crea una región elíptica.

CreateEllipticRgnIndirect Crea una región elíptica.
CreateFont Crea una fuente lógica.
CreateFontIndirect Crea una fuente usando la estructura LOGFONT.
CreateHatchBrush Crea una brocha con trama.
CreateIC Crea un contexto de información.
CreateMetaFile Crea un meta-archivo del contexto de dispositivo.
CreatePalette Crea un color de paleta lógica.
CreatePatternBrush Crea una brocha padre de un bitmap.
CreatePen Crea una pluma.
CreatePenIndirect Crea una pluma usando una estructura LOGPEN.

CreatePolygonRgn Crea una región poligonal.
CreatePolyPolygonRgn Crea una región consistente en polígonos.
CreateRectRgn Crea una región rectangular.
CreateRectRgnIndirect Crea una región usando una estructura RECT.
CreateRoundRectRgn Crea una región rectangular con esquinas redondas.
CreateScalableFontResource Crea un archivo recurso con fuente info.
CreateSolidBrush Crea una brocha sólida con el color especificado.
DeleteDC Borra un contexto de dispositivo.
DeleteMetaFile Invalida un controlador meta-archivo.
DeleteObject Borra un objeto de la memoria.

DeviceCapabilities Recupera la capacidad de un dispositivo.
DeviceMode Despliega una caja de diálogo para el modo de impresión.
DPtoLP Convierte los puntos de un dispositivo a puntos lógicos.
Ellipse Dibuja una elipse.
EndDoc Finaliza la tarea de impresión.
EndPage Finaliza la página.
EnumFontFamilies Recupera la fuente en una familia especificada.

EnumFonts Enumera las fuentes en el dispositivo especificado.
EnumMetaFile Enumera los registros de meta-archivos.
EnumObjects Enumera las plumas y brochas en un contexto de dispositivo.

EqualRgn Compara dos regiones para igualarlas.
Escape Permite acceso al dispositivo capacitado.
ExcludeClipRect Cambia la región de recorte , excluyendo el rectangulo.
ExtDeviceMode Despliega una caja de diálogo para el modo de impresión.
ExtFloodFill Rellena un área con la brocha normal.
ExtTextOut Escribe una cadena de carácteres en una región rectangular.
FillRgn Rellena una región con la brocha especificada.
FloodFill Rellena un área con la brocha normal.
FrameRgn Dibuja el borde alrededor de una región.

GetAspectRatioFilter Recupera el escenario aspecto-razón del filtro.
GetAspectRatioFilterEx Recupera el escenarioaspecto-razón del filtro.
GetBitmapBits Copia los bits de un bitmap a un buffer.
GetBitmapDimension Recupera la anchura y la altura de un bitmap.
GetBitmapDimensionEx Recupera la anchura y la altura de un bitmap.
GetBkColor Recupera el color normal de fondo.
GetBkMode Recupera el modo de fondo.
GetBoundsRect Retorna la corriente acumulada del rectangulo destino.
GetBrushOrg Recupera el origen de la brocha normal.

GetBrushOrgEx Recupera el origen de la brocha normal.
GetCharABCWidths Recupera la anchura de los carácteres TrueType.
GetCharWidth Recupera la anchura de los carácteres.
GetClipBox Recupera un rectangulo para la región de recorte.
GetCurrentPosition Recupera la posición normal en unidades lógicas.
GetCurrentPositionEx Recupera la posición normal en unidades lógicas.
GetDCOrg Recupera la traducción origen para el contexto de dispositivo.
GetDeviceCaps Recupera la capacidad del dispositivo.
GetDIBits Copia los bits DIB hacia un buffer.

GetFontData Recupera los datos de la fuente métrica.
GetGlyphOutline Recupera datos paracl contorno individual de carácteres.
GetKerningPairs Recupera el par kerning para la fuente normal.
GetMapMode Recupera el modo de mapeo.
GetMetaFile Crea un controlador para un meta-archivo especifico.
GetMetaFileBits Crea un objeto global en memoria para un meta-archivo.
GetNearestColor Recupera el color sólido aprobechable.
GetNearestPaletteIndex Recupera el próximo juego para un color.
GetObject Recupera información acerca del objeto.

GetOutlineTextMetrics Recuperamétrica para fuentes TrueType.
GetPaletteEntries Recupera un rango de una entrada de paleta.
GetPixel Recupera el valor del color RGB del pixel especificado.
GetPolyFillMode Recupera el modo normal del poligono-relleno.
GetRasterizerCaps Recupera el rango de la fuente TrueType en sistema.
GetRgnBox Recupera el rectangulo destino para una región.
GetROP2 Recupera el modo normal de dibujo.
GetStockObject Recupera el controlador de las plumas, brochas ó fuentes existentes.
GetStretchBltMode Recupera el modo normal bitmap-estrechar.

GetSystemPaletteEntries Recupera la entrada para la paleta del sistema.
GetSystemPaletteUse Determina el uso de la paleta entera del sistema.
GetTextCharacterExtra Recupera el espaciado inter-carácter.
GetTextAlign Recupera la bandera de alineación-texto.

GetTextColor Recupera el color normal del texto.
GetTextExtent Determina las dimensiones de la cadena de texto especificada.
GetTextExtentPoint Recupera las dimensiones de la cadena de texto especificada.
GetTextFace Recupera el nombre del tipo de la fuente normal.
GetTextMetrics Recupera la métrica para la fuente normal.

GetViewportExt Recupera el alcance del puerto-visual.
GetViewportExtEx Recupera el alcance del puerto-visual..
GetViewportOrg Recupera el origen del puerto-visual.
GetViewportOrgEx Recupera el origen del puerto-visual.
GetWindowExt Recupera la extensión de la ventana.
GetWindowExtEx Recupera la extensión de la ventana.
GetWindowOrg Recupera el origen de la ventana.
GetWindowOrgEx Recupera el origen de la ventana.
IntersectClipRect Crea una región de recorte en una intersección.

InvertRgn Invierte el color en una región.
IsGDIObject Determina si el controlador no es un objeto GDI.
LineDDA Computa los puntos sucesivos en una línea.
LineTo Dibuja una línea desde una posición normal.
LPtoDP Convierte puntos lógicos a puntos de dispositivo.
MoveTo Mueve la posición normal.
MoveToEx Mueve la posición normal.
OffsetClipRgn Mueve la región de recorte.
OffsetRgn Mueve una región por una desviación específica.
OffsetViewportOrg Mueve el origen del visual-puerto.

OffsetViewportOrgEx Mueve el origen del puerto-visual.
OffsetWindowOrg Mueve el origen del puerto-visual.
OffsetWindowOrgEx Mueve el origen del puerto-visual.
PaintRgn Rellena una región con la brocha dada del contexto de dispositivo.
PatBlt Crea un módelo de bitmap.
Pie Dibuja una cuña con forma-tarta.
PlayMetaFile Representa un meta-archivo.
PlayMetaFileRecord Representa un registro de meta-archivo.
Polygon Dibuja un polígono.
Polyline Dibuja segmentos de línea conectadas a los puntos especificados.
PolyPolygon Dibuja una serie de polígonos.

PtInRegion Determina si un punto esta en una región.
PtVisible Determina si un punto esta en una región de recorte.
QueryAbort Determina si término el trabajo de impresión.
Rectangle Dibuja un rectangulo.
RectInRegion Determina si un rectangulo esta en una región sobrepuesta.
RectVisible Determina si un rectangulo esta en una región de recorte.
RemoveFontResource Quita un recurso fuente adicionado.
ResetDC Actualiza el contexto de dispositivo.
RestoreDC Restablece el contexto de dispositivo.

RoundRect Dibuja un rectangulo con las esquinas redondeadas.
SaveDC Salva el estado normal de un contexto de dispositivo.
ScaleViewportExt Escala el alcance del puerto-visual.
ScaleViewportExtEx Escala el alcance del puerto-visual.
ScaleWindowExt Escala el alcance de la ventana.
ScaleWindowExtEx Escala el alcance de la ventana.
SelectClipRgn Selecciona la región de recorte de un contexto de dispositivo.
SelectObject Selecciona un objeto en un contexto de dispositivo.
SetAbortProc Pone una función abortar para un trabajo de impresión.

SetBitmapBits Pone los bits de un bitmap de un arreglo de bytes.

SetBitmapDimension Pone la altura y anchura de un bitmap.
SetBitmapDimensionEx Pone la anchura y altura de un bitmap.
SetBkColor Pone el color normal de fondo.
SetBkMode Pone el color normal de fondo.
SetBoundsRect Controla la acumulación de rectangulos-destinos.
SetBrushOrg Pone el origen de la brocha normal.
SetDIBits Pone los bits a un bitmap.
SetDIBitsToDevice Pone los bits DIB a un dispositivo.
SetMapMode Pone el modo de mapeo
SetMapperFlags Pone la bandera mapeo-fuente.

SetMetaFileBits Crea un objeto en memoria de un meta-archivo.
SetMetaFileBitsBetter Crea en memoria un objeto de un meta-archivo.
SetPaletteEntries Pone el color y la bandera para un color de paleta.
SetPixel Pone el pixel del color especificado.
SetPolyFillMode Pone el modo poligono-relleno.
SetRectRgn Cambia una región hacia el rectangulo especificado.
SetROP2 Pone el modo de dibujo normal.
SetStretchBltMode Pone el modo estrechar-bitmap.
SetSystemPaletteUse Pone el uso del sistema de paleta de colores estaticos.

SetTextAlign Pone banderas de alineación de texto.
SetTextCharacterExtra Pone el espaciado entre carácteres.
SetTextColor Pone el color del texto del primer plano.
SetTextJustification Pone la alineación para el texto de salida.
SetViewportExt Pone el alcance del puerto-visual.
SetViewportExtEx Pone el alcance del puerto-visual
SetViewportOrg Pone el origen del puerto-visual.
SetViewportOrgEx Pone el origen del puerto-visual.
SetWindowExt Pone el alcance de la ventana.
SetWindowExtEx Pone el alcance de la ventana.
SetWindowOrg Pone el origen de la ventana.
WindowOrgEx Pone el origen de la ventana
SpoolFile Coloca un archivo en la cola del spooler (programa de cola de impresión).
StartDoc Comienza a imprimir un trabajo.
StartPage Prepara un driver de impresión para recibir datos.
StretchBlt Copia un bitmap, transformandolo si es necesario.
StretchDIBits Mueve un DIB desde su fuente hacia su rectangulo destino.
TextOut Escribe cadenas de carácteres en la posición especificada.
UnrealizeObject Reestablece el origen de las brochas y lleva a cabo paletas.
UpdateColors Actualiza el color del área cliente.

Kernel funciones (3.1)

_hread
_hwrite
_lclose
_lcreat
_llseek
_lopen
_lread
_lwrite

AccessResource
AddAtom

AllocDStoCSAlias

AllocResource Coloca la memoria para una fuente.
AllocSelector Coloca un nuevo selector.
AnsiToOem Traduce de una fuente de Windows a una fuente de OEM.
AnsiToOemBuff Traduce de una fuente de Windows a una fuente de OEM.
Catch Captura la ejecucuón normal del formato.
CloseSound No se utiliza en Windows.
CountVoiceNotes No se utiliza en Windows.
DebugBreak Provoca una ruptura en el punto de excepción .
DebugOutput Manda o envia mensages a la terminal del depurador.
DeleteAtom Decrementa cuentas de referencia muy pequeñas.

DirectedYield Obliga la ejecución de una tarea especifica para continuar.
DOS3Call Emite un DOS Int 21h cuando la función lo requiera.
FatalAppExit Finaliza una aplicación.
FatalExit Displays Depura la información después del punto de ruptura de excepción .
FindAtom Retoma una cadena pequeña de una tabla local pequeña.
FindResource Localiza una fuente en un archivo.
FreeLibrary Descarga una libreria de una instancia en módulo.
FreeModule Descarga una instancia en modulo.
FreeProcInstance Libera una función de instancia.
FreeResource Descarga una fuente de instancia.

FreeSelector Libera un selector colocado.
GetAtomHandle Recupera un pequeño manejador.
GetAtomName Recupera una cadena local pequeña.
GetCodeHandle Determina la locación de una función.
GetCodeInfo Recupera la información de un segmento de código.
GetCurrentPDB Retorna la dirección del selector de un PDB normal.
GetCurrentTask Retorna al manejador sus tareas normales.
GetDOSEnvironment Retorna un señalador lejano al medio ambiente normal.
GetDriveType Determina el tipo de drive.
GetFreeSpace Retorna el número de bytes libres de un grupo

GetInstanceData Copia los datos de una instancia previa a una corriente.
GetKBCodePage Retorna el codigo de pagina normal.
GetKeyboardType Recupera la información de la ventana.
GetKeyNameText Recupera la cadena representativa del nombre para acceso.
GetModuleFileName Retorna el nombre de archivo de un manejador de modulo.
GetModuleHandle Retorna un manejador de módulo para un módulo nombrado.
GetModuleUsage Retorna la cuenta de referencia para un módulo.
GetNumTasks Retorna el número normal para las tareas.
GetPrivateProfileInt Recupera el valor integro desde la iniciación del archivo.
GetPrivateProfileString Recupera una cadena desde una iniciación de archivo.

GetProcAddress Recupera la dirección de una función DLL exportada.
GetProfileInt Recupera un valor integro de WIN.INI .
GetProfileString Recupera una cadena de WIN.INI .
GetSelectorBase Recupera la selección base de un selector.
GetSelectorLimit Recupera el límite de un selector.
GetSystemDirectory Retorna el directorio del sistema Windows.
GetTempDrive Retorna una carta de disco de drive para un archivo temporal.
GetTempFileName Crea un nombre de archivo temporal.
GetThresholdEvent No utilizable en Windows 3.1.
GetThresholdStatus No utilizable en Windows 3.1.

GetVersion Retorna las versiones DOS y Windows normales..

GetWinDebugInfo Recupera el sistema de depuración de información normal.
GetWindowsDirectory Retorna el directorio de Windows.
GetWinFlags Retorna el sistema normal de configuración de banderas.
GlobalAlloc .Coloca la memoria de un grupo.
GlobalCompact Genera memoria global libre por medio de compactación.
GlobalDosAlloc Coloca la memoria disponible para DOS en un modo real.
GlobalDosFree Libera la memoria global colocada por medio de GlobalDosAlloc .
GlobalFix Asegura el objeto de una memoria global en una memoria lineal.
GlobalFlags Retorna la información acerca del objeto de una memoria global.

GlobalFree Libera el objeto de una memoria global.
GlobalHandle Recupera un manejador para un selector especifico.
GlobalLock Asegura el objeto de una memoria global y retoma el señalador.
GlobalLRUNewest Mueve el objeto de la meoria global a la mas nueva posición de LRU.
GlobalLRUOldest Mueve el objeto de la meoria global a la mas vieja posición de LRU.
GlobalNotify Instala un procedimiento de notificación.
GlobalPageLock Decrementa la memoria global en la cuenta del seguro de pagina.
GlobalPageUnlock Decrementa la memoria global en la cuenta del seguro de pagina.
GlobalReAlloc Cambia el tamaño de los atributos del objeto de la memoria global.

GlobalSize Retorna el tamaño del objeto de la memoria global.
GlobalUnfix Abre el objeto de una memoria global en una memoria lineal.
GlobalUnlock Abre el objeto de la memoria global.
GlobalUnWire No se utiliza en Windows 3.1.
GlobalWire No se utiliza en Windows 3.1.
hmemcpy Copia los bytes de la fuente al buffer destino.
InitAtomTable Tamaño del grupo de la pequeña tabla local.
IsBadCodePtr Determina si el indicador de código es válido.
IsBadHugeReadPtr Determina si el indicador de lectura grande es valido
IsBadHugeWritePtr Determina si el indicador de escritura grande es valido.

IsBadReadPtr Determina si el indicador de lectura es valido.
IsBadStringPtr Determina si el indicador de escritura es valido.
IsBadWritePtr Determina si el indicador de escritura es valido.
IsDBCSLeadByte Determina si el indicador del byte principal es DBCS.
IsTask Determina si el trabajo del controlador es válido.
LimitEmsPages No se utiliza enWindows 3.1.
LoadLibrary Carga el modulo de libreria especificado.
LoadModule Carga y ejecuta al programa.
LoadResource Carga los recursos especificados en la memoria global.
LocalAlloc Coloca memoria del montón local.

LocalCompact Genera memoria libre local compactada.
LocalFlags Retorna el objeto de información a la memoria local.
LocalFree Libera el objeto de la memoria local.
LocalHandle Retorna el controlador de objeto a la memoria local.
LocalInit Inicializa el montón local.
LocalLock Asegura al objeto memoria local y retorna el indicador.
LocalReAlloc Varia el tamaño o los atributos del objeto de la memoria local.
LocalShrink Encoje el montón local especificado.
LocalSize Retorna el tamaño del objeto de la memoria local.
LocalUnlock Desasegura el objeto de la memoria local.

LockResource Retorna la dirección del recurso
LockSegment Asegura un segmento de memoria descargable.
LogError Identifica un mensage de error.
LogParamError Identifica un parámetro de identificación de error.
lstrcat Agregar un string a otro.

lstrcpy Copiar un string a un buffer.
lstrcpyn Copiar caracteres de un string a un buffer.
lstrlen Retorna el número de caracteres de un string.
MakeProcInstance Retorna la dirección para la función de código prolog.
MapVirtualKey Busca código y lo Traduce a código virtual-key.

MulDiv Dos evaluaciones múltiples y divide el resultado
NetBIOSCall Resultado de una llamada 5Ch a NETBIOS.
OemKeyScan Busca mapas de código OEM ASCII.
OemToAnsi Tranduce cadenas OEM a cadenas Windows.
OemToAnsiBuff Tranduce cadenas OEM a cadenas Windows.
OpenFile Crean, abren , vuelven a abrir ó borran un archivo.
OpenSound No se utiliza en Windows 3.1.
OutputDebugString Manda una cadena de caractéres al depurador.
PrestoChangoSelector Convierte código ó selecciona datos.

SetErrorMode Controla la manipulación de la interrupción de error 24h.
SetHandleCount Varia el número de controladores de archivos disponibles.
SetResourceHandler Instala un recurso cargable de una función llamada atrás.
SetSelectorBase Juega la base de un selector existente.
SetSelectorLimit Juego del límite de un selector.
SetSoundNoise No se uitiliza en Windows 3.1.
SetSwapAreaSize Juega la cantidad de memoria usada para el segmento de código.
SetVoiceAccent No se utiliza en Windows 3.1.
SetVoiceEnvelope No se utiliza en Windows 3.1.
SetVoiceNote No se utiliza en Windows 3.1.

SetVoiceQueueSize No se utiliza en Windows 3.1.
SetVoiceSound No se utiliza en Windows 3.1.
SetVoiceThreshold No se utiliza en Windows 3.1.
SetWinDebugInfo Juega la corriente de información del sistema/debugger.
SizeofResource Retorna el tamaño de un recurso.
StartSound No se utiliza en Windows 3.1.
StopSound No se utiliza en Windows 3.1.
SwitchStackBack Restaura la corriente de trabajo de la pila.
SwitchStackTo Cambia la localización de la pila.
SyncAllVoices No se utiliza en Windows 3.1.
Throw Restaura la ejecución del entorno.
ToAscii Traduce el código virtual-key a caractéres de Windows.

UnlockSegment Desasegura un segmento de memoria descargable.
ValidateCodeSegments Examina la memoria sobreescrita.
ValidateFreeSpaces Detiene la memoria libre para la validación del contenido.
VkKeyScan Traduce los caractéres de Windows a código virtual-key.
WaitSoundState No se utiliza en Windows 3.1.
WinExec Corre un programa.
WritePrivateProfileString Escribe una cadena en un archivo inicializado.
WriteProfileString Escribe una cadena en WIN.INI
wsprintf Formats a string
Yield Stops the current task

Funciones de usuario.

AdjustWindowRect Calcula el tamaño necesario de un rectangulo de Windows.
AdjustWindowRectEx Calcula el tamaño necesario de un rectangulo de Windows.
AnsiLower Reduce una cadena.
AnsiLowerBuff Reduce un buffer de cadena.

AnsiNext Mueve una cadena hacia el siguiente caractér.
AnsiPrev Mueve una cadena hacia el caractér anterior.
AnsiUpper Convierte un caractér a un uppercase.
AnsiUpperBuff Convierte un buffer de cadenas a un uppercase.
AnyPopup Indica si el pop-up ó el overlapped existe en ventana.
AppendMenu Agrega un nuevo artículo al menu.

ArrangeIconicWindows Ordena minimizar las ventanas hijas.
BeginDeferWindowPos Crea una estructura de una ventana en una posición.
BeginPaint Prepara a una ventana para pintarla.
BringWindowToTop Descubre una ventana overlapped.
BuildCommDCBTraducir de cadena dispositivo-definición hacia DCB.
CallMsgFilter Pasa un mensaje hacia una función que filtra mensajes.
CallNextHookEx Pasa información baja a caddenas bajas.
CallWindowProc Pasa un mensaje hacia un procedimiento Windows.
ChangeClipboardChain Quita una cadena de la ventana clipboard-viewer.

ChangeMenu No se utiliza en Windows 3.1.
CheckDlgButton Cambia una marca de detención por una caja de dialógo con botón.
CheckMenuItem Cambia una marca de detención por un artículo del menu.
CheckRadioButton Pone en el sitio de una marca de detención un botón redondo.
ChildWindowFromPoint Determina si la ventana hija contiene un punto.
ClearCommBreak Restaura la transmisión de caractéres.
ClientToScreen Convierte un punto cliente a coordenadas de pantalla.
ClipCursor Limita al cursor a un rectangulo especificado.
CloseClipboard Cierra el clipboard.

CloseComm Cierra un dispositivo de comunicación.
CloseWindow Minimiza una ventana.
CloseDriver Cierra un driver instalable.
CopyCursor Copia un cursor.
CopyIcon Copia un icono.
CopyRect Copia las dimensiones de un rectangulo.
CountClipboardFormats Retorna el número de formatos del clipboard.
CreateCaret Crea una nueva forma para el systema caret (sigunos de intercalación).
CreateCursor Crea un cursor con las dimensiones especificadas.
CreateDialog Crea modelos de cajas de dialógo.

CreateDialogIndirect Crea modelos de cajas de dialógo desde patrónes de memoria.
CreateDialogIndirectParam Crea modelos de cajas de dialógo desde patrónes de memoria.
CreateDialogParam Crea modelos de cajas de dialógo.
CreateIcon Crea un icono con las dimensiónes especificadas.
CreateMenu Crea un menu.
CreatePopupMenu Crea un menu pop-up.
CreateWindow Crea una ventana.
CreateWindowEx Crea unas ventanas.
DefDlgProc Hace que la ventana elabore por default un mensaje.
DefDriverProc Llama por default a un procedimiento instalar - driver.
DeferWindowPos Actualiza a multiples estructuras de ventanas- posiciones.

DefFrameProc Hace default MDI enmarcando la preparación del mensaje de ventana.
DefHookProc Llama la siguiente función en una cadena gancho-función.
DefMDIChildProc Hace default MDI al preparar el mensaje de la ventana hija.
DefWindowProc Llama por default al procedimiento de la ventana.
DeleteMenu Borra un artículo del menu.
DestroyCaret Destruye la corriente del caret.
DestroyCursor Destruye un cursor.
DestroyIcon Destruye un icono.

DestroyMenu Destruye un menu.
DestroyWindow Destruye una ventana.
DialogBox Crea un modal de caja de dialógo.

DialogBoxIndirect Crea un modal de caja de dialógo desde un módelo de memoria.
DialogBoxIndirectPara Crea un modal de caja de dialógo desde un módelo de memoria.
DialogBoxParam Crea un modal de caja de dialógo.
DispatchMessage Envia un mensaje hacia una ventana.
DlgDirList Carga una caja de listas de directorios.
DlgDirListComboBox Carga una caja de listas de directorios.
DlgDirSelect Recupera una selección de una caja de listas de directorios.
DlgDirSelectEx Recupera una selección de una caja de listas de directorios.
DlgDirSelectComboBox Recupera una selección de una caja de listas de directorios.
DlgDirSelectComboBoxEx Recupera una selección de una caja de listas de directorios.
DrawFocusRect Dibuja un rectangulo con el estilo enfocado.
DrawMenuBar Redibuja una barra de menu.
DrawText Dibuja la forma del texto en un rectangulo.
EmptyClipboard Vacia el clipboard y deja libre el controlador de datos.
EnableCommNotification Permite ó no permite el destino de WM_COMMNOTIFY.
EnableHardwareInput Contro la cola de entrada del ratón y del teclado.
EnableMenuItem Permite ó no permite un articulo de menu.
EnableScrollBar Permite ó no permite la flecha de scroll-bar.

EnableWindow Juega el estado permite-ventana
EndDeferWindowPos Actualiza la posición y el tamaño de las ventanas de Windows.
EndDialog Textura y modal de una caja de dialógo.
EndPaint Marca el final de la pintura en la ventana especificada.
EnumChildWindows Pasa el controlador de la ventana hija una llamada atrás.
EnumClipboardFormats Retorna el formato aprobechable clipboard
EnumProps Pasa la lista de propiedad de entrada una llamada atrás.
EnumTaskWindows Pasa el controlador de la ventana de trabajo una llamada atrás.
EnumWindows Pasa el controlador de la ventana padre una llamada atrás.

EqualRect Determina si dos rectangulos son iguales.
EscapeCommFunction Pasa una función extendida hacia un dispositivo.
ExcludeUpdateRgn Excluye la región actualizada de la región de recorte.
ExitWindows Restaura o término Windows.
ExitWindowsExec Termina Windows y corre una aplicación MS-DOS.
FillRect Rellena un rectangulo con la brocha especificada.
FindWindow Retorna un controlador de ventana para una clase y un nombre de ventana.
FlashWindow Destella una vez una ventana.
FlushComm Varia el color al recibir ó transmitir una cola.
FrameRect Dibuja el borde de una ventana con la brocha especificada.
GetActiveWindow Recupera el controlador de la ventana activa.
GetAsyncKeyState Determina la llave de estado.
GetCapture Retorna el controlador para la captura del mouse en la ventana.
GetCaretBlinkTime Retorna el ritmo de parpadeo al caret.
GetCaretPos Retorna la posición normal caret.
GetClassInfo Retorna información de la clase ventana.
GetClassLong Retorna la larga evaluación de la clase ventana
GetClassName Retorna el nombre de la clase ventana.

GetClassWord Retorna a la clase de ventana la evaluación de palabra.
GetClientRect Retorna las coordenadas del area cliente de la ventana.
GetClipboardData Retorna un controlador hacia la corriente de datos del clipboard.
GetClipboardFormatName Retorna el registro del nombre del formato del clipboard.
GetClipboardOwner Retrna al clipboard el controlador ventana-possedor.
GetClipboardViewer Retorna el primer controlador de ventana clipboard-espectador.

GetClipCursor Retorna las coordenadas del rectangulo cursor-limites.
GetCommError Retorna el rango comunicación-dispositivo.
GetCommEventMask Recupera el dispositivo del evento disfrazado .
GetCommState Lee el rango de dispositivo-comunicación.

GetCurrentTime Retorna el lapso de tiempo desde que inicia Windows.
GetCursor Retorna el controlador normal del cursor.
GetCursorPos Retorna la posición normal del cursor.
GetDC Retorna el controlador de ventana del contexto-dispositivo.
GetDCEx Recupera el controlador del contexto-dispositivo.
GetDesktopWindow Retorna al escritorio el controlador de ventana.
GetDialogBaseUnits Retorna la unidad base de la caja de diálogo.
GetDlgCtrlID Retorna el control a la ventana hija.
GetDlgItem Retorna el control del controlador a la caja de diálogo.
GetDlgItemInt Traduce el texto de la caja de diálogo a un entero.

GetDlgItemText Recupera la caja de diálogo el control del texto.
GetDoubleClickTime Retorna el ratón el tiempo del doble-clic.
GetDriverModuleHandle Recupera el controlador de instancia la instalación-driver.
GetDriverInfo Recupera el dato instalable-driver.
GetFocus Retorna el enfoque normal del controlador de ventana.
GetFreeSystemResources Retorna el porcentaje de espacio libre de sistema-recurso.
GetInputState Retorna el rango del ratón y del teclado.
GetKeyboardState Retorna el rango de la llave virtual-teclado.
GetKeyState Retorna el estado especificado de la llave-virtual.
GetLastActivePopup Determina la más reciente ventana activa pop-up.

GetMenu Retorna el controlador de menu para la ventana especificada.
GetMenuCheckMarkDimensions Retorna las dimensiones por default de un bitmap.
GetMenuItemCount Retorna el número de artículos de un menu.
GetMenuItemID Retorna el identificador del artículo del menu.
GetMenuState Retorna el rango de la bandera para el artículo del menu especificado.
GetMenuString Copia la etiqueta de un artículo del menu hacia un buffer.
GetMessage Recupera un mensaje de la cola de mensajes.
GetMessageExtraInfo Recupera información acerca de mensajes de hardware.
GetMessagePos Retorna la posicion del cursor para el último mensaje.
GetMessageTime Retorna el tiempo para el último mensaje.

GetNextDlgGroupItem Retorna el controlador del próximo ó anterior grupo de control.
GetNextDlgTabItem Retorna el próximo ó anterior control WS_TABSTOP.
GetNextDriver Enumera las instancias instalables-driver.
GetNextWindow Retorna la próxima ó anterior ventana manejadora de ventana.
GetOpenClipboardWindow Retorna el controlador hacia ese abierto-clipboard .
GetParent Retorna el controlador de ventana padre.
GetPriorityClipboardFormat Retorna el primer formato clipboard
GetProp Retorna el dato controlador de la lista de propiedad de ventana.
GetQueueStatus Determina el tipo de cola de mensajes.

GetScrollPos Retorna la posición normal al hojear con scroll-bar.
GetScrollRange Retorna la posicón máxima y mínima del scroll-bar.
GetSubMenu Retorna el controlador de menu pop-up.
GetSysColor Retorna el color del elemento desplegado.
GetSysModalWindow Retorna el modo de sistema del controlador de ventana.
GetSystemDebugState Retorna la información del estado del sistema hacia el depurador.
GetSystemMenu Provee acceso hacia el menu del sistema.
GetSystemMetrics Recupera la métrica del sistema.
GetTabbedTextExtent Determina las dimensiones de la tabbed de cadenas.
GetTickCount Recupera la cantidad de tiempo Windows has been running.

GetTimerResolution Recupera la resolución del tiempo.
GetTopWindow Retorna el controlador para la hija dada la ventana.
GetUpdateRect Retorna las dimensiones de la región actualizada de la ventana.
GetUpdateRgn Retorna la región actualizada de la ventana.
GetWindow Retorna el controlador de ventana especificado.
GetWindowDC Retorna el contexto del dispositivo de ventana.
GetWindowLong Retorna la longuitud del valor de la memoria extra de la ventana.
GetWindowPlacement Retorna el estado de la función ventana y su posición máxima y mínima.

GetWindowRect Recupera las coordenadas de ventana en pantalla.
GetWindowTask Retorna el trabajo asociado con la ventana.
GetWindowText Copia el texto de la barra de título de la ventana a un buffer.
GetWindowTextLength Retorna la longuitud del texto de la barra de título de la ventana.
GetWindowWord Retorna el valor de palabra de la memoria extra de la ventana.
GlobalAddAtom Añade una cadena a la pequeña tabla del sistema.
GlobalDeleteAtom Decrementa la cuenta de referencia global.
GlobalFindAtom Recupera la pequeña cadena de la pequeña tabla global.
GlobalGetAtomName Recupera la pequeña cadena global.
GrayString Dibuja el texto gray en la localización especificada.
hardware_event Situa el mensaje del harware en la cola del sistema
HideCaret Quita el caret de la pantalla
HiliteMenuItem Cambia el momento de un top-level del artículo del menu.
InflateRect Cambia las dimensiones del rectangulo.
InSendMessage Determina si la ventana prepara a SendMessage.
InsertMenu Inserta un nuevo artículo en el menu.
IntersectRect Calcula la intersección del rectangulo.
InvalidateRect Adiciona un rectangulo a la región de actualización.

InvalidateRgn Adiciona la región a la región actualizada.
InvertRect Invierte la región rectangular.
IsCharAlpha Determina si el carácter es alfabetico.
IsCharAlphaNumeric Determina si el carácter es alfanumerico.
IsCharLower Determina si el carácter es de acción-infertor.
IsCharUpper Determina si el carácter es de acción-superior.
IsChild Determina si la ventana es ventana hija.
IsClipboardFormatAvailable Determina el aprobechamiento de los datos dado el formato.
IsDialogMessage Determina si el mensaje es para una caja de diálogo.
IsDlgButtonChecked Determina el estado del botón de control.
IsIconic Determina si la ventana es minimizada.
IsMenu Determina si el controlador del menu es válido.
IsRectEmpty Determina si el rectangulo ésta vacio.
IsWindow Determina si el controlador de ventana es válido.
IsWindowEnabled Determina si la ventana acepta entrada de usuarios.
IsWindowVisible Determina el estado de visibilidad de la ventana.
IsZoomed Determina si la ventana es maximizada.
KillTimer Quita el tiempo.

LoadAccelerators Carga un acelerador de la tabla.
LoadBitmap Carga un recurso bitmap.
LoadCursor Carga un recurso cursor.
LoadIcon Carga un recurso icono.
LoadMenu Carga un recurso menu.
LoadMenuIndirect Obtiene un controlador de menu para un módelo de menu.
LoadString Carga un recurso cadena.
LockInput Asegura entradas a todo el trabajo excepto una corriente.
LockWindowUpdate Capacita ó incapacita el dibujar en una ventana.
lstremp Compara dos cadenas de carácteres.
lstrempi Compara dos cadenas de carácteres.

MapDialogRect Mapa de la caja de diálogo en unidades de pixel.
MessageBeep Genera un sonido beep
MessageBox Crea una caja de mensaje en la ventana.
MapWindowPoints Convierte puntos a otras coordenadas de sistema.
ModifyMenu Cambia un artículo de menu existente.
MoveWindow Cambia la posición y dimensiones de una ventana.
OffsetRect Mueve un rectangulo por una compensación.
OpenClipboard Abre el clipboard.

OpenComm Abre el dispositivo de comunicación.
OpenDriver Abre un driver instalable.
OpenIcon Activa o minimiza la ventana.

PeekMessage Detiene un mensaje de la cola.
PostAppMessage Pone un mensaje en una aplicación.
PostMessage Situa un mensaje en la cola de mensajes de una ventana.
PostQuitMessage Dice a Windows cuando la aplicación termina.
PtInRectDetermina si el punto es en un rectangulo.
QuerySendMessage Determina si el mensaje produce un trabajo.
ReadComm Lee desde un dispositivo de comunicación.
RealizePalette Mapa de la lógica de entrada para la paleta del sistema.
RedrawWindow Actualiza una región ó rectangulo cliente.
RegisterClass Registra la clase ventana.

RegisterClipboardFormat Registra un nuevo formato clipboard.
RegisterWindowMessage Define un nuevo y único mensaje de ventana.
ReleaseCapture Libera la captura del ratón.
ReleaseDC Libera un contexto de dispositivo.
RemoveMenu Borra un artículo del menu y un menu pop-up.
RemoveProp Quita la lista de propiedad de entrada.
ReplyMessage Responde un SendMessage.
ScreenToClient Convierte un punto de pantalla a coordenadas cliente.
ScrollDC Traza un rectangulo horizontal y vertical.
ScrollWindow Traza el área cliente de una ventana.
ScrollWindowEx Traza el área cliente de una ventana.

SelectPalette Selecciona una paleta en un contexto de dispositivo.
SendDlgItemMessage Envía un mensaje al control de una caja de diálogo.
SendDriverMessage Envía un mensaje a un driver instalable.
SendMessage Envía un mensaje a un ventana.
SetActiveWindow Elabora un top-level activo en ventana.
SetCapture Captura el ratón en una ventana.
SetCaretBlinkTime Sets the caret blink rate
SetCaretPos Pone la posición del caret.
SetClassLong Pone el largo del valor en la memoria extra clase.
SetClassWord Pone el valor de la palabra en la memoria extra clase.

SetClipboardData Pone el dato en el clipboard.
SetClipboardViewer Adiciona una ventana para una cadena espectador-clipboard.
SetCommBreak Suspende la transmisión de caracteres..
SetCommEventMask Permite eventos en un dispositivo disfrazados de eventos.
SetCommState Pone el estado del dispositivo de comunicación.
SetCursor Cambia el cursor del ratón.
SetCursorPos Pone la posición del cursor del ratón en coordenadas de pantalla.
SetDlgItemInt Convierte un entero a una cadena de texto de una caja de diálogo.
SetDlgItemText Coloca el título de la caja de diálogo o el texto del artículo.
SetDoubleClickTime Pone el tiempo del doble clic del ratón.

SetFocus Pone el enfoque de entrada para la ventana.
SetKeyboardState Pone la tabla de estado del teclado.
SetMenu Pone el menu para una ventana.
SetMenuItemBitmaps Asocia un bitmap con un artículo del menu.
SetMessageQueue Crea una nueva cola de mensajes.
SetParent Cambia una ventana padre a hija.
SetProp Adiciona ó cambia la entrada de lista-propiedad.
SetRect Pone las dimensiones de un rectangulo.
SetRectEmpty Crea un rectangulo vacio.

SetScrollPos Pone la posición del indicador de la barra-scroll.
SetScrollRange Pone las posiciones máximo y mínimo en la barra-scroll.
SetSysColors Pone uno ó mas colores de sistema.
SetSysModalWindow Elabora una ventana el modal-sistema de ventana.
SetTimer Instala el tiempo del sistema.
SetWindowLong Pone el largo del valor en la memoria extra ventana.
SetWindowPlacement Pone el estado de presentación de la ventana y las posiciones max/min.
SetWindowPos Pone el tamaño de la ventana, la posición y el orden.
SetWindowsHook Instala una función gancho.
SetWindowsHookEx Instala una función gancho.

SetWindowText Coloca texto en el título capturado ó en el control de ventana.
SetWindowWord Pone el valor de la palabra en memoria extra ventana.
ShowCaret Muestra el caret.
ShowCursor Muestra ó oculta el cursor del ratón.
ShowOwnedPopups Muestra ó oculta el pop-up de Windows.
ShowScrollBar Muestra ó oculta la barra scroll.
ShowWindow Pone el estado visible de la ventana.
SubtractRect Creates rect from difference of two rects
SwapMouseButton Invierte el significado de los botónes del ratón.

SystemParametersInfo Pregunta ó pone los parámetros del sistema-ancho.
TabbedTextOut Escribe una tabla de cadena de carácteres.
TrackPopupMenu Despliega y sigue un menu pop-up
TranslateAccelerator Procesa comandos del menu aceleradores de teclado.
TranslateMDISysAccel Procesa aceleradores de teclado MDI.
TranslateMessage Traduce mensajes llave-virtual.
TransmitCommChar Situa un caractér en la cola de transmisión.
UngetCommChar Pone un caractér en la cola de recibir.
UnhookWindowsHook Quita una función filtro.
UnhookWindowsHookEx Quita una función de la cadena gancho.

UnionRect Crea la unión de dos rectangulos.
UnregisterClass Quita la clase ventana.
UpdateWindow Actualiza el área cliente de ventana.
ValidateRect Quita un rectangulo de la región actualizada.
ValidateRgn Quita una región de la región actualizada.
WaitMessage Suspende una aplicación y cede el control.
WindowFromPoint Retorna el controlador del punto contenido en la ventana.
WinHelp Invoca la ayúda de Windows.
WNetAddConnection Adiciona la conexión red.
WNetCancelConnection Quita la conexión red.

WNetGetConnection Lista la conexión red.
WriteComm Escribe de un dispositivo de comunicación.
wvsprintf Formato de cadena.

G L O S A R I O .

CONCEPTOS GENERALES.

SISTEMA BINARIO .- DESAFORTUNADAMENTE, EL SISTEMA NUMERICO DECIMAL NO SE PRESTA PARA UNA IMPLANTACION CONVENIENTE EN SISTEMAS DIGITALES. ES POR ELLO QUE UTILIZAMOS EL SISTEMA NUMERICO BINARIO (BASE 2).

EN EL SISTEMA BINARIO SOLAMENTE HAY DOS SIMBOLOS O POSIBLES VALORES DIGITALES, EL 0 Y EL 1. CON LOS CUALES PODEMOS REPRESENTAR CUALQUIER CANTIDAD QUE SE PUEDA DENOTAR EN DECIMAL U OTROS SISTEMAS NUMERICOS. EN EL SISTEMA BINARIO PARA REPRESENTAR CUALQUIER CANTIDAD, POR LO GENERAL SE REQUIEREN DE MUCHOS DIGITOS.

EL SISTEMA BINARIO ES ASIMISMO UN SISTEMA DE VALOR POSICIONAL, EN DONDE CADA DIGITO BINARIO TIENE VALOR PROPIO EXPRESADO COMO POTENCIA DE 2. (EL CERO SE TOMA COMO NUMERO). POR EJEMPLO: EL 1011.101, LOS ESPACIOS A LA IZQUIERDA DEL PUNTO BINARIO SON POTENCIAS POSITIVAS DE 2, Y LOS DE LA DERECHA SON POTENCIAS NEGATIVAS DE 2. PARA DETERMINAR SU

EQUIVALENCIA EN EL SISTEMA DECIMAL SIMPLEMENTE TOMAMOS LA SUMA DE LOS PRODUCTOS DE CADA VALOR DIGITAL (0 O 1) POR SU VALOR POSICIONAL. ES DECIR:

1011.101 (base 2) = (1*2 a la potencia 3) + (0*2 a la 2) + (1*2 a la 1) + (1*2 a la 0) + (1*2 a la -1) + (0*2 a la -2) + (1*2 a la -3) = 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125
= 11.625 (base 10)

EN LA OPERACION ANTERIOR, LOS SUBINDICES 2 Y 10 SE UTILIZAN PARA INDICAR LA BASE DEL NUMERO INDICADO.

EN EL SISTEMA BINARIO, A MENUDO EL TERMINO DIGITO BINARIO SE ABREVIA COMO BIT. EN EL EJEMPLO ANTERIOR TENEMOS ENTONCES 4 BITS A LA IZQUIERDA DEL PUNTO Y 3 A LA DERECHA. EL BIT MAS SIGNIFICATIVO (MSB) ES AQUEL MAS A LA IZQUIERDA DEL PUNTO (EL QUE TIENE EL MAYOR VALOR). EL BIT MENOS SIGNIFICATIVO (LSB) ES AQUEL MAS A LA DERECHA DEL PUNTO (EL DE MENOR VALOR). SI UN NUMERO NO PRESENTARA PUNTO BINARIO, DE IGUAL MANERA SE APLICAN LOS CONCEPTOS ANTERIORES.

NOTA: UN GRUPO DE 8 BITS = 1 BYTE.

CONTEO BINARIO.- AL UTILIZAR LOS NUMEROS BINARIOS, NOS ENFRENTAMOS CON EL PROBLEMA DE CUANTOS BITS UTILIZAREMOS. ESTO ES PORQUE LOS CIRCUITOS POR MEDIO DE LOS CUALES REPRESENTAMOS A ESTOS NUMEROS SON DE CAPACIDAD LIMITADA EN CUANTO AL NUMERO DE BITS QUE PUEDEHN MANEJAR.

POR EJEMPLO, SI MANEJAMOS 4 BITS, EL NUMNERO MAXIMO QUE PODREMOS REPRESENTAR ES EL 15 ES DECIR SI TENEMOS A LOS CUATRO BITS EN 1, DE LA FORMA QUE SIGUE: 1111 (BASE 2) = 15 (BASE 10). EL NUMERO MAXIMO TOMANDO EL VALOR POR POSICION SERA EL 15 EN DECIMAL.

NOTA: EN EL SISTEMA BINARIO SIEMPRE COMENZAMOS A CONTAR A PARTIR DEL 0,(VALOR MAS PEQUE~0).

LA SIGUIENTE TABLA MUESTRA TODAS LAS COMBINACIONES POSIBLES CONSIDERANDO 4 BITS:

2 A LA 3=8 2 A LA 2= 4 2 A LA 1=2 2 A LA 0=1

0	0	0	0	=	0
0	0	0	1	=	1
0	0	1	0	=	2
0	0	1	1	=	3
0	1	0	0	=	4
0	1	0	1	=	5
0	1	1	0	=	6
0	1	1	1	=	7
1	0	0	0	=	8
1	0	0	1	=	9
1	0	1	0	=	10
1	0	1	1	=	11
1	1	0	0	=	12
1	1	0	1	=	13
1	1	1	0	=	14
1	1	1	1	=	15

EN EL SISTEMA BINARIO, UTILIZANDO N BITS, PODEMOS REALIZAR HASTA 2 A LA N CONTEOS. POR EJEMPLO CON LOS 4 BITS ANTERIORES TENEMOS 2 A LA 4 = 16 CONTEOS ((0000 A 1111) BASE 2.

REPRESENTACION DE CANTIDADES BINARIAS.- EN LOS SISTEMAS DIGITALES, LA INFORMACION QUE ESTA SIENDO PROCESADA POR LO GENERAL SE PRESENTA EN FORMA BINARIA. LAS CANTIDADES BINARIAS SE PUEDEN REPRESENTAR POR MEDIO DE CUALQUIER DISPOSITIVO QUE SOLAMENTE TENGA DOS ESTADOS DE OPERACION O POSIBLES CONDICIONES. POR EJEMPLO, UN INTERRUPTOR TIENE SOLO DOS ESTADOS: ABIERTO Y CERRADO. PODEMOS HACER QUE UN INTERRUPTOR ABIERTO REPRESENTE EL 0 BINARIO O BIEN QUE UNO CERRADO REPRESENTE EL 1 BINARIO. CON ESTA ASIGNACION PODEMOS REPRESENTAR CUALQUIER NUMERO BINARIO.

POR EJEMPLO, CONSIDEREMOS 5 INTERRUPTORES, DONDE CERRADOS REPRESENTAN EL 1 BINARIO Y ABIERTO EL 0 BINARIO.

1 ABIERTO 2 CERRADO 3 CERRADO 4 ABIERTO 5 ABIERTO= 01100

EXISTEN MUCHOS OTROS DISPOSITIVOS CON DOS ESTADOS DE OPERACION O BIEN QUE PUEDEN OPERAR EN DOS CONDICIONES EXTREMAS.

TRANSMISION DE BITS - EXISTEN DOS MANERAS POR MEDIO DE LAS CUALES PODEMOS TRANSMITIR UN GRUPO DE BITS. UNA ES EN PARALELO Y LA OTRA EN SERIE.

EN LA TRANSMISION EN PARALELO VAMOS A TRANSMITIR LOS BITS EN GRUPOS TAN GRANDES COMO LO PERMITA LA COMPUTADORA, TODOS SIMULTANEAMENTE. POR EJEMPLO UNA COMPUTADORA DE 8 BITS

(RECORDAR QUE 8 BITS = 1 BYTE). TRANSMITE 8 BITES A LA VEZ.

EN LA TRANSMISION EN SERIE VAMOS A TRANSMITIR UN BIT A LA VEZ, UNO POR UNO (ESTA TRANSMISION ES MAS LENTA).

MEMORIA.- LOS DISPOSITIVOS Y CIRCUITOS DE MEMORIA DESEMPE~AN UN PAPEL IMPORTANTE EN LOS SISTEMAS DIGITALES DEBIDO A QUE OFRECEN MEDIOS PARA ALMACENAR NUMEROS BINARIOS TEMPORAL O PERMANENTEMENTE, CON LA CAPACIDAD DE CAMBIAR LA INFORMACION ALMACENADA EN CUALQUIER INSTANTE. LOS DIVERSOS DISPOSITIVOS DE MEMORIA INCLUYEN CINTAS MAGNETICAS Y OTROS UTILIZAN CIRCUITOS ELECTRONICOS DENOMINADOS MULTIVIBRADORES BIESTABLES (FLIP - FLOPS).

CELDA BINARIA.- DISPOSITIVO O CIRCUITO ELECTRICO QUE SE HUSA PARA ALMACENAR UN SOLO BIT (0 O 1). POR EJEMPLO EL FLIP - FLOP.

PALABRA DE MEMORIA.- GRUPO DE BITS (CELDAS) EN UNA MEMORIA QUE REPRESENTA INSTRUCCIONES O DATOS DE ALGUN TIPO. POR EJEMPLO, UN REGISTRO QUE CONSTA DE OCHO FLIP -FLOPS PUEDE CONSIDERARSE COMO UNA MEMORIA QUE ALMACENA UNA PALABRA DE 8 BITS. LOS TAMA~OS DE PALABRA EN LAS COMPUTADORA VARIA DE 4 BITS A 64 BITS, SEGUN LA DIMENSION DE LA COMPUTADORA.

CAPACIDAD.- FORMA DE ESPECIFICAR CUANTOS BITS PUEDEN ALMACENARSE EN UN DISPOSITIVO DE MEMORIA PARTICULAR O BIEN EN UN SISTEMA DE MEMORIA COMPLETO. POR EJEMPLO SUPONGASE UNA MEMORIA QUE ALMACENA 4096 PALABRAS DE 20 BITS CADA UNA. ESTO SIGNIFICA UNA CAPACIDAD TOTAL DE 81920 BITS.(4096 * 20 O EXPRESADO DIFERENTE 4K * 20).

NOTA: EL NUMERO DE PALABRAS CONTENIDAS EN UNA MEMORIA A MENUDO ES MULTIPLO DE 1024 BITS. UTILIZAREMOS LA EQUIVALENCIA DE 1024 BITS = 1 K.

DIRECCION.- NUMERO QUE IDENTIFICA LA LOCALIDAD DE UNA PALABRA EN LA MEMORIA. CADA PALABRA ALMACENADA EN UN DISPOSITIVO DE MEMORIA O SISTEMA DE MEMORIA TIENE UNA DIRECCION UNICA. LAS DIRECCIONES SIEMPRE SE ESPECIFICAN COMO UN NUMERO BINARIO, AUNQUE ALGUNAS VECES PODEMOS UTILIZAR NUMEROS HEXADECIMALES, DECIMALES Y OCTALES.

OPERACION DE LECTURA.- ES LA OPERACION CON LA CUAL LA PALABRA BINARIA ALMACENADA EN UNA LOCALIDAD (DIRECCION) ESPECIFICA DE LA MEMORIA ES CAPTADA Y DESPUES TRANSFERIDA AQ OTRA LOCALIDAD.

OPERACION DE ESCRITURA.- OPERACION POR MEDIO DE LA CUAL SE COLOCA UNA NUEVA PALABRA EN CIERTA LOCALIDAD DE LA MEMORIA. TAMBIEN SE LE LLAMA OPERACION DE ALMACENAR . SIEMPRE QUE UNA NUEVA PALABRA SE ESCRIBE EN UNA LOCALIDAD DE LA MEMORIA , ESTA RECOLOCA LA PALABRA QUE SE ENCONTRABA ANTES AHI.

TIEMPO DE ACCESO.- MEDIDA DE LA VELOCIDAD DEL DISPOSITIVO DE MEMORIA. ES LA CANTIDAD DE TIEMPO QUE SE REQUIERE PARA REALIZAR UNA OPERACION DE LECTURA. ES DECIR, ES EL TIEMPO ENTRE LA MEMORIA QUE RECIBE UNA SE~AL DE COMANDO Y LOS DATOS

QUE SE PONEN A DISPOSICION EN LA SALIDA DE LA MEMORIA.

TIEMPO DE CICLO.- ES LA CANTIDAD DE TIEMPO NECESARIA PARA QUE LA MEMORIA REALICE UNA OPERACION DE LECTURA O ESCRITURA Y DESPUES REGRESE A SU ESTADO ORIGINAL LISTA PARA EJECUTAR EL SIGUIENTE COMANDO. ESTE TIEMPO ES NORMALMENTE MAS LARGO QUE EL TIEMPO DE ACCESO.

MEMORIA VOLATIL.- CUALQUIER TIPO DE MEMORIA QUE REQUIERE LA APLICACION DE ENERGIA ELECTRICA PARA ALMACENAR INFORMACION. SI SE RETIRA LA ENERGIA, TODA LA INFORMACION ALMACENADA EN LA MEMORIA SE PERDERA. MUCHAS MEMORIA SEMICONDUCTORAS SON VOLATILES, MIENTRAS QUE LAS MEMORIA MAGNETICAS NO LO SON.

MEMORIA DE ACCESO ALEATORIO (RAM).- MEMORIA EN LA CUAL LA LOCALIZACION FISICA REAL DE UNA PALABRA DE LA MEMORIA NO TIENE EFECTO SOBRE EL TIEMPO QUE TARDE EN LEER DE ESA LOCALIDAD O BIEN ESCRIBIR EN ELLA. ES DECIR, EL TIEMPO DE ACCESO ES EL MISMO PARA CUALQUIER DIRECCION EN LA MEMORIA. MUCHAS MEMORIAS SEMICONDUCTORAS Y DE NUCLEO MAGNETICO SON RAM.

MEMORIA CAHE.- SECCION DE ALTA VELOCIDA DE RAM, QUE SIRVE PARA ALMACENAR APUNTADORES, IMAGENES DE REGISTROS,PROGRAMAS Y DATOS. PUEDE UTILIZARSE PARA ALMACENAR DATOS DEL USUARIO ACTIVO EN ESE MOMENTO EN UN SISTEMA MULTIUSUARIO.

MEMORIA DE ACCESO SECUENCIAL (SAM).- TIPO DE MEMORIA EN LA CUAL EL TIEMPO DE ACCESO NO ES CONSTANTE, SINO QUE VARIA SEGUN LA LOCALIDAD DE LA DIRECCION. CIERTA PALABRA ALMACENADA ES HALLADA POR SUCESION A TRAVES DE TODAS LAS LOCALIDADES DE DIRECCIONES HASTA QUE SE LLEGA A LA DIRECCION DESEADA. (PRODUCE TIEMPOS DE ACCESO MUY LARGOS). ALGUNOS DE ESTOS DISPOSITIVOSSON:EL DISCO MAGNETICO, LA CINTA Y LA MEMORIA DE BURBUJA MEGNETICA.

MEMORIA PARA LECTURA Y ESCRITURA (RWM).- CUALQUIER MEMORIA DE LA QUE SE PUEDA LEERSE INFORMACION O BIEN ESCRIBIRSE EN ELLA CON LA MISMA FACILIDAD.

MEMORIA SOLO PARA LECTURA (ROM) - MEMORIA SEMICONDUCTORA DISE-ADA PARA APLICACIONES DONDE LA PROPORCION DE OPERACIONES DE LECTURA A OPERACIONES DE ESCRITURA ES MUY ALTA. EN UNA ROM SOLO PUEDE ESCRIBIRSE (PROGRAMARSE) UNA VEZ Y ESTA OPERACION NORMALMENTE ES EFECTUADA EN LA FABRICA. EN LA ROM SOLO PUEDE LEERSE. EXISTEN OTROS TIPOS DE ROM CON ALGUNAS VARIANTES.

DISPOSITIVOS DE MEMORIA ESTATICA.- DISPOSITIVOS DE MEMORIA SEMICONDUCTORA EN LOS CUALES LOS DATOS ALMACENADOS SE QUEDARAN PERMANENTEMENTE GUARDADOS EN TANTO SE APLIQUE ENERGIA ELECTRICA, SIN NECESITAR REESCRIBIR PERIODICAMENTE LOS DATOS EN LA MEMORIA.

DISPOSITIVOS DE MEMORIA DINAMICA.- DISPOSITIVOS DE MEMORIA SEMICONDUCTORA EN LOS CUALES LOS DATOS ALMACENADOS NO SE QUEDARAN PERMANENTEMENTE GUARDADOS,AUN CON ENERGIA APLICADA, A MENOS QUE LOS DATOS SE REESCRIBAN EN FORMA PERIODICA EN LA

MEMORIA. A ESTA OPERACION SE LE CONOCE COMO OPERACION DE REFRESCO.

INSTANCIA.- UNA SOLA APARICION DE UNA ENTIDAD GRAFICA EN UNA ESCENA O IMAGEN. UNA COPIA QUE SE EJECUTA DE UNA APLICACION WINDOWS. SE PUEDE ESTAR EJECUTANDO MAS DE UNA COPIA A LA VEZ.

CONCLUSIONES:

Al trabajar con una plataforma gráfica como lo es Windows, nos damos cuenta de su verdadero poder en el desarrollo de aplicaciónes. Windows es la tendencia a seguir, ya que su interfaz gráfica ofrece gran cantidad de recursos a explotar, con lo cual la unica limitante en el desarrollo de aplicaciónes sera nuestro propio ingenio de programación.

La aplicación que se desarrollo solo es una pequeña muestra de las posiblilidades de desarrollo que Windows nos brinda. Recordemos que el lenguaje a utilizar para programar con Windows queda abierto a su selección.

Para llegar a ser un experto programador bajo Windows, tendremos que dedicarle tiempo de estudio; un buen método para aprender a programar es estudiando los còdigos desarrollados por gente experta, así como la constante práctica en el desarrollo de programas.

BIBLIOGRAFIA:

- SISTEMAS DIGITALES.
  RONALD J. TOCCI
  PRENTICE HALL.

- PROGRAMACIÓN AVANZADA DE GRAFICOS EN C PARA WINDOWS.
  LEE ADAMS
  MC GRAW HILL.

- ENCICLOPEDIA DE TERMINOS DE MICROCOMPUTACIÓN.
  LINDA GAIL CHRISTIE/JOHN CHRISTIE
  PRENTICE HALL.

- PROGRAMACIÓN EN WINDOWS 3.1
  WILLIAM H. MURRAY Y CHRIS H. PAPPAS
  MC GRAW HILL.