UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

ELEMENTOS DE INTERFACE CON EL USUARIO: ANALISIS, DISEÑO E IMPLEMENTACION CON LENGUAJE C EN PC.

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

P r e s e n t a :

MIGUEL ANGEL RODRIGUEZ RUBIO

San Juan de Aragón, Estado de México 1995

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
CAMPUS ARAGÓN

UNIDAD ACADÉMICA

Ing. SILVIA VEGA MUYTOY
Jefe de la Carrera de Ingeniería
en Computación,
P r e s e n t e .

En atención a la solicitud de fecha 4 de octubre del año en curso, por la que se comunica que el alumno MIGUEL ÁNGEL RODRÍGUEZ RUBIO, de la carrera de Ingeniero en Computación, ha concluido su trabajo de investigación intitulado "ELEMENTOS DE INTERFACE CON EL USUARIO: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN CON LENGUAJE C EN PC", y como el mismo ha sido revisado y aprobado por usted se autoriza su impresión; así como la iniciación de los trámites correspondientes para la celebración del examen profesional.

Sin otro particular, le reitero las seguridades de mi distinguida consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPÍRITU
San Juan de Aragón, México., 5 de octubre de 1995
EL JEFE DE LA UNIDAD

Lic. ALBERTO IBARRA ROSAS

c c p Asesor de Tesis.
c c p Interesado.

AIR'lla.

AGRADECIMIENTOS.

A mis padres por todo su apoyo durante mis estudios, a mis hermanos, a todos mis compañeros por su amistad y compañía, a todos aquellas personas que hicieron posible la realización de esta tesis: maestros y amigos; a mi asesor de tesis por su apoyo y entusiasmo mostrado ante este
tema; a todos ellos quisiera darles mis mas sinceros agradecimientos por auxiliarme y darme todo su apoyo para poder seguir adelante en esta carrera; no quiero dar sus nombres para evitar la exclusión errónea de alguno de ellos.

Gracias a su valiosa ayuda fue posible la realización de este material, el cual fue creado con gran esfuerzo y voluntad, esperando que sea una valiosa fuente de consulta y aprendizaje para muchas personas; con el fin de continuar por el mismo camino de todas aquellas personas que contribuyeron en mi desarrollo profesional, yo intento seguirlo con el desarrollo de esta tesis; esperando que incremente los conocimientos de los lectores.

TEMA

ELEMENTOS DE INTERFACE CON EL USUARIO : ANÁLISIS, DISEÑO E IMPLEMENTACIÓN CON LENGUAJE C EN PC.

OBJETIVO : El objetivo principal de esta tesis consiste en introducir al usuario el los conceptos de los elementos de interface, sus características, importancia, usos, limitaciones y ventajas de su utilización dentro de nuestros programas, así como la generación de rutinas de algunos de estos elementos de interface, las cuales se realizarán en el lenguaje de programación C, se incluirá una sección en esta tesis explicando el porque se utilizó este lenguaje de programación en la elaboración de las funciones.

La idea básica y mas importante es que el usuario identifique a los elementos de interface, conozca sus características y algunos métodos para implementarlos, así como auxiliarse de las funciones que se generen para el desarrollo de sus programas; o genere sus propias rutinas sobre los elementos que se exponen en esta tesis u otros nuevos si así se requiere.

TEMARIO

CONVENCIONES :

En el desarrollo de esta tesis se tomaran una serie de convenciones con el fin de diferenciar algunas referencias a partes de código o programas del texto normal; las principales convenciones utilizadas son las siguientes :

CURSIVA -> VARIABLES
ejemplo: *TOTAL*

NEGRILLA Y CURSIVA -> FUNCIONES
ejemplo: ***salva_pantalla()***

NEGRILLA Y SUBRAYADO -> REGISTROS DEL MICROPROCESADOR
ejemplo: **AX**, **BX**, **CX**, **DX**

DOBLE SUBRAYADO -> REFERENCIA A TECLAS
ejemplo: INICIO, FIN, INSERT, RE. PAG.

NEGRILLA ITALICA -> NUMEROS DE INTERRUPCIONES
ejemplo: ***10H, 33H***

NEGRILLA ROMAN -> DIRECCIONES DE MEMORIA
elemplo : **A000:0000,B800:000**

INTRODUCCIÓN

En la actualidad, el uso de la computadora se ha extendido a prácticamente todas las áreas de trabajo como son bancos, industrias, escuelas, oficinas, hospitales, locales comerciales e inclusive en el propio hogar; la computadora ha dejado de ser un elemento al cual tenían acceso solamente un pequeño grupo de profesionales capacitados para su uso, operación y mantenimiento; hoy en día, la computadora puede tener aplicaciones en todas las áreas, y prácticamente cualquier persona tiene acceso a una de ellas, ya sea como forma de su trabajo, parte de su enseñanza, diversión o complemento de alguna actividad en la cual se puede aplicar su uso.

Ante este increíble aumento tan rápido en sus aplicaciones, el software experimento grandes cambios tanto en beneficio como en contra del usuario; el lado negativo radica en el costo del software, el cual ha tendido a incrementarse en los últimos años; sin embargo, este costo se ve influido por la cantidad y calidad de tareas que se puede realizar con él, ya que últimamente, el software ha crecido en este aspecto, siendo cada vez más completo, lo cual forma la parte positiva de este punto. Ante el aumento de sus aplicaciones y tipo de usuarios el software no solamente debía de satisfacer una necesidad, tener una aplicación práctica, sino que ahora tenía que incluir elementos que facilitaran su utilización y aprendizaje; a estos elementos se les denomina "ELEMENTOS DE INTERFACE CON EL USUARIO", y son de suma importancia debido a que es la forma de interactuar un sistema con el operador y representa la primera impresión que el usuario tiene del sistema.

Los ELEMENTOS DE INTERFACE CON EL USUARIO pueden tener una gran variedad de formas, debido a la gran diversidad de criterios

que se pueden formar entre distintas personas; de hecho, cada persona puede imaginarse una forma diferente de establecer una comunicación entre un sistema y un usuario; sin embargo, algunos de los elementos de interface han sobresalido en forma notable debido a su facilidad de manejo y aprendizaje, y son herramientas muy utilizadas en el desarrollo de software profesional; el propósito de esta tesis comprende el analizar y desarrollar algunos de estos elementos de uso generalizado en el desarrollo de programas.

Solamente se desarrollarán las rutinas de algunos de los elementos de interface debido a la gran cantidad y variantes que existen de ellos, y analizar e implementar funciones para cada uno sería un trabajo muy extenso, incluso, hay temas que se puede profundizar bastante y podían ser motivo de una tesis debido a la cantidad de material que existe e implementación de rutinas que se pueden crear, sin embargo, el material que se seleccionó para esta tesis comprende los elementos mas comunes y populares, con el fin de que los lectores se introduzcan en ellos, su utilización y ventajas, además de poder aplicarlos en sus programas, realizar cambios para adaptarlos a sus necesidades, e incluso, generar sus propias rutinas tomando como base las que se incluyen, de manera que se pueda tener una librería propia la cual pueda complementarse de acuerdo a las necesidades.

Los elementos de interface con el usuario que se analizarán incluyen tanto elementos en modo texto, como gráficas (**GUI Graphical User Interface**), estos últimas con una tendencia a seguir evolucionando debido a las ventajas que tiene sobre las interfaces en modo texto, como pueden ser: combinación de texto con gráficos, mayor resolución, una mejor presentación, etc. Estos elementos influyen en gran parte en la decisión o preferencia hacia un programa en especial, de aquí surge su gran importancia que tienen en el desarrollo de software de cualquier tipo. Los elementos de interface han tenido un gran auge en los últimos años debido a la gran competencia que ha existido entre las grandes empresas que se dedican a la creación de todo tipo de software, ya que dependerá en gran parte de los elementos de interface que presente sus programas la decisión o preferencia del usuario hacia alguno de ellos en especial.

hoy en día, ante una necesidad de un usuario, o grupo de ellos, pueden existir varias alternativas de solución con una gran variedad de software, los cuales pueden presentar una gama de poderosas herramientas que resuelvan su problema, ante esto, el usuario opta por aquel sistema que sea mas económico, le facilite la realización de su tarea, sea mas rápido, que pueda aprender a utilizarlo de una manera sencilla y rápida, para que de esta manera, pueda explotarlo y obtener provecho mas rápidamente; los programas o sistemas que son mas populares combinan todos estos aspectos : herramientas, elementos de interface y costo; de manera que el usuario se decida por su producto.

Se ha establecido en los últimos años una batalla entre las grandes firmas o compañías creadoras de software, cuyo único fin es la de ganarse a el usuario, y de esta forma ser la empresa líder en el mercado de venta de software, esto se puede lograr mediante los siguientes aspectos fundamentales:

- Crear software de calidad con poderosas herramientas.
- Que el software satisfaga una necesidad en su totalidad.
- Utilizando elementos de interface con el usuario eficientes.
- Que el costo no sea muy elevado.

Hasta el momento, ninguna compañía ha ganado totalmente la guerra por el usuario, aunque existen algunos productos de compañías que tienen una mayor venta que otros debido a su gran impacto que han tenido en ellos, sin embargo, la batalla no ha sido ganado por ninguna compañía, las cuales continúan mejorando el nivel del software en todos sus aspectos, tanto en herramientas, como en elementos de interface, para obtener la preferencia del usuario y tener un mayor nivel de ventas. Ante esta perspectiva, el futuro de una mejor y más fácil interacción usuario-sistema esta asegurado, y se prevé un gran auge en su desarrollo en los próximos años, lo cual va a resultar de gran beneficio hacia los millones de operadores que diariamente utilizan las computadoras como herramientas indispensables en su trabajo, así como en aquellos

usuarios potenciales de sistemas de cómputo.

El área de Hardware también esta estrechamente ligada a el desarrollo de elementos de interface, ya que contribuye en el desarrollo de nuevos y mejores monitores, con una mayor resolución, nuevos y mejores dispositivos apuntadores, equipos con una mayor velocidad de procesamiento, etc., lo cual viene a fortalecer el avance y futuro de los elementos de interface, y además, a mejorar aquellos elementos ya existentes y que han sobresalido en forma notable debido a su sencilla utilización.

CAPITULO I

DEFINICIÓN E IMPORTANCIA DE LOS ELEMENTOS DE INTERFACE Y VENTAJAS DEL LENGUAJE C EN LA IMPLEMENTACIÓN DE LAS FUNCIONES.

1.1 VENTAJAS DEL USO DE LA PC.

En la actualidad, el uso de las computadoras se ha extendido a prácticamente todas las áreas de trabajo, sin embargo, existen una gran diversidad o gama de plataformas sobre las cuales se puede trabajar, cada una con sus características propias; uno de los objetivos de esta tesis, es la de explicar al usuario que es un elemento de interface, y cuales son las características con las cuales podemos diferenciarlos; otro objetivo fundamental propuesto consiste en la generación de una serie de funciones las cuales puedan implementarse en los programas para que estos sean de mejor calidad; para ello se necesitaba definir bajo que plataforma se trabajaría, ya que seria imposible analizarlas todas; se eligió utilizar la PC debido a 3 razones fundamentales:

1.- La PC es líder en el mercado de ventas de equipo hoy en día, superando ampliamente a todas la demás plataformas, según datos obtenidos de estadísticas sobre el mercado de ventas (Ver figura 1.1 y 1.2).

2.- Al tener un mayor numero de venta; la mayoría de los usuario, y por lo tanto de los lectores potenciales de esta tesis, podrán tener acceso a una de ellas, con el fin de observar los elementos de interface, y si así se desea, practicar con las funciones que se crearan posteriormente.

3.- Existen una mayor variedad de aplicaciones disponibles bajo esta plataforma, bajo DOS y/o WINDOWS; teniendo un numero mayor de usuarios este tipo de aplicaciones.

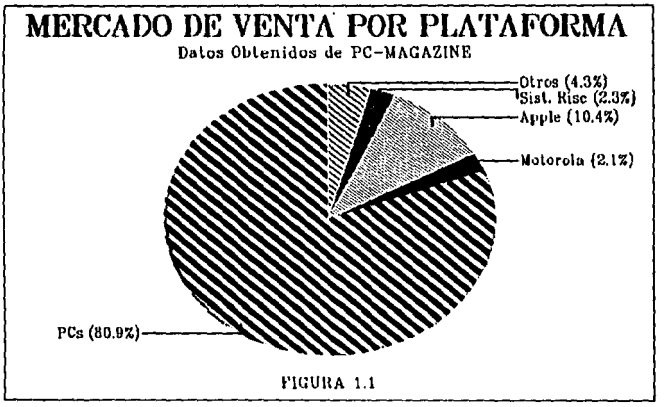


FIGURA 1.1

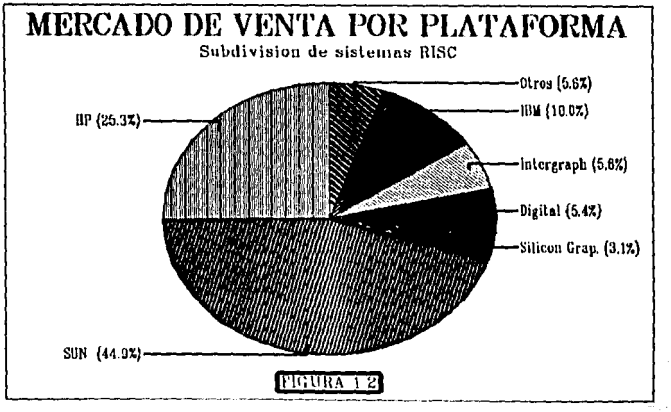


FIGURA 1.2

Esto no significa que la PC sea mas poderosa y rápida que todos los demás equipos; ni tampoco que sea la plataforma ideal para implementar alguna aplicación en concreto, ya que para establecer alguna conclusión de este tipo, necesitaríamos evaluar las características de los equipos, así como el software existente bajo todas las plataformas que se relaciones con nuestras necesidades; tomando en cuenta la cantidad de datos que se van a manejar y la rapidez con que se pueden obtener resultados, realizando comparaciones entre los distintos programas que existen y analizando sus ventajas, desventajas, diferencias y limitaciones.

Al ser la PC líder en ventas, un numero mayor de usuarios tiene acceso a una ellas, o puede tener alguna en casa, a través de la cual compruebe las ventajas del uso estos elementos en los sistemas, si el lector es un programador o ingeniero, seguramente comprenderá que implementar una buena interface con el usuario es muy difícil, que en ocasiones, cuesta mas trabajo que el proceso mismo; llegando a consumir en ocasiones la mitad del código fuente en el uso de estos elementos; sin embargo, los resultados son satisfactorios, y el usuario final tiene una serie de ventajas al utilizar el sistema.

En la actualidad existen bajo todas las plataformas interfaces graficas con el usuario (GUI[1]), las cuales hacen uso de ventanas, menús y otros objetos gráficos; además de que habilitan a el usuario a interactuar en la aplicación con el uso del ratón; los principales interfaces graficas que podemos encontrar están:

PLATAFORMA	GUI
PC's	MICROSOFT WINDOWS
WORKSTATION	OPEN LOOK OSF/MOTIF
NeXTSTATION	NeXTSTEP
APPLE MACINTOSH	MACINTOSH OPERATING SYSTEM

TABLA 1.1

[1] De las iniciales del inglés **G**raphical **U**ser **I**nterface

Cada uno de estos GUI contienen métodos de interactuar con la aplicación, lo cual le proporciona una distinta apariencia (LOOK) y comportamiento (FEEL), lo que muchos autores denominan "LOOK AND FEEL"; cada GUI esta basado en un conjunto de reglas para su diseño, y contiene una libreria de rutinas (TOOLKIT) con una interface de programación bien definida, el cual también es como API[2], el cual habilita a los programadores a realizar aplicaciones que puedan hacer uso de las facilidades del sistema de ventanas; desafortunadamente, cada uno de estos puntos difiere de un GUI a otro perdiéndose la consistencia entre una plataforma a otra; cada GUI tiene su único estilo y terminología, sin embargo, proveen facilidades para construir interfaces de usuario para aplicaciones.

Los elementos de interface que se explican en esta tesis son muy generales bajo todas las plataformas, siendo sus características principales muy similares; y prácticamente son utilizadas en todos los nuevos programas cualquiera que sea su plataforma; para efecto de esta tesis, solamente la aplicación será totalmente basada en PC bajo DOS.

En la actualidad existen algunos productos en el mercado, los cuales funcionan bajo alguna plataforma anterior, y proporcionan herramientas para el uso y diseño de interfaces gráficas, sin embargo, en la mayoría de los casos, los requerimientos para su funcionamiento son excesivos, requiriendo gran cantidad de memoria y velocidad rápida, lo cual en muchos casos no se cuenta; sin embargo, hay que hacer la aclaración, muchos de estos paquetes contienen elementos muy buenos, y acompañado de un buen equipo, se tiene resultados excelentes, los cuales, no se podían obtener anteriormente; otro aspecto que influye en la compra de este tipo de software es el costo, ya que en muchos de los casos es excesivo. Algunos de este tipo de programas existentes en el mercado son los siguientes[3]:

---

[2] De las iniciales en Inglés **A**plication **P**rogramming **I**nterface

[3] Datos obtenidos de PC MAGAZINE. "RAD TOOLS". Noviembre 8, 1994 Volumen 13 Numero 19

HERRAMIENTAS DE DESARROLLO DE GUI's			
VENDEDOR	PRODUCTO	SISTEMA OPERATIVO SOPORTADO	GUI
META SOFTWARE	Design/OA	PC/MS-DOS APPLE MACINTOSH	Microsoft Windows Macintosh
MOZART SYSTEMS	Mozart	PC/MS-DOS	Microsoft Windows
MULTI-SOFT	Easy5AA	PC/MS-DOS	Microsoft Windows
NEURON DATA	Open Interface	APPLE MACINTOSH PC/MS-DOS UNIX; VAX/VMS OS/2	Open Look, Motif Microsoft Windows Macintosh
NON STANDARD LOGICS	WISH2	UNIX	Motif X-Windows
PARCPLACE SYSTEMS	Objectworks/ Smalltalk	VAX/VMS UNIX PC/MS-DOS MACINTOSH	Microsoft Windows Motif; Open Look Macintosh
POWERSOFT	Power Builder	PC/MS-DOS	Microsoft Windows
RAPID SYSTEM DEVELOPMENT	Hyper Analist	PC/MS-DOS	Microsoft Windows
SERVIO LOGIC CORP.	GemSTone	PC/MS-DOS VAX/VMS SU UNIX	Object-Oriented DBMS Interface
THE SOFTWARE ORGANIZATION	Dialog Coder	PC/MS-DOS	Microsoft Windows
TELESOFT	Telouse	VAX/VMS UNIX	Motif; Open Look
UNIFY CORP.	Accell/SQL	UNIX	Motif; Open Look Microsoft Windows
UNISYS	Unisys Designer WorkBench	PC/MS-DOS	Microsoft Windows
VIEW POINT SYSTEMS	I/F Builder	PC/MS-DOS	Microsoft Windows
V.I. CORP.	DataViews	PC/MS-DOS VAX/VMS SU; HP	Microsoft Windows Open Look

HERRAMIENTAS DE DESARROLLO DE GUI's			
XVT CORP.	Extensive Virtual Toolkit (XTV)	UNIX PC/MS-DOS	Open Look; X-Windows; Motif Microsoft Windows Macintosh
THE WHITEWATER CORP.	Actor	PC/MS-DOS	Microsoft Windows

TABLA 1.2

1.2 DEFINICIÓN E IMPORTANCIA DE LOS ELEMENTOS DE INTERFACE CON EL USUARIO.

1.2.1 DEFINICIÓN E IMPORTANCIA.

Los elementos de interface con el usuario son los medios con los cuales un programa o sistema interactua con el operador para facilitarle la utilización de todas las herramientas que contenga y auxiliarlo en sus funciones y uso; además se pueden utilizar varios de ellos en un mismo sistema; todo ello encaminado hacia un fin, auxiliar y facilitar a el usuario el uso, manejo y comprensión del sistema, "Una interface con el usuario es mas que la apariencia de una aplicación en la pantalla, son todas las formas con las que la aplicación se comunica con el usuario, y el usuario con la aplicación"[4]. Un buen elemento de interface presenta las siguientes características:

- Resulta fácil de utilizar.
- El aprendizaje sobre su uso es de forma sencilla.
- Presenta una velocidad de respuesta rápida.
- Ofrece versatilidad y rapidez al sistema.

[4] NeXTSTEP: USER INTERFACE GUIDELINES. Ed. Addison Wesley Publishing company. Primera Edición. Sept. 1992.

Debido a que los elementos de interface interactuan directamente con el usuario, dependerá de ellos en gran parte la opinión o juicio que se forme del programa, ya que un buen conjunto de herramientas de trabajo combinado con buenos elementos que faciliten el uso de éstas formarán un buen sistema el cual será preferido por los usuarios que tengan la necesidad de utilizar alguno que se relacione con su área de trabajo; de hecho, sistemas que son fáciles de utilizar tiene una cara competitiva en el mercado de venta de software.

Existen algunos programas los cuales no necesitan de su utilización, esto se debe a que realizan solamente una tarea concreta y específica, es decir, son programas sencillos que realizan un proceso y termina su función; sin embargo, a medida que se incrementa el número de tareas que puede realizar, la complejidad en su uso aumenta y se hace indispensable su uso de manera que faciliten su utilización; la tendencia hoy en día en los sistemas de todo tipo es incrementar el número de funciones que efectúan, esto con el fin de formar un sistema mas completo y efectivo; sin embargo, si en éstos no se incluye un buen elemento de interface solamente lograrán que sus usuario encuentren frustración y desesperación al utilizarlo, lo cual causará una mala impresión del software, y el usuario evitará utilizarlo aunque las herramientas del mismo sean excelentes.

Los elementos de interface solamente establecen la comunicación usuario-sistema y viceversa, sin embargo, su importancia radica en que es el medio con el cual se realiza la interface o enlace entre ambos elementos, facilitando de esta manera el uso del mismo, permitiendo que el usuario termine su trabajo mas rápidamente e incrementando la productividad. Un sistema que combine buenos elementos de interface que facilite su manejo y uso, además de buenas herramientas de trabajo que resuelvan completamente algunas necesidades en concreto y presente características como rapidez, buena presentación y ayuda entre otras, además de tener un costo accesible será un sistema el cual tendrá muy buena popularidad y aceptación entre los usuarios, los cuales preferirán este tipo de software.

1.2.2 ANTECEDENTES Y PERSPECTIVAS.

En los principios de la computación el uso de los programas era muy problemático debido a que no presentaban elementos de interface de buena calidad que facilitaran su utilización, o eran muy rudimentarios, así como el equipo en el cual se debería de implantar el programa. Sin embargo, conforme fue aumentando el auge de la computación, éstas se comenzaron a introducir a muy diversas áreas de trabajo como pueden ser : oficinas, fábricas, hospitales, hoteles, bancos, incluso en el mismo hogar; ante éste rápido crecimiento del área de trabajo de la computación, el software también sufrió cambios notables en la forma de interactuar con el usuario. Todo esto se vio favorecido en gran medida con el desarrollo y evolución del equipo físico de la computadora (**HARDWARE**), como puede ser : el uso de monitores a color, dispositivos apuntadores como el ratón, pantallas touchscreen y plumas luminosas; mayor capacidad de almacenamiento, mayor velocidad en los equipos, etc..

Ante todo este crecimiento en el área de aplicación de las computadoras el software también experimento cambios notables, ya que ahora, sus usuarios no eran solamente profesionales dedicados a el área de la computación, los cuales conocían los principios y funcionamiento de la computadora, sino, que ahora son de diversa índole, de diversas áreas, e inclusive, usuarios que no tienen nociones sobre el tema de la computación que tienen la necesidad de aplicar las ventajas del uso de la computadora para auxiliarse en sus trabajos; ante este incremento y diversidad en el número de usuarios, el software tenía que modificar su actitud ante ellos de manera que facilitará su utilización a todos aquellos que pudieran utilizarlo.

De esta manera se busca implementar buenos elementos que faciliten la operación de los programas y presenten en forma rápida ayuda sobre su uso, la cantidad de elementos de interface y variantes de ellos que podemos encontrar es inmensa, sin embargo, existen algunas de ellas que han resaltado en forma notable debido a su sencilla utilización y fácil aprendizaje, las cuales han

seguido evolucionando de acuerdo a las nuevas tecnologías y dispositivos que van surgiendo como pueden ser el uso de dispositivos apuntadores; alrededor de 1983 surge el concepto de las interfaces gráficas (GUI), en los cuales fueron pioneros XEROX y OPEN LOOK, los cuales fueron adoptados posteriormente por casi todos los sistemas.

Mediante estos elementos de interface se logra la comunicación con el usuario, y se facilita la operación del sistema; debido a esta gran importancia, las empresas que se dedican a el desarrollo de software tratan de implementar de la mejor forma posible los elementos de mayor calidad y aceptación por parte del usuario en sus sistemas, además, de que tratan de mejorar las ya existentes y buscan crear algunas otras las cuales tengan un mayor impacto que sus antecesores; ante esta política, los elementos de interface han tenido un rápido crecimiento en los últimos años, y aun siguen creciendo en forma notable.

Con el crecimiento de las empresas desarrolladoras de software, este mercado creció rápidamente, y un usuario potencial puede elegir entre más de un software para satisfacer sus necesidades, es decir, ante una necesidad, se puede encontrar mas de un producto, los cuales resuelven su problema, por supuesto, un usuario cuidadoso y precavido seleccionaría aquel cuyo costo sea mas bajo, resuelva totalmente sus necesidades, sea mas fácil de utilizar, aprender y que obtenga resultados en forma rápida.

Con la gran competencia de software que se establece en el mercado, surge una lucha entre las compañías para ganarse a los usuarios, para lo cual se establece una batalla en diversos aspectos como son:

- Elementos de interface modernos y eficientes que faciliten el trabajo.

- Ayuda en línea sobre todas las funciones del sistema.

- Poderosas y modernas herramientas de software que resuelvan problemas concretos y realicen tareas en forma eficiente y rápida.

En donde los elementos de interface son uno de los puntos fundamentales para ganar la batalla en el cual el único fin es que la decisión de un usuario al comprar algún software en especial se incline hacia sus productos, por tal motivo, los diseñadores y programadores buscan estar a la cabeza en todos los aspectos anteriores, contribuyendo con ello en su desarrollo.

Hay varios aspectos que contribuyen a el desarrollo de los elementos de interface y a su futuro, entre estos se pueden mencionar :

- Diseñadores de software e investigadores están explorando:
  - Técnicas de selección de menús.
  - Uso de gráficos, animación y colores.
  - Manipulación directa de la pantalla.
  - Facilidad de utilizar el lenguaje natural.
  - Manejo de mensajes de error y prevención de ellos.
  - Formatos de Pantalla mas eficientes.
- Desarrolladores de Hardware ofrecen nuevas posibilidades como:
  - Monitores de alta resolución y mayor tamaño.
  - Tiempo de respuesta rápido en sus dispositivos.
  - Nuevos dispositivos apuntadores o de selección.
  - Voz como entrada y salida.
  - Grandes dispositivos de almacenamiento.

Sin embargo, no todos los elementos de interface tienen la misma aceptación por parte de los usuarios, y si no se utilizan en forma eficiente, causarán ansiedad y desesperación, entre los enemigos de los elementos de interface se encuentran :

- Comandos inconsistentes.
- Confusas secuencias de operación.
- Caóticos formatos de desplegado.
- Terminología inconsistente.

- Procedimientos complejos.
- Amenazantes mensajes de error.

Existen algunos factores u objetivos que se deben de tomar en cuanta al diseñar o evaluar algún elemento de interface, entre los principales podemos mencionar:

- Tiempo de aprendizaje.
- Velocidad de respuesta.
- Indice de errores por parte del usuario.
- Satisfacción subjetiva.
- Tiempo de retención (cuanto tiempo mantiene el usuario el conocimiento sobre su uso).

Los elementos de interface con el usuario tienden a mejorar día con día en beneficio de los millones de personas que utilizan diariamente la computadora como una herramienta básica en su área de trabajo; con esto, los sistemas se han hecho mas accesibles hacia ellos facilitándole en gran medida su uso.

1.2.3 ELEMENTOS DE INTERFACE MAS COMUNES.

Existen una gran diversidad de formas para establecer una comunicación entre un sistema y un usuario, sin embargo, algunas de ellas han destacado en forma notable hasta llegar al grado de ser herramientas muy utilizadas en el desarrollo de software profesional y competitivo en el mercado, estos elementos de interface pueden ser basadas en modo texto o en modo gráfico, dependiendo de la aplicación y entorno del programa; las características principales de ambos modos son :

- INTERFACES BASADAS EN TEXTO:

Su característica principal radica en la utilización del monitor completamente en modo texto, de manera que no se pueden representar líneas ni dibujos, pero se puede hacer uso

de cualquier carácter contenido en el código ASCII, ya sean caracteres numéricos, alfabéticos o especiales; algunos de estos caracteres especiales se pueden utilizar ya sea para formar algún recuadro, o para colocar en la pantalla algún símbolo que necesitemos para algún caso, la principal limitación consiste en el número de caracteres que se pueden desplegar en un mismo tiempo en la pantalla, los cuales son:

- Puede representar un total de 80 caracteres por renglón y un total de 25 renglones, es decir, un total de 2,000 caracteres en una sola pantalla

- Con un tamaño de 40 caracteres por renglón y un total de 25 renglones, para un total de 1,000 caracteres.

Esta es la principal desventaja de este tipo de interface, teniéndose que adaptarse a esta limitación y realizar un mejor diseño posible para no saturarla rápidamente. Su principal ventaja radica en que la velocidad de desplegado es mucho mas rápida que en modo gráfico, teniéndose mejores resultados en cuanto a velocidad de respuesta.

- INTERFACES GRÁFICAS:

Este medio es hoy en día mas comúnmente usado debido a que la utilización de una pantalla gráfica con una buena resolución incrementa el número de caracteres que podemos desplegar, teniendo la posibilidad de desplegar mas datos, mas ventanas, mas menús, o en general mas elementos de interface; la combinación de estos elementos con gráficos facilitan el aprendizaje, ya que el usuario lo puede asociar con elementos reales de uso común, además, todos los elementos que se presentan en modo texto pueden representarse en modo gráfico con una mejora notable en la presentación y en su impacto hacia el usuario.

Su principal desventaja radica en que la velocidad de desplegado es menor que en modo texto, pero, hoy en día la

velocidad de los dispositivos de video y el equipo de cómputo en general ha aumentado en forma notable, así como la cantidad de memoria destinada para el video, con esto, en equipos modernos, esta desventaja se disminuye considerablemente.

Entre los principales elementos de interface que pueden encontrarse y que se pueden representar tanto en modo texto como en modo gráfico a excepción de los iconos, podemos mencionar los siguientes:

- LINEA DE COMANDOS :

En este tipo de interface el usuario comunica al sistema la acción que debe de ejecutar por medio de una serie de comandos que ya tiene previamente establecido el programa. Un ejemplo de este tipo de interface lo presenta el programa **COMMAND.COM**, el cual tiene integrados una serie de comandos internos como son: **COPY, DIR, CD, MD, RD,** etc.; y la forma de acceder a ellos es teclear el comando que se desea utilizar con sus parámetros respectivos. La principal desventaja de este elemento de interface es que el usuario esencialmente debe de aprender un nuevo lenguaje mediante la sintaxis de los comandos, por supuesto que en manos de un usuario experto, su utilización es un elemento muy poderoso, rápido y eficaz, sin embargo, en manos de un usuario inexperto o novato es muy difícil de utilizar y lleva tiempo el aprendizaje sobre los comandos y su sintaxis; un ejemplo puede representarse con un comando del sistema operativo:

```
C:> COPY TEXTO1.TXT TEXTO2.TXT
```

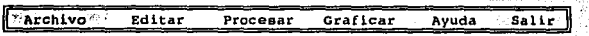
- MENÚS :

Este elemento de interface permite al usuario seleccionar comandos, acciones, o en general alguna opción de una lista de ellas mostrada en la pantalla ya sea en forma horizontal o en forma vertical, además, las opciones que muestra dan acceso a una serie de aplicaciones

funcionales o a otros elementos de interface; los tipos mas comunes son: el menú vertical, horizontal, matricial y el menú en cascada, el cual es una combinación de los dos primeros; se pueden identificar fácilmente por su característica principal de presentar una serie de opciones para que el usuario seleccione alguna de ellas y posteriormente ejecutará alguna acción.

- **Menú Horizontal:**

  Presenta las opciones disponibles en forma horizontal, el cual generalmente se coloca en la parte superior de la pantalla, y representa un menú principal; debido a la poca cantidad de caracteres que se puede desplegar en un renglón (80 o 40 caracteres), estas opciones desplegadas tiene que representarse por una mínima expresión que la represente en forma concisa.



- **Menú Vertical:**

  Presenta las opciones a través de una lista en forma vertical en cualquier parte de la pantalla, su principal ventaja sobre el menú horizontal radica en que utiliza los renglones para colocar las opciones disponibles teniendo por tanto una mayor área de desplegado.

- **Menú Matricial:**

  Su característica principal consiste en el desplegado de las opciones aprovechando tanto los renglones como las columnas de la pantalla, de

manera que se puedan desplegar un número mayor de ellas usando el espacio y dimensiones de la pantalla, generalmente es usado para colocar una serie de módulos o programas que se pueden ejecutar y se desea desplegar todos ellos a la vez para que el usuario seleccione solamente uno de ellos.

COPIAR ARCHIVOS	CREAR SUBDIRECTORIO
COPIAR DISCO	BORRAR SUBDIRECTORIO
COPIA DE SEGURIDAD	CAMBIAR SUBDIRECTORIO
FORMATEAR	COMPARAR DISCOS
RESTAURAR	COMPARAR ARCHIVOS
COMPILADOR BASIC	BORRAR ARCHIVOS
COMPILADOR C	CAMBIAR ATRIBUTOS
COMPILADOR FORTRAN	ANTI-VIRUS

- **Menús En cascada:**

  Es una combinación de los menús vertical y horizontal, en el cual las opciones principales se representan en el Menú Horizontal, y las opciones secundarias se colocan en los menús verticales.

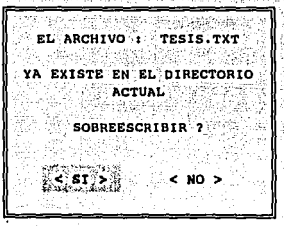
- AYUDA EN LINEA :

Se caracteriza por presentar una pantalla con la información necesaria sobre el uso, características funcionamiento y resultados del comando o módulo que el usuario haya elegido mediante la colocación del cursor, es decir, no es necesario indicar sobre que comando u opción deseamos la ayuda, sino que dependiendo del lugar donde tengamos ubicado el cursor, será desplegada una

ventana con la información relacionada con esta opción, esto con el fin de facilitar su uso y ahorrar tiempo para la solución de los problemas que pudiera tener en el entendimiento y aplicación de la opción.

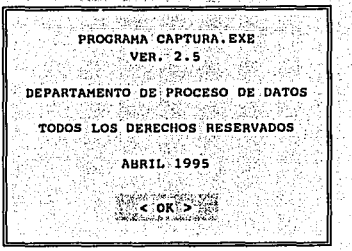
- MENSAJES DE DECISIÓN :

Presenta en la pantalla un mensaje, el cual consiste en una pregunta referente a alguna acción que debe de ejecutar el sistema, pero en donde necesita la autorización del usuario; generalmente estos mensajes desaparecen una vez que el usuario ha respondido al sistema restaurando la parte de la pantalla en la cual se había desplegado.



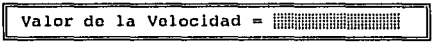
- MENSAJES DE INFORMACIÓN O ALERTA :

Presenta un recuadro o caja que contiene un aviso para el usuario el cual tiene la finalidad de advertirle sobre alguna falla del sistema, de escritura, de lectura, de incoherencia entre los datos, o simplemente como información de la tarea que esta ejecutando, de manera que el usuario pueda enterarse de la función que esta realizando el sistema en el momento.



- CAMPOS DE TEXTO O NUMÉRICOS :

Esta es la forma de entrada de mayor uso en los programas, debido a que por lo general siempre es necesario la introducción de datos al sistema, ya sean datos numéricos, alfabéticos o alfanuméricos, y el campo de texto o numéricos es el medio que se utiliza para su lectura; las principales características de este elemento de interface radica en la colocación de un campo de tamaño determinado con sus correspondientes atributos de color de fondo y de los caracteres, en donde el usuario podrá introducir los datos al sistema, el campo de texto no permitirá que el usuario exceda el tamaño destinado para esa variable, por ejemplo :



- CAMPO DE SELECCIÓN MÚLTIPLE

Se caracteriza por presentar a una variable que puede tener mas de dos valores, pero con un límite, de manera que el usuario puede fijar la variable en uno de sus posibles valores, los cuales se colocan en un solo campo y conforme el usuario va cambiando de posición desaparece el valor activo y aparece el siguiente, este tipo de elemento de interface tiene la desventaja de necesitar un

elemento apuntador para su funcionamiento, la forma de intercambiar entre los valores que se pueden tener es mediante las flechas o símbolos que se colocan a los extremos del campo indicando, la opción anterior y la opción posterior respectivamente; por ejemplo :

Velocidad | ▲ | LENTA | ▼ |

- LISTA DE ARCHIVOS O DIRECTORIO :

Se caracteriza por mostrar en una ventana los archivos que existen en un determinado directorio; esto con el fin de que el usuario seleccione alguno para realizar una tarea con él, como puede ser : editarlo, copiarlo, borrarlo o cargarlo como archivo de trabajo. Esto facilita la selección de archivos a el usuario, de manera que pueda realizarlo de forma sencilla, rápida y eficiente, algunas de sus características son :

- Desplegado de las unidades de discos existentes, tanto flexibles como rígidas, para que el usuario pueda seleccionar la unidad que desea.

- Desplegado en fondo inverso del archivo en el cual tiene posicionado el cursor por parte del usuario.

- La posibilidad de seleccionar el archivo tecleando solamente el nombre, sin la necesidad de tener que realizar su búsqueda.

- Utilizar el desplazamiento (SCROLL) de archivos hacia arriba y abajo de la ventana destinada para su desplegado, ya que generalmente el número de archivos es variable y en la mayoría de los casos mayores que el espacio destinado para presentarlos.

Un ejemplo de este tipo de interface es:

ARCHIVO : TEXTO1.TXT	
.. <DIR>	REPORTE.REP
TEXTO1.TXT	PARTE1.DOC
TEXTO2.TXT	PARTE2.DOC
TEXTO3.TXT	PARTE3.DOC
GRAFICO1.GFX	TESIS1.DOC
COMMAND.COM	TESIS2.DOC
AUTOEXEC.BAT	TECLADO.BMP

- BOTONES

Se llaman de así debido a su analogía que presentan con los llamados "SWITCH" o "BOTONES", los cuales tienen la característica de tener 1 estado de 2 posibles como pueden ser "encendido" o "apagado"; se utiliza para representar a una variable que tenga esta característica, de manera que el usuario puede cambiar el valor de uno a otro dependiendo de su decisión; por ejemplo una variable que defina si la computadora tiene coprocesador o no; en modo gráfico puede representarse con una forma real para que el usuario lo pueda identificar y asociar fácilmente; su variedad de forma que puede adoptar es muy grande.

- FORMAS DE ENTRADA Y SALIDA :

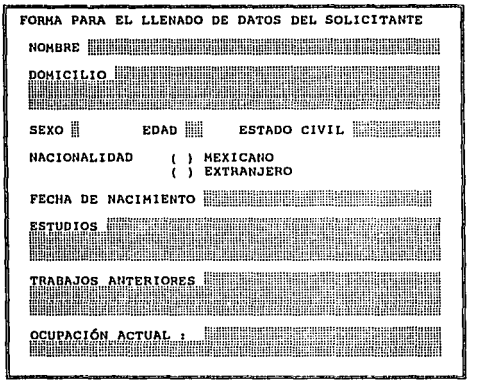
Se le llama así por su analogía que presenta con las formas de datos o documentos que puede encontrarse en el trabajo diario, como puede ser :

- Forma de solicitud de empleo
- Forma de Inscripción
- Forma de exámenes extraordinarios
- Forma de pago, etc.

Su principal característica consiste en que su contenido en cuanto a el tipo de variables siempre es el mismo, aunque el valor de las variables cambia de acuerdo a cada persona o usuario; utiliza para su formación elementos de interface sencillos como son :

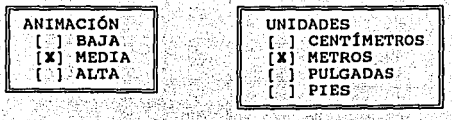
- Campos de texto
- Botones
- Campos de selección múltiple

En un mismo sistema pueden existir varias FORMAS, dependiendo de las necesidades; un ejemplo es :



- LISTA DE CONTROLES O BOTONES:

Este elemento de interface tiene semejanza con los menús y con los campos de selección múltiple, se caracteriza por presentar una serie de valores que puede tener una variable en forma de una lista, la diferencia con los menús radica en que no dan acceso a aplicaciones funcionales, solamente fijan una variable a un determinado valor, y muestran estos posibles valores en la pantalla en forma de una lista, de manera que puedan apreciarse todos a la vez, y el usuario pueda seleccionar solamente uno de los que se muestran en la pantalla, por ejemplo :



- DESLIZADORES (SLIDER'S)

Su función principal consiste en fijar un valor entre un mínimo y un máximo, el usuario puede modificar su valor inicial mediante el arrastre de un botón o interruptor hacia alguno de los extremos del deslizador, su forma puede ser horizontal o vertical dependiendo de las necesidades, un ejemplo es el siguiente:

- ICONOS :

Son una representación gráfica de un comando, módulo o instrucción que puede ejecutar el sistema. Esta representación tiene un tamaño pequeño, de manera que pueden representarse varios de ellos en una pequeña parte de la pantalla, la ventaja mas importante de este elemento de interface es que el usuario puede asociar mas rápidamente un dibujo de un objeto real con una acción del sistema, de manera que su aprendizaje y utilización es mas rápida y sencilla.

Tanto en las interfaces en modo texto como gráficas puede ser utilizado un dispositivo apuntador, como el ratón, para facilitar el modo de selección de todos los tipos de interface, incluso, hoy en día todos los programas comerciales incluyen el uso de este dispositivo, por lo cual se considera indispensable para una selección rápida de las opciones disponibles.

1.2.4 LA RE-USABILIDAD EN LOS ELEMENTOS DE INTERFACE.

En el pasado, los fabricantes de software ponían muy poca atención hacia los elementos de interface, porque resultaba mucho mas económico; además que el Hardware imponía muchas limitaciones a los fabricantes; hoy en día, los usuarios son más exigentes y en conjunto con la gran competencia existente, los fabricantes tratan de que su producto le otorgue la mayoría de las facilidades hacia los usuarios; esto ha influido en forma considerables en el costo del software y en su tamaño.

Se estima que en la década de los 50's 90% del costo de un sistema de cómputo era en Hardware y 10% era para el software; hoy en día, por el contrario, hay software cuyo costo iguala, e incluso llega a superar al costo del hardware.

De igual forma, las líneas de código utilizadas para los elementos de interface dentro de un programa han ido creciendo en forma considerable en los últimos años; en la década de los 50's, muy pocas líneas del código correspondían a la interface con el usuario; para 1984, se estimaba que los elementos de interface requerían del 30% al 35% de las líneas del código total; una estimacion mas reciente lo coloca entre el 47% al 60%.

Todo esto influye notablemente en el costo del software; según estimaciones, cerca del 29% del costo del software desarrollado corresponde en los elementos de interface para equipos de alta resolución como estaciones de trabajo; y cerca del 17% si el

software no esta orientado a equipos de muy alta resolución[5].

Ante todas estas situaciones, y considerando la importancia de los elementos de interface, así como en su contribución en el costo del software; se ha puesto mas énfasis en la re-usabilidad de los elementos de interface; es decir, la finalidad es tener alguna libreria con estos elementos ya generados, para ahorrar trabajo en su diseño y en el costo final del programa; no se trata de crear elementos de interface para cada programa.

Además, se han establecido una serie de reglas, para que exista una consistencia entre todos los elementos de interface, tratando con ello de unificar criterios en el desarrollo de las interfaces, y evitar la proliferacion de interfaces inconsistentes, las cuales puedan confundir al usuario.

1.3.- EL LENGUAJE C EN EL DESARROLLO DE LAS RUTINAS DE LOS ELEMENTOS DE INTERFACE.

1.3.1.- CARACTERÍSTICAS DEL LENGUAJE C

C es un lenguaje de programación catalogado como de nivel medio, el cual fue desarrollado inicialmente para la programación de sistemas como pueden ser :

- Editores
- Interpretes
- Compiladores
- Sistemas Operativos
- Paquetes Integrales.

Sin embargo, su popularidad creció rápidamente y comenzó a

[5] Ray E. Eberts. User interface Design. Editorial Prentice Hall. Primera Edición 1994.

utilizarse para cualquier tipo de tarea que se quisiera programar, y hoy en día se ha convertido en un lenguaje de uso general; sus principales características son:

- ES UN LENGUAJE DE NIVEL MEDIO

C esta clasificado como un lenguaje de nivel medio, en donde la clasificación total es :

LENGUAJES	EJEMPLOS
ALTO NIVEL	BASIC, PASCAL, FORTRAN, MODULA-2
NIVEL MEDIO	C, FORTH, MACRO-ENSAMBLADOR
BAJO NIVEL	ENSAMBLADOR

TABLA 1.3

La clasificación anterior, no se basa en jerarquías o potencialidad de los compiladores, sino que esta agrupación se basa en determinadas características importantes que marcan una diferencia sustancial entre todos ellos. Un lenguaje de bajo nivel permite la manipulación de bits, bytes y direcciones mediante un lenguaje simbólico difícil de entender, además, permite el uso de los registros del microprocesador y el uso de interrupciones a través de las funciones del **BIOS** y **DOS**. Un lenguaje de **ALTO NIVEL** maneja los conceptos de tipos de datos, y además tiene incluidas ciertas funciones que permite al usuario manejar los recursos de la computadora.

El hecho de que C este clasificado como un lenguaje de nivel medio no implica que sea menos potente que los de ALTO NIVEL, ni tampoco que su tipo de programación sea parecida al ensamblador; la clasificación de C implica que puede manejar elementos de lenguaje de alto nivel y elementos de bajo nivel como pueden ser manipulación de bits y direcciones de las variables.

El lenguaje C no tiene incluida ninguna instrucción de entrada y salida de datos, funciones matemáticas, funciones de acceso a ficheros, funciones relacionadas con gráficos, etc.; es decir, el lenguaje de programación C carece de estas funciones integradas propiamente en el compilador; bajo esta panoramica. la programación con el lenguaje C no muestra ninguna ventaja importante sobre los lenguajes de alto nivel, al contrario, pareceria un lenguaje rudimentario y dificil; sin embargo, el lenguaje C suple estas deficiencias con la **"LIBRERIA ESTANDARD"**, la cual contiene una serie de funciones que realizan una diversidad de tareas que no realiza el compilador propiamente, esta libreria estandard debe de ser la misma para todos los compiladores de C, de manera que se garantice la unificación de ellos.

Además de la libreria estandard del lenguaje C, los fabricantes de los compiladores puede anexar adicionalmente otra serie de funciones nuevas, especiales o propias, esto con el fin de dar una mayor potencialidad a su compilador y ofrecer a los programadores un mayor número de herramientas para realizar su tarea; como se observa el lenguaje C puede realizar las funciones que nos dan los lenguajes de alto nivel con la ayuda de las librerías que nos proporciona, pero con la ventaja de utilizar elementos de un lenguaje de bajo nivel.

- C BASA SU FUNCIONAMIENTO EN EL USO DE FUNCIONES.

La programación en lenguaje C es a base de funciones, las cuales forman el concepto de construcción con módulos en este lenguaje; un programa en C esta formado por una serie de funciones, de las cuales una es la primera en ejecutarse y por lo tanto siempre debe incluirse, además de que representa el programa principal esta función es *main()* y representa el punto de inicio en la ejecución del programa; en esta función, al igual que las otras pueden incluirse sentencias, bucles, ciclos o llamadas a otras funciones. Una función es una subrutina, la cual puede realizar una o más tareas, puede contener sentencias de programación en C y llamadas a otras funciones; la forma fundamental de una función en C es la

siguiente :

```
[void] Nombre_de_la_funcion (Lista_de_argumentos)
lista_de_argumentos, declaracion
 {     /* INICIO DE LA FUNCION */
    ---     /* CUERPO DE LA FUNCION */
    ---     /* SENTENCIAS EN C Y/O LLAMADAS A */
    ---     /* OTRAS FUNCIONES  */
  [return();]
 }      /* FIN DE LA FUNCION  */
```

- la palabra reservada **void** encerrada con corchetes solamente se coloca si la función en cuestión no retorna ningún valor al terminar de ejecutarse, en este caso, no deberá de colocarse la palabra reservada **return**, en caso contrario, si la función en cuestión retornara algún valor, no deberá de colocarse la palabra reservada **void**.

- El nombre de la función **NO** debe de ser una palabra reservada o igual a otra función declarada anteriormente.

- La lista de argumentos debe de ser solamente el nombre de las variables que se están pasando a la función, el tipo de ella se declara en la declaración de la lista_de_argumentos como se muestra a continuación :

```
suma(x1,x2)
int x1,x2;
 {
   return(x1+x2);
 }
```

En el caso anterior podemos identificar cada una de las partes de la función

Nombre_de_la_función = *suma*
lista_de_argumentos = *x1,x2*
lista_de_argumentos,declaración = int x1,x2;
valor retornado = x1+x2

Algunos compiladores soportan la declaración de la lista de argumentos inmediatamente después del nombre de la función como se muestra a continuación:

```
suma(int x1, int x2)
 {
  return(x1+x2);
 }
```

- La lista de argumentos que puede recibir la función varía dependiendo de la función que realice; en caso de que no se necesiten argumentos deberá de incluirse los paréntesis, para indicar que se trata de una función.

- los caracteres "/*" y "*/" indican al compilador inicio y fin de algún comentario respectivamente; algunos compiladores (como es el caso de turbo C) reconocen el carácter "//" como comentario por línea, se llama de esta manera debido a que marca el inicio de algún comentario, el final del mismo es tomado al final de la línea.

- C ES UN LENGUAJE ESTRUCTURADO.

C ha llegado a considerarse como un lenguaje estructurado, debido a que la característica básica de estos es la programación a través de módulos o bloques de instrucciones, de manera que un programa muy grande pueda dividirse en pequeños partes, además de la utilización de estructuras de control e iterativas.

Un lenguaje estructurado soporta el concepto de subrutinas con variables locales a través de procedimientos y funciones; además, soporta instrucciones iterativas o bucles mediante sentencias como **for**, **while**, **repeat**, etc. C soporta todas las características anteriores, además tiene sus propias sentencias iterativas y de decisión de manera que en este aspecto C no tiene ningún problema.

Existen algunos autores que consideran al lenguaje C como un lenguaje **SENCILLAMENTE ESTRUCTURADO**, argumentando que un lenguaje **TOTALMENTE ESTRUCTURADO** soporta el concepto de procedimientos, los cuales son considerados como un pequeño programa interno, y por lo tanto puede tener incluidos funciones y procedimientos propios, y esto contrasta con la incapacidad que tiene el lenguaje C de integrar funciones dentro de funciones.

- C ES PORTABLE

Una de las principales ventajas que tiene el lenguaje C sobre los demás es su portabilidad, esto significa que si se genera un programa en un equipo **APPLE**, este puede ser llevado fácilmente a un equipo **IBM PC** o compatible; es decir, puede trasladarse de un tipo de sistema a otro con un mínimo o ninguna modificación sobre el programa fuente, esto se debe a que las palabras reservadas que tienen los compiladores de C son las mismas, así como sus reglas de sintaxis, y su libreria standard.

Debido a la popularidad del Lenguaje C, pueden encontrarse compiladores en todo tipo de sistema que se desee utilizar y bastará con compilar un programa en otro sistema para que este sea trasladado, lo cual se denomina PORTABILIDAD, esta característica es la mas importante de todas y una de la mas potentes de C sobre los otros lenguajes, incluso los de alto nivel, algunos de los cuales se han generalizado y existen muchas versiones de ellos, cada uno con sus características y reglas propias, lo cual hace muy difícil el traslado de un programa creado con un tipo de compilador con otro.

La portabilidad de C se basa en el número de instrucciones de control o palabras clave que tiene, las cuales son la base y las mismas en la mayoría de los compiladores de C, la diferencia fundamental radica en las librerías de funciones que pueda proporcionar y en el medio ambiente de programación que pueda establecer. En caso de no encontrar una función al trasladar un programa de un sistema a otro, bastara con

implementarla en el nuevo compilador con lo cual se resolverá el problema.

1.3.2.- C EN EL DESARROLLO DE SOFTWARE PROFESIONAL.

El lenguaje C se utilizo inicialmente para la programación de sistemas, la cual se refiere a la clase de programas que son parte o interfieren directamente con el sistema operativo del computador, así como en su soporte, por ejemplo :

- Sistema Operativos
- Interpretes
- Compiladores
- Bases de Datos
- Editores

Debido a su gran popularidad, muchos programadores, lo utilizan para realizar cualquier tarea debido a su eficiencia. Además, los compiladores de C producen un código muy compacto y de rápida ejecución, por esta razón es bastante utilizado, además que pone pocas restricciones a la programación y permite crear bibliotecas con funciones propias que se ajusten a su personalidad, tarea y necesidad.

Muchos autores consideran al lenguaje C como un lenguaje para programadores debido a que C da al programador lo que quiere mediante una serie de alternativas y ventajas como son:

- Impone pocas restricciones en su uso.
- Ofrece estructuras de bloque
- Manejo de funciones independientes
- Un compacto número de palabras clave o reservadas
- Rapidez en la ejecución de su código
- Permite una fácil manipulación de bits, bytes y direcciones.
- La portabilidad de su código.

C ha sido muy utilizado en el desarrollo de software debido a

que combina la eficiencia del código ensamblador con la estructuración de un lenguaje de alto nivel como PASCAL; por este motivo, ha llegado a convertirse en un lenguaje de programación de propósito general.

Una vez familiarizado con el código del lenguaje C, éste es muy legible y se puede seguir el flujo del control y la lógica del programa fácilmente, de esta manera, la detección de errores se facilita; además de permitir el trabajo entre grupos de personas o programadores mediante la creación de funciones independientes para posteriormente integrarlas en un sólo sistema; debido a todos los aspectos anteriores, el lenguaje C se ha utilizado para la creación de software profesional y competitivo.

Sin embargo, habrá que tomar en consideración que el lenguaje no hace al sistema, es decir, no basta con utilizar el lenguaje de programación C para obtener buenos programas o con buena calidad, tendrá que tomarse en cuenta estilos y métodos de programación, así como la conceptualización del problema y métodos de solución, ya que si se tienen muy en cuenta estas últimas características se pueden realizar buenos programas en cualquier tipo de lenguaje; sin embargo, tomar en cuenta cual lenguaje de programación es el mas adecuado para resolver el problema ayuda bastante en su solución, ya que se pueden tener ciertas características que nos ofrezca un lenguaje en especial y que nos sean de utilidad para resolver el problema.

1.3.3 VENTAJAS DEL LENGUAJE C PARA EL DESARROLLO DE LAS RUTINAS EN LOS DIFERENTES ELEMENTOS DE INTERFACE.

El lenguaje de programación C fue seleccionado para la elaboración de las rutinas de elementos de interface debido a que ofrece algunas ventajas que nos son de utilidad en el desarrollo de las rutinas que elaboraremos en el transcurso de esta tesis, estas ventajas nos permitirán crear rutinas eficientes, rápidas y sobre

todo agruparlas en una libreria especial creada especialmente para ellas, estas ventajas son :

- Crea un código ejecutable, compacto y de rápida ejecución a semejanza del lenguaje ensamblador, lo cual nos ayudara en la creación de elementos de interface rápidos , así como en la generación de programas ejecutables que no ocupen mucho espacio en disco, estos son elementos importantes en el desarrollo de cualquier programa y en general una de sus características más importantes, las cuales ya se han señalado anteriormente.

- Ofrece la ventaja de poder hacer referencia a las direcciones de las variables, y de la memoria; este aspecto nos será muy útil al trabajar directamente con la memoria del computador con el fin de obtener con esto una mayor rapidez en su velocidad de respuesta.

- El uso de funciones en el lenguaje C nos obliga a fraccionar los programas en pequeños módulos o funciones que realicen una tarea especifica y posteriormente ser llamados por una función la cual se encargara de generar el elemento de interface y retornar un valor relacionado con la decisión del usuario; esto además tiene que ver con el desarrollo de programas estructurados y modulares, que es una de las cuestiones que se tratan de cuidar y seguir en las rutinas que se crearan posteriormente.

- El lenguaje permite la agrupación de funciones en librerías propias, lo cual nos será de utilidad al terminar de generar todas las rutinas que se desean implementar en esta tesis; ya que estas se podrán agrupar en una sola libreria, y posteriormente podremos utilizarlas para cualquier programa que se quiera realizar, incluyendo solamente el uso de esta libreria y haciendo referencia a las funciones con sus argumentos o parámetros respectivos, lo cual nos ahorrara bastante trabajo y nos será de mucha utilidad permitiendonos la creación de programas con un mejor nivel de calidad.

- La portabilidad del lenguaje es una de las principales características que nos ha hecho inclinar en favor de la utilización de este lenguaje, ya que al realizar las rutinas especialmente en una marca de compilador y sistema, estas mismas rutinas se podrán implementar en otra marca de compilador y en otro sistema con un mínimo de cambios, favoreciendo esto la utilización de las rutinas creadas en diversos compiladores y medios.

- La experiencia de haber trabajado con el lenguaje de programación C en la creación de programas de diversa índole, lo cual nos ayudará a generar mejores funciones, programas y en general mejores resultados, debido a que no se tratará de generar elementos de interface simples debido al desconocimiento del lenguaje; al contrario, el objetivo es crear elementos de buena calidad conservando todas sus características principales.

CAPITULO II

CONOCIMIENTOS BÁSICOS.

2.1 OBJETIVOS DEL CAPITULO

Este capitulo tiene como objetivos fundamentales explicar al lector una serie de conceptos indispensables, los cuales utilizaremos a lo largo de los capítulos posteriores, estos temas incluyen:

a) El manejo eficiente del teclado.

b) El uso de las interrupciones.

c) Los conceptos básicos sobre los adaptadores de video.

d) El uso y manejo del ratón(mouse) en nuestros programas.

Con el fin de dar mayor fluidez a la lectura, no se incluirán ejemplos sobre los temas que se incluyen en esta sección, ya que la aplicación de estos conceptos podrá apreciarse en los capítulos posteriores en la parte de generación de rutinas de los elementos de interface, tratando de agilizar con esto la lectura.

2.2 EL ROM BIOS Y SU USO.

La arquitectura de una computadora como el IBM PC esta formada por una serie de capas funcionales, en donde la capa o nivel mas bajo es el hardware y el mas alto es el programa de aplicación, el cual establece la interface con el usuario. Entre el hardware y el programa de aplicación existe el software del

sistema, el **IBM** incluye en su arquitectura una capa de software a bajo nivel entre el hardware y el software del sistema llamado **BIOS**[6] , el cual provee servicios primitivos de entrada y salida útiles para la programación a través del manejo de interrupciones del hardware.

Para asegurar compatibilidad entre el software en futuras generaciones de computadoras, el BIOS ha mantenido un enlace o vínculo con sus antecesores, de esta forma, los microprocesadores **INTEL 80x86 (80286, 80386, 80486)** pueden ejecutar el conjunto de instrucciones del 8088 y del 8086. Las computadoras basadas en el microprocesador 80x86 son principalmente controladas a través del uso de interrupciones, las cuales pueden ser generadas por el microprocesador, por el hardware o por el software; cuando ocurre una interrupción el control es transferido a una rutina para su manejo, una vez terminada esta rutina, son retornados ciertos valores en los registros del microprocesador, y el control regresa al lugar donde fue llamada la interrupción; cada una de ellas tiene asignado un número único asociado, el cual esta comprendido entre un rango de *00H* a *FFH*; por convención ciertos rangos de números de interrupciones son reservados para usos especiales; por ejemplo, los números entre *20H-3FH* son reservados para el DOS.

Como se menciono anteriormente, la interrupción puede ser invocada o llamada por el procesador, hardware o software, la forma en que cada uno de estos elementos utiliza las interrupciones es diferente como se indica a continuación:

- **PROCESADOR** : También son llamadas interrupciones Lógicas (INT *00H - 04H* son reservadas para el procesador), y son invocadas cuando resulta un termino inusual del programa, como puede ser una división entre cero.

- **HARDWARE** : Son invocadas por algún dispositivo

---

6 De las siglas en Inglés **B**asic **I**nput **O**utput **S**ystem (Sistema básico de entrada/salida).

periférico que necesita de su uso, por ejemplo el teclado, el cual llama una interrupción cuando se presiona una tecla, los numeros reservadas para el hardware son ***08H-0FH*** y ***70H-77H***, también son conocidas como rutinas de servicios de interrupciones (ISR Interrupt Services Routines).

- **SOFTWARE** : Estas son invocadas mediante la instrucción INT del 80x86 y también son llamadas Rutinas de Servicios de Dispositivos (DSR Device service Routines).

Las interrupciones que más nos interesan son aquellas que son invocas mediante software (DSR), debido a que las utilizaremos en el desarrollo de nuestras rutinas.

En las Interrupciones de software, un parámetro es pasado como argumento en el registro **AH**, la mayoría de las interrupciones del BIOS contienen varias funciones; para usar alguna de ellas se colocara el parámetro respectivo en los registros **AL** o **BL**. La llamada a una función del BIOS incluye el paso de algunos argumentos a través de los registros del microprocesador, y estas a su vez retornan información al programa mediante los mismos registros; aquellos que no son utilizados al realizar la llamada a la función mantienen sus valores anteriores.

A continuación mostraremos una tabla con algunas de las interrupciones del BIOS, así como sus funciones y argumentos[7]:

[7] Para Una mayor información sobre las interrupciones puede recurrirse al libro SYSTEM BIOS for IBM, PC's, compatibles and Eisa Computers. Phoenix Technologies LTD. Editorial Addison-Wesley. Segunda Edición 1991. o el libro titulado Guía del programador para el IBM PC y PS/2. Peter Norton & Richard Wilton. Ed. Microsoft Press.

SERVICIO DE IMPRIMIR PANTALLA		El servicio del BIOS de imprimir pantalla realiza la función de imprimir el contenido del video actual de la pantalla a la impresora.
INT.	PARÁMETRO	DESCRIPCIÓN
05 H	NINGUNO	IMPRIME LA PANTALLA ACTUAL DESPLEGADA

TABLA 2.1

SERVICIO DE VIDEO		El servicio del BIOS de Vídeo provee soporte de entrada y salida para los adaptadores de video, ya sea en modo texto o gráfico.
INT.	PARÁMETRO	DESCRIPCIÓN
10 H	AH = 00H	Fija el modo de vídeo
	AH = 01H	Fija el tipo de cursor
	AH = 02H	Fija la posición del cursor
	AH = 03H	Lee la posición actual del cursor
	AH = 04H	Lee la posición de la pluma luminosa
	AH = 05H	Selecciona nueva página de video
	AH = 06H	Desplazar la ventana hacia arriba
	AH = 07H	Desplazar la ventana hacia abajo
	AH = 08H	Leer carácter y atributo
	AH = 09H	Escribir carácter y atributo
	AH = 0AH	Escribir carácter en el cursor
	AH = 0BH	Fijar paleta de color
	AH = 0CH	Escribir pixel para gráficos
	AH = 0DH	Leer pixel para gráficos
	AH = 0EH	Escribir texto en modo de teletipo
	AH = 0FH	Retorna el estado del video
	AH = 13H	Escribe una cadena

TABLA 2.2

Cada una de estas funciones puede necesitar de una serie de parámetros adicionales para realizar su tarea, el tipo de parámetros dependerá de la función en cuestión por ejemplo:

- La función 02H de la interrupción *10H* : la cual realiza la tarea de fijar la posición del cursor en la pantalla, necesita de los siguientes parámetros los cuales son pasados en los registros del microprocesador :

  BH = Número de página
  DH = Renglón (0 es la esquina superior izquierda)
  DL = Columna (0 es la columna mas a la izquierda)

2.3 EL TECLADO COMO DISPOSITIVO DE ENTRADA.

El teclado es el instrumento o dispositivo principal mediante el cual podemos comunicar a la computadora los comandos, funciones y en general las instrucciones que necesita para el proceso de nuestra información; en una primera instancia podemos definir el teclado como el dispositivo que permita al usuario interactuar con la computadora; el propósito principal de este inciso es explicar la forma en que podemos obtener un mayor provecho de este dispositivo en nuestros programas, el tener mayor control de él y poder obtener una lectura de todas las teclas que la conforman.

Para nuestros propósitos, consideraremos al teclado como el elemento principal con el cual podemos obtener la decisión del usuario sobre una serie de alternativas que le presenta el elemento de interface que tiene en ese momento en la pantalla, esta serie de alternativas que puede seleccionar el usuario, puede contener la utilización de una gama de teclas de diversa índole, como pueden ser:

- Las teclas de flechas para el movimiento del cursor
- Las teclas de Función
- El teclado numérico

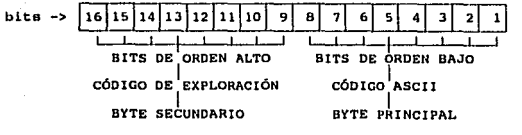
- El teclado alfabético
- Algunas otras teclas especiales que pueden ser presionadas en cualquier momento como pueden ser : INS, SUPR, INICIO, FIN, etc.

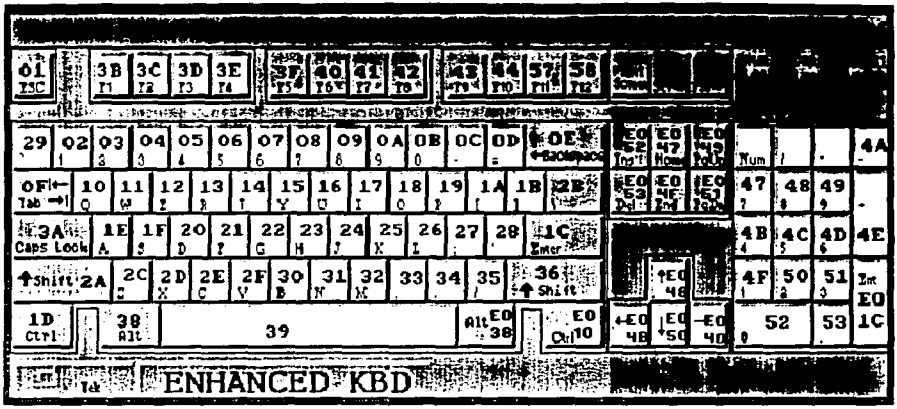
Cada una de las teclas tiene asociado un código llamado **CÓDIGO DE EXPLORACIÓN (SCAN CODE)**, el cual envía el microprocesador del teclado al computador cuando detecta que es presionada una tecla; este código es único para cada una de ellas; en la figura 2.1 puede observarse la disposición de las teclas y su respectivo código de exploración, de esta forma, podemos detectar e identificar a todas las teclas que forman parte del teclado, incluyendo a:

- Las teclas de Flechas
- Las tecla de Función
- Las teclas INS, SUPR, INICIO, FIN, REG. PAG., AV. PAG, etc.

Cuando se presiona alguna tecla, el teclado reconoce esta acción y notifica al computador, esto lo hace en la forma de una interrupción de hardware, la cual simplemente notifica que ha sucedido una acción en el teclado, la interrupción causa que el microprocesador haga a un lado lo que estaba realizando y ejecute un programa para su manejo, el cual envía una señal para que reporte que ha sucedido; el teclado responde enviando el código de exploración de la tecla que fue presionada.

La rutina para el tratamiento de la interrupción traduce el código de exploración a valores de dos bytes; el byte de orden más bajo contiene el código ASCII y se le denomina BYTE PRINCIPAL; el byte de orden más alto corresponde a el código de EXPLORACIÓN y se le denomina BYTE AUXILIAR, este par de bits, es colocado en el buffer del ROM BIOS.





CODIGOS DE EXPLORACION PARA EL TECLADO 101/102 TECLAS (PC/AT Y PS/2)

VALORES EN HEXADECIMAL

Cuando el BYTE PRINCIPAL es diferente de cero; es decir, tiene un valor entre 1 y 256 (01H a FFH), significa que ha sido presionada una de los caracteres estándar del teclado, el cual se encuentra contenido dentro del código ASCII; sin embargo, si este byte tiene un valor de cero (00H), significa que ha sido presionada una tecla especial. Los códigos de exploración o valores para el byte auxiliar de estas teclas especiales y combinaciones validas de ellas pueden observarse en la siguiente tabla:

VALOR		
HEX	DEC	TECLA PULSADA
3B	59	F1
3C	60	F2
3D	61	F3
3E	62	F4
3F	63	F5
40	64	F6
41	65	F7
42	66	F8
43	67	F9
44	68	F10
85	133	F11
86	134	F12
68	104	Alt-F1
69	105	Alt-F2
6A	106	Alt-F3
6B	107	Alt-F4
6C	108	Alt-F5
6D	109	Alt-F6
6E	110	Alt-F7
6F	111	Alt-F8
70	112	Alt-F9
71	113	Alt-F10
8B	139	Alt-F11
8C	140	Alt-F12
78	120	Alt-1
79	121	Alt-2
7A	122	Alt-3

VALOR		
HEX	DEC	TECLA PULSADA
7B	123	Alt-4
7C	124	Alt-5
7D	125	Alt-6
7E	126	Alt-7
7F	127	Alt-8
80	128	Alt-9
81	129	Alt-0
82	130	Alt-guión
83	131	Alt-=
54	84	Shift-F1
55	85	Shift-F2
56	86	Shift-F3
57	87	Shift-F4
58	88	Shift-F5
59	89	Shift-F6
5A	90	Shift-F7
5B	91	Shift-F8
5C	92	Shift-F9
5D	93	Shift-F10
87	135	Shift-F11
88	136	Shift-F12
10	16	Alt-Q
11	17	Alt-W
12	18	Alt-E
13	19	Alt-R
14	20	Alt-T
15	21	Alt-Y

VALOR		
HEX	DEC	TECLA PULSADA
16	22	Alt-U
17	23	Alt-I
18	24	Alt-O
19	25	Alt-P
1E	30	Alt-A
1F	31	Alt-S
20	32	Alt-D
21	33	Alt-F
22	34	Alt-G
23	35	Alt-H
24	36	Alt-J
25	37	Alt-K
26	38	Alt-L
2C	44	Alt-Z
2D	45	Alt-X
2E	46	Alt-C
2F	47	Alt-V
30	48	Alt-B
31	49	Alt-N
32	50	Alt-M
5E	94	Ctrl-F1
5F	95	Ctrl-F2
60	96	Ctrl-F3
61	97	Ctrl-F4
62	98	Ctrl-F5

VALOR		
HEX	DEC	TECLA PULSADA
63	99	Ctrl-F6
64	100	Ctrl-F7
65	101	Ctrl-F8
66	102	Ctrl-F9
67	103	Ctrl-F10
89	137	Ctrl-F11
8A	138	Ctrl-F12
0F	15	Shift-Tab
47	71	Home (Inicio)
48	72	Flecha Arriba
49	73	PgUp (RePág)
4B	75	Flecha izquierda
4D	77	Flecha Derecha
4F	79	End (Fin)
50	80	Flecha Abajo
51	81	PgDn (AvPág)
52	82	Insert (Ins)
53	83	Del (Supr)
72	114	Ctrl-PrtSc (Ctrl-Imp Pant)
73	115	Ctrl-Flecha izquierda
74	116	Ctrl-Flecha derecha
75	117	Ctrl-End (Ctrl-Fin)
76	118	Ctrl-PgDn (Ctrl-AvPág)
77	119	Ctrl-Home (Ctrl-Inicio)
84	132	Ctrl-PgUp (Ctrl-RePág)

Tabla 2.3

El primer problema que nos encontramos, es que las funciones de lectura que nos presentan los compiladores no leen estos dos bytes, solamente obtienen los ocho bits de orden bajo, perdiéndose los 8 bits restantes; esto no causa ningún problema cuando la tecla presionada corresponde a algún carácter contenido dentro del código ASCII, ya que con ellos podemos detectar cual tecla fue presionada.

Por este motivo, necesitamos los dos bytes que se generan (CÓDIGO ASCII y CÓDIGO DE EXPLORACIÓN) para poder detectar cualquier tecla que se presione; primeramente necesitamos del código ASCII, si este código es diferente de cero, podemos identificarla fácilmente, sin embargo, si este código es cero, tendremos que utilizar el código de exploración para verificar cual fue presionada (F1, F2 ,..., F10, ↑ , ↓, →, ←, INICIO, FIN, RE PAG, etc.), de esta manera podemos tener un control total sobre el teclado; para la realización de las funciones que se generarán, necesitaremos alguna que realice esta tarea , es decir, que nos retorne tanto el código ASCII como el código de exploración; en el compilador TURBO C++ existe la función ***bioskey()***, la cual realiza esta tarea, su funcionamiento es el siguiente :

bioskey(*opcion*)

Esta función utiliza la interrupción *22* (*16 HEX*), y el parámetro "opcion" determina la operación a realizar y los valores retornados, como se muestra en la siguiente tabla:

opcion	DESCRIPCIÓN
0	- Si los 8 bits mas bajos son diferentes de cero, la función retorna el carácter ASCII de la tecla presionada, esperando la siguiente tecla que se presione. - Si los 8 bits bajos son ceros, entonces los 8 bits altos contienen el código de exploración de la tecla presionada
1	Espera hasta que una tecla sea presionada, retorna un valor de cero mientras no se presione alguna tecla; se retorna un valor de -1 si CTRL-BRK ha sido presionado. La tecla presionada es retornada por la siguiente llamada de bioskey() con un parámetro de cero
2	Retorna un byte codificado en el cual puede apreciarse el estado de las teclas INS, CAPS, NUM LOCK, ALT, CTRL, SHIFT izquierdo y SHIFT derecho.

TABLA 2.4

Esta función cumple con nuestros requerimientos, en caso de no existir se tendrá que buscar alguna equivalente o implementarse usando la interrupción *22*, designada para uso de este dispositivo.

2.4 ADAPTADORES DE VIDEO.

Existen varios tipos de adaptadores de video, los cuales determinan las características de los caracteres que se pueden desplegar, dependiendo del tipo de adaptador de video que tenga nuestra computadora determinara los siguientes aspectos:

- El número de caracteres que se despliegan en la pantalla.
- La cantidad de pixeles que se pueden desplegar en modo gráfico (RESOLUCIÓN).
- El número de colores que podemos desplegar.
- El tamaño de la memoria para desplegar gráficos.
- Las coordenadas de la pantalla virtual para la utilización del ratón.

Como se puede observar, el adaptador de video determina una gran cantidad de aspectos que influyen directamente en varias características y limitantes de nuestro programa; de aquí la importancia de conocer el tipo de adaptador de video en el cual funcionarán nuestro programas, para que de esta forma su funcionamiento en ellas este asegurado.

El adaptador de video es una tarjeta de circuitos integrados, la cual es insertada en los slots de expansión de la tarjeta principal de la computadora; algunos marcas (IBM, ACER, HP, COMPAQ, etc., y Portatiles en general) construyen algunos modelos las cuales ya la tienen integrada en la tarjeta principal; este adaptador de video contiene un bloque de memoria dedicada que mantiene la información desplegada en el monitor, los adaptadores de video mas comunes son:

- Adaptador de desplegado monocromatico (MDA Monochrome Display Adapter).

- Adaptador gráfico de color (CGA Color Graphics Adapter).
- Adaptador Gráfico Mejorado (EGA Enhanced Graphics Adapter).
- Arreglo Gráfico Multi-color (MCGA Multi-Color Graphical Array).
- Arreglo Gráfico de video (VGA Video Graphics Array).

La siguiente tabla muestra los modos de video mas comunes, sus características y los tipos de adaptadores que la soportan:

MODO	TIPO	DIMENSIONES	ADAPTADORES
0	TEXTO B/N	40 X 25	CGA,EGA,MCGA,VGA
1	TEXTO 16 COLORES	40 X 25	CGA,EGA,MCGA,VGA
2	TEXTO B/N	80 X 25	CGA,EGA,VGA
3	TEXTO 16 COLORES	80 X 25	CGA,EGA,VGA
4	GRAFICOS 4 COLORES	320 X 200	CGA,EGA,VGA
5	GRAFICOS 4 COLORES	320 X 200	CGA,EGA,VGA
6	GRAFICOS 2 COLORES	640 X 200	CGA,EGA,VGA
7	TEXTO B/N	80 X 25	CGA,EGA,MCGA,VGA
8	GRAFICOS 16 COLORES	160 X 200	PCjr
9	GRAFICOS 16 COLORES	320 X 200	PCjr
10	GRAFICOS 4 COLORES PC jr. 15 COLORES EGA	640 X 200	PCjr,EGA
13	GRAFICOS 16 COLORES	320 X 200	EGA,VGA
14	GRAFICOS 16 COLORES	640 X 200	EGA,VGA
15	GRAFICOS 4 COLORES	640 X 350	EGA,MCGA,VGA

TABLA 2.5

Las tarjetas de video mas avanzadas, soportan los modos de video de sus antecesores, de manera que la compatibilidad entre ellas y el funcionamiento de los programas en las nuevas tarjetas este asegurado. De los tipos de tarjetas mencionados anteriormente, algunos de ellas son obsoletas, por ejemplo MDA, HERCULES, CGA y EGA, las cuales ya no se fabrican, el estándar en los últimos años lo ha establecido la tarjeta VGA, sin embargo, hoy en día, la mayoría de los equipos nuevos incluyen la tarjeta SVGA (Super VGA), la cual, parece va a establecer el standard el los próximos años; incluso, algunos de los paquetes que se pueden encontrar en la

actualidad tienen como requerimiento que la computadora tenga esta tarjeta.

EL BUFFER DE VIDEO :

Las computadores PC y compatibles usan una sección de memoria conocida como el buffer de video, la cual es utilizada para su uso, la dirección de inicio de la memoria varia de acuerdo al tipo de adaptador que se este utilizando, por ejemplo :

- Para cualquiera de los modos de Texto en 16 colores, la dirección de memoria empieza a partir de la posición HB800:0000H.

- Para los modos de video gráficos, depende completamente del adaptador y tipo de video que se este utilizando:

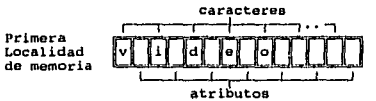
  - Para los adaptadores CGA y Compatibles comienza en el segmento de memoria HB800:0000
  - Para las tarjetas monocromaticas (MCGA, HERCULES y otras) empieza en HB000:0000
  - Para las tarjetas EGA y VGA empiezan en HA000:0000

Esta memoria dedicada especialmente al video es utilizada de la siguiente forma en **todos los modos de video en texto con 16 colores:**

- La primera localidad de la memoria es utilizada para el primer carácter que se despliega en el monitor.

- La siguiente localidad contiene el atributos para el carácter.

- La tercera localidad de memoria contiene el segundo carácter de la pantalla, la cuarta localidad tiene el atributo del carácter anterior, y así sucesivamente, por ejemplo, cuando se tiene escrito la palabra "video" en la esquina superior izquierda de la pantalla, su mapa de video será :



Este arreglo de memoria también es conocido como **MAPA DE MEMORIA DEL VIDEO**, debido a que en el puede encontrarse todos los caracteres que están desplegados en el monitor, así como sus características de atributo; cuando se hace un cambio en el buffer del video (memoria del video), inmediatamente tiene efecto en la pantalla y se refleja el cambio en la misma.

ATRIBUTOS EN MODO TEXTO :

Los tipos de atributos que se pueden utilizar en modo texto dependen del tipo de tarjeta de video que se este utilizando; por ejemplo :

- En tarjetas monocromaticas pueden generarse texto normal, intenso, inverso y subrayado.

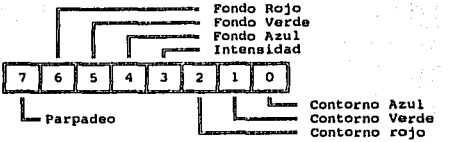
- En la tarjeta de video CGA puede generarse texto, con color de fondo y del carácter en base a paletas de 4 colores cada uno, dependiendo de la paleta que se este utilizando serán los 4 colores que se puedan utilizar.

- En las tarjetas de video EGA y VGA pueden utilizarse 16 colores para el carácter y 8 para el color de fondo.

Cada carácter desplegado en la pantalla tiene su propio atributo, de esta manera, pueden desplegarse caracteres con diferente color. Hoy en la actualidad, la tarjeta de video mas utilizada es la VGA, sin embargo, ha comenzado a ser desplazada con las nuevas tecnologías, como es el caso de la tarjeta SVGA (Super VGA).

Para casos prácticos, explicaremos como funciona el almacenamiento del video para el adaptador EGA y VGA en donde se

pueden desplegar hasta 16 colores simultáneamente; en ellas, el byte de atributo es utilizado para almacenar el color del carácter, el color del fondo, la intensidad y la característica de parpadeo, la forma en que lo almacena es la siguiente :



De esta forma, se pueden producir combinaciones de colores para obtener a los restantes, por ejemplo: Rojo y Verde producen el color CAFE si la intensidad no esta fija, en caso contrario producen el color AMARILLO. la tabla completa de los colores que se pueden producir con las combinaciones de estos bits son :

BIT'S								COLOR	ATRIBUTO (DECIMAL)
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	NEGRO	0
0	0	0	0	0	0	0	1	AZUL	1
0	0	0	0	0	0	1	0	VERDE	2
0	0	0	0	0	0	1	1	AZUL CLARO (CYAN)	3
0	0	0	0	0	1	0	0	ROJO FUERTE	4
0	0	0	0	0	1	0	1	ROSA (MAGENTA)	5
0	0	0	0	0	1	1	0	ROJO CLARO (CAFE)	6
0	0	0	0	0	1	1	1	BLANCO	7
0	0	0	0	1	0	0	0	GRIS	8
0	0	0	0	1	0	0	1	AZUL INTENSO	9
0	0	0	0	1	0	1	0	VERDE INTENSO	10
0	0	0	0	1	0	1	1	AZUL CLARO INTENSO	11
0	0	0	0	1	1	0	0	ROJO INTENSO	12
0	0	0	0	1	1	0	1	ROSA INTENSO	13
0	0	0	0	1	1	1	0	AMARILLO INTENSO	14
0	0	0	0	1	1	1	1	BLANCO INTENSO	15

TABLA 2.6

Para los colores de fondo, solamente se tienen 3 bits para poder almacenar su atributo, por lo tanto, las combinaciones posibles son :

BIT's								COLOR	ATRIBUTO (DECIMAL)
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	NEGRO	0
0	0	0	1	0	0	0	0	AZUL	16
0	0	1	0	0	0	0	0	VERDE	32
0	0	1	1	0	0	0	0	AZUL CLARO (CYAN)	48
0	1	0	0	0	0	0	0	ROJO FUERTE	64
0	1	0	1	0	0	0	0	ROSA (MAGENTA)	80
0	1	1	0	0	0	0	0	ROJO CLARO	96
0	1	1	1	0	0	0	0	BLANCO	112

TABLA 2.7

El ultimo bit representa al parpadeo, si este bit esta encendido, es decir, contiene un 1, el carácter asociado a este atributo se encontrara parpadeando, el atributo en decimal asociado a esta característica será 128.

2.5 EL RATÓN COMO DISPOSITIVO DE ENTRADA SECUNDARIO.

Este dispositivo ha generalizado su uso hacia muy diversas aplicaciones, y es considerado el segundo dispositivo de entrada de mayor uso en los computadores. El ratón, generalmente es usado para manipular el cursor de texto o gráfico que se presenta en la pantalla; esta compuesto de sensores internos los cuales controlan el movimiento del cursor; cuando el ratón es cambiado de posición, los sensores internos detectan este movimiento y mandan información

al computador sobre esta dirección, para que se refleje en el movimiento del cursor.

El ratón puede tener 1, 2 o 3 botones dependiendo del fabricante; estos botones proveen datos adicionales de entrada que no tienen relación con el movimiento del mismo. Esencialmente, este dispositivo puede encontrarse en 2 formas o tipos :

- La primera de ellas corresponde al ratón mecánico, el cual contiene una bola en su parte inferior, cuando se mueve el ratón de una posición a otra, la bola también se mueve y esto activa los sensores internos del mismo, los cuales trasladan esta información en datos de movimiento direccional que son utilizados para mover el cursor.

- El segundo tipo corresponde al ratón óptico, el cual usa dos leds y dos fototransistores para detectar el movimiento. Un led emite luz roja y el otro produce luz infrarroja, este ratón óptico es usado con un pad especial que altera la luz de los LED's, el pad contiene líneas en dos direcciones de tal manera que cuando el ratón se mueve en una dirección la luz roja es absorbida y en la otra dirección la luz infrarroja es absorbida, el color y número de rompimiento de la captación de luz por parte de los fototransistores determinan la dirección y la distancia del movimiento del cursor.

Este dispositivo electrónico, envía señales a la computadora, las cuales representan para el software movimientos del cursor y presión de botones; sin embargo, estas señales son muy difíciles de utilizar en su forma original además de que varían de acuerdo al tipo y marca de ratón que se este utilizando; es por esto que los fabricantes proveen lo que se llama el manejador del ratón (DRIVER) para dar al programador una interface consistente y fácil de utilizar. Un manejador del ratón en realidad es un software que le permite al sistema operativo interpretar las señales provenientes del ratón; para efecto de esta tesis, explicaremos el manejador de la marca **MICROSOFT** por ser uno de los más utilizados en la actualidad; el cual provee de 53 funciones por medio de la

interrupción No. *33H* las cuales permiten al programa checar el movimiento y el estado de los botones del mismo.

PANTALLA VIRTUAL :

El uso del ratón tiene una relación muy estrecha con el tipo de monitor y modo de video que se este utilizando; el modo de video define el numero de pixeles y el tipo de objetos que se podrán desplegar en la pantalla; la forma en que el ratón interactua con la pantalla es a través de lo que se llama una **"PANTALLA VIRTUAL"**, la cual es una especie de malla que se sobrepone a la pantalla física, esta pantalla virtual esta compuesta como una matriz de puntos horizontales y verticales, la longitud o tamaño de esta matriz varia de acuerdo a el modo de video que se este utilizando; todo lo que se relacione con la posición del cursor en la pantalla tendrá que utilizarse las coordenadas en pantalla virtual; en la tabla 2.8 puede apreciarse los modos de video así como sus dimensiones de la pantalla virtual, la cual fue obtenida del libro **MICROSOFT MOUSE PROGRAMMER**[8] págs 81,82 y 83 la parte correspondiente a el tamaño de la celda representa la mínimo resolución del movimiento del ratón en las coordenadas X y Y en términos de pantalla virtual.

En los modos de video en texto no se puede hacer referencia a todos los pixeles individuales de la pantalla virtual, puesto que solamente se pueden desplegar caracteres los cuales están formados por un grupo de 8x8 pixeles o 16x8 pixeles, solamente se podrá hacer referencia a la esquina superior izquierda de cada carácter, tomando en consideración que la esquina superior izquierda de la pantalla corresponde a las coordenadas (0,0), de esta manera el siguiente carácter estará en la posición (8,0) o (16,0) dependiendo del modo de video que se este utilizando, y así sucesivamente hasta terminar con ese renglón; la primera coordenada representa el eje

[8] MICROSOFT MOUSE PROGRAMMER'S. Ed. MICROSOFT PRESS. Segunda Edición 1991. Págs. 81,82 y 83.

horizontal de la pantalla o eje X y la segunda coordenada representa al eje vertical o eje Y.

Entre todas las funciones que contiene el manejador del ratón que son 53 en total, podemos encontrar una gran variedad de conceptos relacionados con su uso y características principales como pueden ser información sobre la presión y soltado de los botones, información sobre la pantalla virtual, aparecer el cursor y ocultarlo, etc.; la información que se podría dar sobre este dispositivo es bastante y como solamente se tomará este dispositivo como una entrada alternativa de información no se precisará ahondar demasiado en todo lo que respecta a su uso y a las 53 funciones que lo componen, solamente se mencionarán las mas importantes para que nos auxiliemos de ellas en la construcción de nuestras rutinas.

MODO HEX.	ADAPTADOR GRÁFICO	TIPO	PANTALLA VIRTUAL (X,Y)	TAMAÑO DE LA CELDA
0	CGA,EGA,MCGA,VGA,3270	TEXTO	640 X 200	16 X 8
1	CGA,EGA,MCGA,VGA,3270	TEXTO 40 X 25 16 COLORES	640 X 200	16 X 8
2	CGA,EGA,MCGA,VGA,3270	TEXTO 80 X 25 16 COLORES	640 X 200	8 X 8
3	CGA,EGA,MCGA,VGA,3270	TEXTO 80 X 25 16 COLORES	640 X 200	8 X 8
4	CGA,EGA,MCGA,VGA,3270	GRÁFICO 320 X 200 4 COLORES	640 X 200	2 X 1
5	CGA,EGA,MCGA,VGA,3270	GRÁFICO 320 X 200 4 COLORES	640 X 200	2 x 1
6	CGA,EGA,MCGA,VGA,3270	GRÁFICO 640 X 200 2 COLORES	640 X 200	1 X 1
7	CGA,EGA,MCGA,VGA,3270 ,MDA	TEXTO 80 X 25	640 X 200	8 X 8
8	PCjr. SOLAMENTE	GRÁFICO 160 X 200 16 COLORES	640 X 200	4 X 1
9	PCjr. SOLAMENTE	GRÁFICO 320 X 200 16 COLORES	1280 X 200	1 X 1
A	PCjr. SOLAMENTE	GRÁFICO 640 X 200 16 COLORES	640 X 200	1 X 1
D	EGA,VGA	GRÁFICO 320 X 200 16 COLORES	640 X 200	2 X 1

MODO HEX.	ADAPTADOR GRÁFICO	TIPO	PANTALLA VIRTUAL (X,Y)	TAMAÑO DE LA CELDA
E	EGA,VGA	GRÁFICO 640 X 200 16 COLORES	640 X 200	1 X 1
F	EGA,VGA,MDA	GRÁFICO 640 X 350	640 X 350	1 X 1
10	EGA,VGA	GRÁFICO 640 X 350 16 COLORES	640 X 350	1 X 1
11	MCGA,VGA	GRÁFICO 640 X 480 2 COLORES	640 X 480	1 X 1
12	VGA	GRÁFICO 640 X 480 16 COLORES	640 X 480	1 X 1
13	MCGA,VGA	GRÁFICO 320 X 200 256 COLORES	640 X 200	2 x 1
23	GENIUS VHR	GRÁFICO 728 X 1008 2 COLORES	728 X 1008	1 X 1
24	HP VECTRA	GRÁFICO 640 X 400	640 X 400	1 X 1
25	IBM 8514,XGA	GRÁFICO 1024 X 768 16 COLORES	1024 X 768	1 X 1

TABLA 2.8 MODOS DE VIDEO Y TAMAÑO DE LA PANTALLA VIRTUAL.

MDA = ADAPTADOR DE DESPLEGADO MONOCROMATICO (MONOCHROME DISPLAY ADAPTER)
CGA = ADAPTADOR GRÁFICO DE COLOR (COLOR GRAPHICS ADAPTER)
EGA = ADAPTADOR GRÁFICO MEJORADO (ENHANCED GRAPHICS ADAPTER)
MCGA = ARREGLO GRÁFICO MULTI-COLOR (MULTI-COLOR GRAPHICS ARRAY)
VGA = ARREGLO GRÁFICO DE VIDEO (VIDEO GRAPHICS ARRAY)
XGA = ARREGLO GRÁFICO EXTENDIDO (EXTENDED GRAPHICS ARRAY)

LA BANDERA INTERNA DEL CURSOR :

Independientemente del tipo de cursor desplegado, se mantiene una bandera interna que determina cuando aparecerá el cursor en la pantalla, el valor de esta bandera es 0 o 1; cuando el valor de esta bandera es 0 el cursor es desplegado, cuando el valor de la bandera es diferente que cero el cursor se oculta.

NUMERO DE FUNCIÓN	NOMBRE DE LA FUNCIÓN
0	ESTADO Y REINICIALIZA EL RATÓN
1	MUESTRA EL CURSOR
2	OCULTA EL CURSOR
3	RETORNA EL ESTADO DEL CURSOR Y POSICIÓN DEL RATÓN
4	FIJA LA POSICIÓN DEL CURSOR DEL RATÓN
5	OBTIENE INFORMACIÓN DE BOTONES PRESIONADOS
6	OBTIENE INFORMACIÓN DE SOLTADO DE BOTONES
7	FIJA LA POSICIÓN HORIZONTAL MÍNIMA Y MÁXIMA DEL CURSOR
8	FIJA LA POSICIÓN VERTICAL MÍNIMA Y MÁXIMA DEL CURSOR
9	FIJA EL BLOQUE DEL CURSOR GRÁFICO
10	FIJA EL TIPO DE CURSOR EN MODO TEXTO
11	LEE EL CONTADOR DE MOVIMIENTO DEL RATÓN
13	EMULACIÓN DE PLUMA LUMINOSA MODO ENCENDIDO
14	EMULACIÓN DE PLUMA LUMINOSA MODO APAGADO
15	FIJA SENSITIVIDAD DEL CURSOR (MICKEY/PIXEL)
16	APAGADO CONDICIONAL
19	FIJA DOBLE VELOCIDAD
20	CAMBIA LAS SUBRUTINAS DE INTERRUPCIÓN
22	SALVA EL ESTADO DEL MANEJADOR DEL RATÓN
23	RESTAURA EL ESTADO DEL MANEJADOR DEL RATÓN
26	FIJA LA SENSITIVIDAD DEL RATÓN
27	OBTIENE LA SENSITIVIDAD DEL RATÓN
31	DESABILITA EL MANEJADOR DEL RATÓN
32	HABILITA EL MANEJADOR DEL RATÓN
33	REINICIALIZA EL SOFTWARE
36	OBTIENE LA VERSIÓN DEL MANEJADOR Y TIPO DE MOUSE.
37	OBTIENE INFORMACIÓN GENERAL DEL MANEJADOR.
38	OBTIENE MÁXIMAS COORDENADAS VIRTUALES.
40	FIJA MODO DE VIDEO
41	ENUMERA LOS MODOS DE VIDEO
47	REINICIALIZA EL HARDWARE DEL RATÓN
49	OBTIENE COORDENADAS VIRTUALES MÍNIMAS Y MÁXIMAS

TABLA 2.9

SENSITIVIDAD DEL CURSOR :

El movimiento del ratón por medio de lo bola rodante que contiene representan la dirección del cursor, este movimiento se expresa en una unidad de distancia llamada **mickey**, la cual es aproximadamente 1/200 pulgadas, cuando se mueve el ratón se obtienen distancias en unidades mickey, el manejador del ratón se encarga de trasladar estas distancias en movimiento del cursor dados en pixeles; el numero de pixeles que mueve el cursor no siempre corresponde uno a uno con el número de mickey que se mueve la bola rodante, el manejador del ratón define una sensitividad para el ratón el cual es el número de mickey requeridos para mover 8 pixeles en la pantalla.

A continuación mencionaremos algunas de las funciones más importantes de la tabla anterior, así como la tarea que realiza, los parámetros que necesita en los registros del microprocesador y los valores que retorna :

FUNCIÓN 0 : REINICIALIZA Y OBTIENE EL ESTADO DEL RATÓN.

PARÁMETROS : **AX** = 0

RETORNA : **AX** = ESTADO DEL RATÓN
BX = NUMERO DE BOTONES

DESCRIPCIÓN : Retorna el estado actual del mouse tanto en software como en hardware; si se tiene instalado en software y hardware del ratón (manejador ver 6.25 o mayor) el estado es -1, si no tiene instalado el valor es de 0; además, si el puntero del cursor esta visible, esta función lo oculta como parte del proceso de inicialización; los valores que inicializa por default esta función son:

```
Posición del cursor          = Centro de la pantalla
Bandera interna del cursor = -1 (cursor Oculto)
Cursor Gráfico               = Flecha
```

```
Cursor de texto              = Bloque en video inverso
relación mickey por radio
          horizontal         = 8  a 8
          vertical           = 16 a 8
Posición horizontal mínima
          del cursor         = 0
Posición horizontal máxima
del cursor                   = Depende del modo de video seleccionado.
Posición vertical mínima
          del cursor         = 0
Posición vertical máxima     = Depende del modo de video seleccionado.
```

FUNCIÓN 1 : MUESTRA EL CURSOR.

PARÁMETROS : **AX** = 1

RETORNA : NADA

DESCRIPCIÓN : Incrementa la bandera interna del cursor, si el valor de la bandera es 0 despliega el cursor en la pantalla; el valor por default de la bandera del cursor es -1, y se establece cuando se inicializan los datos del ratón. Si el valor de la bandera interna del cursor es 0, la función mantiene este valor.

FUNCIÓN 2 : CURSOR OCULTO

PARÁMETROS : **AX** = 2

RETORNA : NADA

DESCRIPCIÓN : Oculta el cursor de la pantalla y decrementa la bandera interna del cursor, esta función solamente oculta el cursor, el manejador del ratón continua checando el movimiento del mismo y por lo tanto continua cambiando la posición del cursor de acuerdo a la posición del ratón.

FUNCIÓN 3 : OBTIENE EL ESTADO DE LOS BOTONES Y POSICIÓN DEL RATÓN

PARÁMETROS : **AX** = 3

RETORNA : **BX** = ESTADO DE LOS BOTONES
CX = COORDENADAS HORIZONTALES DEL CURSOR
DX = COORDENADAS VERTICALES DEL CURSOR

DESCRIPCIÓN : Retorna el estado del botón izquierdo y botón derecho del ratón, también retorna el estado de las coordenadas de pantalla virtuales verticales y horizontales

El estado de los botones es un valor entero: El bit 0 representa el botón izquierdo y el Bit 1 representa el botón derecho, el valor de un bit de 1 corresponde al botón presionado y 0 de lo contrario.

FUNCIÓN 4 : FIJA LA POSICIÓN DEL CURSOR DEL RATÓN.

PARÁMETROS : **AX** = 4
CX = NUEVA COORDENADA HORIZONTAL DEL CURSOR
DX = NUEVA COORDENADA VERTICAL DEL CURSOR

RETORNA : NADA

DESCRIPCIÓN : Fija la posición del cursor en las nuevas coordenadas vertical y horizontal especificadas, las cuales deben de ser coordenadas virtuales; los valores de los parámetros deben de estar dentro de las coordenadas mínima y máxima del rango dependiendo del modo de pantalla seleccionado.

FUNCIÓN 5 : OBTIENE INFORMACIÓN SOBRE BOTONES PRESIONADOS.

PARÁMETROS : **AX** = 5
BX = BOTÓN

RETORNA : **AX** = ESTADO DEL BOTÓN
BX = NUMERO DE PRESIONES DEL BOTÓN
CX = COORDENADA HORIZONTAL DEL CURSOR EN LA ULTIMA PRESIÓN
DX = COORDENADA VERTICAL DEL CURSOR EN LA ULTIMA PRESIÓN

DESCRIPCIÓN : El parámetro en **BX** especifica cual botón va a checar, si el valor es 0 checara el estado del botón izquierdo, si el parámetro es 1 checara el botón derecho. El estado del botón en un valor entero, el bit 0 representa a el botón izquierdo y el bit 1 representa a el botón derecho; el valor de un bit como 1 corresponde a botón presionado y 0 de lo contrario.

FUNCIÓN 6 : OBTIENE INFORMACIÓN SOBRE CUANDO ES SOLTADO EL BOTÓN.

PARÁMETROS : **AX** = 6
BX = BOTÓN

RETORNA : **AX** = ESTADO DEL BOTÓN
BX = NUMERO DE VECES QUE ES SOLTADO EL BOTÓN
CX = COORDENADA HORIZONTAL DEL CURSOR EN LA ULTIMA VEZ QUE SE SOLTÓ EL BOTÓN
DX = COORDENADA VERTICAL DEL CURSOR EN LA ULTIMA VEZ QUE SE SOLTÓ EL BOTÓN

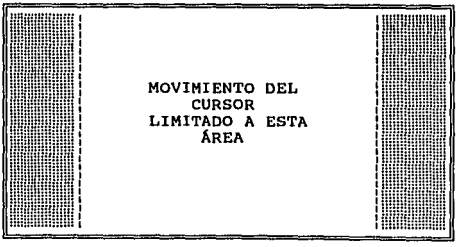
DESCRIPCIÓN : El parámetro en **BX** especifica cual botón va a checar, si el valor es 0 checara el botón izquierdo y si es 1 el botón derecho, el estado del botón es un valor entero; el bit 0 representa al botón izquierdo y el 1 al botón derecho, el valor del bit a 1 corresponde a botón presionado y 0 a botón suelto o colocado en libertad. Los valores de las coordenadas vertical y horizontal correspondan a coordenadas virtuales de pantallas, los cuales corresponden a la última vez que se soltó el botón.

FUNCIÓN 7 : FIJA LA POSICIÓN HORIZONTAL MÍNIMA Y MÁXIMA DEL CURSOR.

PARÁMETROS : **AX** = 7
CX = POSICIÓN MÍNIMA
DX = POSICIÓN MÁXIMA

RETORNA : NADA

DESCRIPCIÓN : Restringe el movimiento del cursor a una área especifica. Fija la posición de las coordenadas horizontales mínima y máxima para el movimiento del cursor en la pantalla;

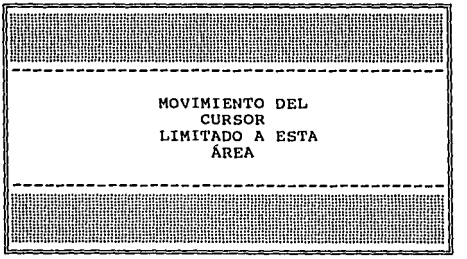


FUNCIÓN 8 : FIJA LA POSICIÓN VERTICAL MÍNIMA Y MÁXIMA DEL CURSOR.

PARÁMETROS : **AX** = 8
CX = POSICIÓN MÍNIMA
DX = POSICIÓN MÁXIMA

RETORNA : NADA

DESCRIPCIÓN : Fija la posición de las coordenadas verticales mínima y máxima del cursor en la pantalla; restringe el movimiento del cursor a una área especifica.



FUNCIÓN 11 : LEE EL CONTADOR DE MOVIMIENTO DEL RATÓN

PARÁMETROS : **AX** = 11

RETORNA : **CX** = CONTADOR MICKEY HORIZONTAL
DX = CONTADOR DE MICKEY VERTICAL

DESCRIPCIÓN : Retorna el numero de mickey horizontal y vertical que se desplazo el ratón desde la última llamada a esta función. Este contador de unidades mickey varia en un rango de -32,768 a 32,767, un contador positivo horizontal indica movimiento hacia la derecha, cuando el contador horizontal es negativo indica movimiento hacia la izquierda. Un contador positivo vertical indica movimiento hacia abajo de la pantalla, cuando es negativo indica movimiento hacia la parte superior de la pantalla.

FUNCIÓN 15: FIJA LA RELACIÓN MICKEY/PIXEL.

PARÁMETROS : **AX** = 15
CX = RELACIÓN HORIZONTAL MICKEY/PIXEL
DX = RELACIÓN VERTICAL MICKEY/PIXEL

RETORNA : NADA

DESCRIPCIÓN : Fija la relación mickey por pixel para el movimiento horizontal y vertical del ratón, la relación especifica el número de mickey por cada 8 pixeles de la pantalla virtual; los valores deben de estar dentro del rango de 1 hasta 32,767.
El valor por default para la relación horizontal es 8 mickey equivalen a 8 pixeles de la pantalla virtual, para la relación vertical es 16 mickey equivalen a 8 pixeles de la pantalla virtual.

FUNCIÓN 38 : OBTIENE LAS MÁXIMAS COORDENADAS DE LA PANTALLA VIRTUAL.

PARÁMETROS : **AX** = 38

RETORNA : **BX** = Bandera desabilitada del ratón
CX = Máxima X virtual
DX = Máxima Y virtual

DESCRIPCIÓN : Retorna una bandera en **BX** la cual indica si el manejador del ratón esta desabilitado e indica las máximas coordenadas virtuales en **CX** y **DX** para el modo actual de video. Las máximas coordenada virtuales son los valores por default para el actual modo de video; si se utilizo la función 7 y 8 para limitar el área de movimiento del cursor, esta función ignora estas coordenadas y retorna los valores máximos para el modo de video que se esta utilizando. Esta función es disponible en manejadores versión 6.26 o mas actualizada.

FUNCIÓN 49 : OBTIENE COORDENADAS VIRTUALES MÍNIMAS Y MÁXIMAS.

PARÁMETROS : **AX** = 49

RETORNA : **AX** = MÍNIMO X VIRTUAL
BX = MÍNIMO Y VIRTUAL
CX = MÁXIMO X VIRTUAL
DX = MÁXIMO Y VIRTUAL

DESCRIPCIÓN : Retorna las coordenadas verticales y horizontales mínima y máxima para el modo de video actual. Estos valores se ven influenciados por las funciones 7 y 8, los cuales determinan los rangos dentro de los cuales se limita el movimiento del cursor.

Estas son algunas de las funciones de mayor uso dentro de lo que es el uso del ratón, sin embargo como se habrá podido observar, faltaron un número muy grande de ellas por explicar, esto se debe a la falta de uso de la mayoría de ellas; además, seria imposible analizarlas dentro de esta tesis, si el lector esta interesado en algún tema en particular sobre el uso del ratón o desea introducirse aún mas, le recomendamos leer el libro **MICROSOFT MOUSE PROGRAMMER**[9], en donde podrá encontrar una descripción mas detallada de todas las funciones que contiene el manejador del ratón.

[9] MICROSOFT MOUSE PROGRAMER'S REFERENCE. Obra citada.

CAPITULO III

MÉTODOS DE SELECCIÓN CON MENÚS.

3.1.- OBJETIVOS DEL CAPITULO

Los objetivos principales de este capítulo se pueden sintetizar en los siguientes puntos :

a) Explicar que es un menú.

b) Indicar cuales son los tipos de menús mas importantes y sus características.

c) Analizar los métodos de lectura utilizados en su desplegado.

d) Mencionar los métodos de ayuda mas utilizados tanto para los menús como para otros elementos de interface.

e) Generar las funciones que realicen la tarea de colocar el menú en la pantalla tratando de incluir todas sus características mas representativas.

Todas las funciones que se mencionan en este capítulo se encuentran definidas en al Apéndice I correspondiente al código Fuente, al final del capítulo se incluye un ejemplo utilizando menús.

3.2.- ¿ QUE ES UN MENÚ ?

Se le llama menú a todo elemento de interface con el usuario cuya característica principal es presentar en la pantalla una serie

de opciones en forma de una lista para que el usuario seleccione alguna de ellas, cada una de las cuales representa a un módulo, proceso, o desplegara otra serie de opciones relacionadas mediante otro menú o cualquier otro elemento de interface, es decir, dan acceso a una serie de aplicaciones funcionales u otros elementos de interface[10]; este aspecto es muy importante para poder diferenciarla de lo que se llama una **LISTA DE BOTONES**, la cual se caracteriza por presentar una lista de opciones pero con la diferencia de que no ejecutará ningún módulo o proceso sino que la opción seleccionada servirá para establecer un estado, variable o unidad en nuestro programa; un ejemplo del menú es el siguiente:

Un ejemplo de una lista de botones es el siguiente:

El menú constituye uno de los principales elementos de interface que existen hoy en día, y es muy usado en el desarrollo de software profesional, debido a la gran facilidad de uso y aprendizaje que presenta a el usuario; se caracterizan por presentar una serie de opciones disponibles en forma de una lista, ya sea en posición horizontal, vertical o en una combinación de ambas; de tal manera que el usuario pueda elegir alguna de ellas, facilitando la elección de tareas, dándole mayor flexibilidad, rapidez, control y entendimiento al sistema. Su principal ventaja

10 SUN Microsystem INC. Open Look: Graphical User Interface. Editorial Addison-Wesley Publishing Company INC. Primera Edición 1989.

radica en que las opciones son mostradas en la pantalla, con lo cual, el usuario no tiene que memorizarlas como sucede en el caso de la **"LINEA DE COMANDOS"**, con esto, tanto un usuario experto como uno inexperto podrán utilizarlo fácilmente sin tener que aprender manejos complicados sobre su uso.

En general la lista de opciones que conforman al menú son colocadas en grupos de acuerdo a características que tienen en común, y cada grupo es ordenado en relación a su importancia o frecuencia de uso[11]; no se acostumbra ordenar las opciones alfabéticamente, ya que aquellas de mayor importancia pueden ser colocadas en las últimas posiciones. El uso del orden alfabético es solamente si no es posible el orden por importancia.

Su uso no garantiza que los programas vayan a ser del agrado de los usuarios, ni fáciles de utilizar, ya que deben de considerarse una serie de conceptos fundamentales en su organización como pueden ser:

- Una organización adecuada de las opciones de los menús.
- Una buena estructura menú-sistema.
- Una terminología acorde con el tipo de usuarios que vayan a utilizarlo.
- Tiempo de respuesta rápido.
- Número y secuencia de las opciones de cada menú.
- Ayuda en línea.

3.3.- TIPOS DE MENÚS

Básicamente pueden clasificarse a los menús en 3 tipos diferentes, basándose en la forma en que son desplegados en la pantalla como se muestra a continuación :

11 RAY E. Eberts. User Interface Design. Obra citada.

- **MENÚ HORIZONTAL O PRINCIPAL :**

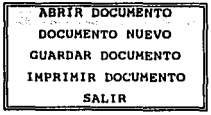
Se caracteriza por presentar todas sus opciones en un solo renglón de la pantalla; se utiliza para representar menús principales, generalmente la selección de alguna opción desplegará otro menú, su principal desventaja consiste en la cantidad de caracteres que se pueden desplegar; en modo texto se limita a 80 caracteres, y en modo gráfico dependerá del tipo de tarjeta de video y el modo gráfico que se este utilizando; por este motivo las opciones que se incluyan deberán de cumplir con las siguientes características:

- Estar reducidas a su mínima expresión.
- Utilizar una terminología acorde con el tipo de usuarios que van a utilizar el sistema.
- El texto deberá de ser claro, conciso y breve

Un ejemplo de este tipo de menú es el siguiente :

- **MENÚ VERTICAL :**

Como su nombre lo dice, se despliega en forma vertical ocupando cada opción un renglón o parte de él, generalmente es usado como menú secundario y de orden inferior; su principal ventaja consiste en que puede tener un numero mayor de opciones que el menú horizontal, además que cada opción puede tener una longitud mayor que el caso del menú horizontal, un ejemplo del menú vertical :



- **MENÚ MATRICIAL O DE OPCIONES MÚLTIPLES :**

Este tipo de menú muy pocas veces es utilizado, su característica principal consiste en que despliega las

opciones tanto en forma vertical como en forma horizontal (en forma de matriz), teniendo la capacidad de desplegar un número mayor de ellas que los dos anteriores, generalmente se usa cuando las opciones son muchas y se desea que todas ellas aparezcan simultáneamente; en lugar se usar este tipo de menú se prefiere dividir las opciones disponibles en varios submenús los cuales se desplegarán conforme el usuario vaya seleccionando; un ejemplo de un menú matricial es:

COPIAR ARCHIVOS	CREAR SUBDIRECTORIO
COPIAR DISCO	BORRAR SUBDIRECTORIO
COPIA DE SEGURIDAD	CAMBIAR SUBDIRECTORIO
FORMATEAR	COMPARAR DISCOS
RESTAURAR	COMPARAR ARCHIVOS
COMPILADOR BASIC	BORRAR ARCHIVOS
COMPILADOR C	CAMBIAR ATRIBUTOS
COMPILADOR FORTRAN	ANTI-VIRUS

Existe otra forma muy popular de representar menús que se conoce como menús en cascada(o **PULL-DOWN**), la cual consiste en representar un menú principal, y una vez que se seleccione una opción se presenta un menú en forma vertical para que el usuario observe otra serie de opciones relacionadas con la que eligió anteriormente, de estas, se podrá seleccionar alguna de ellas, y en caso necesario podrá desplegarse otro menú; por este motivo se le llama menús en cascada, ya que se muestra uno tras otro hasta llegar a mostrar las últimas opciones; aunque no hay un límite en el numero de menús y submenús que se pueden mostrar, se considera que la mejor alternativa es mostrar los menús en un máximo de 3 o 4 niveles.

3.4.- MÉTODOS DE SELECCIÓN EN LOS MENÚS.

La forma en que el usuario pueda seleccionar una opción de un menú puede variar dependiendo del tipo y forma de menú que este utilizando, las formas mas comunes de selección son las siguientes:

- **MEDIANTE IDENTIFICADORES DE NUMEROS PARA LAS OPCIONES :**

Este método despliega las opciones disponibles acompañadas de un número el cual identifica a la opción; en la parte inferior se despliega una pregunta sobre cual opción desea, esperando que el usuario presione un número; este tipo de menú no es muy utilizado profesionalmente debido a que para el usuario es muy difícil asociar números a cada una de las opciones. Un ejemplo de este tipo de menú es:

```
OPCIONES

1) Altas
2) Bajas
3) Cambios
4) Consultas
5) Salida

Seleccione del 1-5 y presione ENTER : ___
```

- **USANDO ALGUNA LETRA CONTENIDA EN LA OPCIÓN :**

Este método de selección elimina la utilización de un número para identificar a cada una de las opciones, en este caso se utiliza una letra de cada una de ellas, la cual tiene que hacerse resaltar para que el usuario pueda identificarla; esto se puede lograr de diferentes formas como pueden ser:

- Con la letra que se representa con mayúsculas
- Con un color diferente al de las demás letras que representan a la opción.
- Con alguna letra subrayada.

Este método es mas utilizado que el anterior debido a que el usuario puede identificar y asociar más fácilmente la opción que desea en base a una letra que integra el enunciado, además, con esta forma de selección se ahorra

tiempo al ejecutar inmediatamente la opción una vez que se ha presionado la tecla que lo identifica sin la necesidad de presionar la tecla <ENTER>, sin embrago, la desventaja de este método consiste en que usuarios inexpertos pueden equivocarse presionando otra tecla en forma accidental, lo cual causará que seleccione una opción que no desea; un ejemplo de este menú es :

```
OPCIONES

Altas
Bajas
Cambios
cOnsultas
Salida

Cual es su selección : __
```

- **MEDIANTE EL USO DE UN CURSOR EN FONDO INVERSO.**

Este método es muy utilizado en cualquier tipo de menú y es el mas preferido por parte de los usuarios novatos, su característica principal consiste en el desplegado de una opción, la que se encuentra activa, en lo que se conoce como fondo inverso, es decir, de todas las opciones que conforman el menú, una es desplegada con un color de fondo diferente, el cual representará aquella que se ejecutará una vez que se presione <ENTER>, este fondo inverso cambiara de posición conforme el usuario vaya utilizando las teclas de flechas.

La principal ventaja de este método de selección consiste en que disminuye la probabilidad de error, ya que el usuario observa cual es la opción que esta escogiendo, la cual esta desplegada con fondo inverso, además de que es muy fácil de manejar; la principal desventaja consiste en que el tiempo de selección aumenta notablemente, ya que el usuario debe posicionar el cursor hasta la opción deseada y presionar la tecla <ENTER> para que se ejecute.

Debido a este problema, se incluye en este tipo de menú la característica de poder desplazarse hacia los extremos una vez que se ha llegado al inicio o final de las opciones desplegadas. Un ejemplo es:

- **MEDIANTE EL USO DEL RATÓN.**

  Este dispositivo apuntador ha tenido un auge muy grande y ha llegado a convertirse en un dispositivo indispensable en muchos programas; el ratón despliega un cursor en la pantalla el cual puede ser un carácter con atributos diferentes en modo texto o una flecha en modo gráfico, el cual cambia de posición de acuerdo a el movimiento del ratón. Para muchos usuarios se facilita la selección de las opciones de los menús posicionando el cursor en la opción deseada y presionar algún botón del mismo; este método tiene las ventajas de que aumenta la rapidez de selección y es muy fácil de entender y aprender por parte de los usuarios.

Como se puede apreciar existen varios métodos de selección en los menús, cada uno con sus ventajas y desventajas, para aprovechar lo mejor de todas ellas, se acostumbra incluir varios de ellos, de manera que el usuario escoja el que mas le agrade, entienda y facilite la selección, esto con el fin de darle una mayor flexibilidad al sistema; generalmente se incluyen los tres últimos métodos; para efecto de las rutinas que generaremos en este capítulo utilizaremos estos 3 métodos de selección.

Existe otro método de lectura alternativo que sustituye la lectura a través de los menús, nos estamos refiriendo a la lectura a través de la **"LINEA DE COMANDOS"**, esta opción suele presentarse en algunos sistemas para agilizar el trabajo a los usuarios

expertos, los cuales consideran que la selección a través de menús no es un buen elemento de interface[12], debido a que el número de pasos para llegar a una opción final en ocasiones es muy largo y tedioso; sin embargo, la LINEA DE COMANDOS no representa una forma de obtener la decisión por parte del usuario hacia un menú, sino que representa una forma alternativa de operar el sistema.

3.5.- CARACTERÍSTICAS SECUNDARIAS DE LOS MENÚS.

Independientemente del tipo de menú y de las formas de selección que se este utilizando, existen una serie de características secundarias que pueden incluirse su desplegado, generalmente son utilizados para dales una mayor presentación, entre las mas comunes podemos encontrar :

- La utilización de colores para las opciones, el fondo inverso y el carácter asociado a la opción; lo cual da una mejor apariencia y presentación al menú.

- Utilizar un recuadro que encierre a todas las opciones, lo cual permite la identificación del elemento de interface en una sola entidad.

- Colocar un titulo en la parte superior del menú, para que el usuario este enterado del menú en el que se encuentra.

- Utilizar el efecto de "SOMBRA" al costado del menú, lo cual da la apariencia de ser un objeto diferente y sobrepuesto a los datos que se encuentra debajo de él.

12 RAY E. EBERTS. USER INTERFACE DESIGN. Obra citada.

Estas son las características secundarias mas utilizadas en los menús, aunque no sea indispensable su utilización, ya que no influyen en la característica que lo distinguen, pero si le dan una mayor elegancia y toque de profesionalidad que le agradan a el usuario, dándole un poco de mayor colorido e individualidad al elemento de interface, separándolo de lo que se encontraba anteriormente en la pantalla.

3.6.- MÉTODOS DE AYUDA EN LOS MENÚS

Las pantallas de ayuda son una de las más útiles características que pueden ser agregadas a una aplicación; con ellas se despliegan información sobre la función o módulo que se va a ejecutar, esto es de gran importancia para los usuarios inexpertos o novatos que empiezan a utilizar el programa, además de representar una buena fuente de consulta para los usuarios expertos que ignoren la función que se va a realizar. Básicamente existen 3 métodos de ayuda en el uso de los menús, los cuales pueden identificarse muy fácilmente como se observa a continuación :

- **POR MEDIO DE UNA OPCIÓN EN EL MENÚ :**

  Se caracteriza debido a que aparece como una opción en el menú, de manera que el usuario pueda posicionar el cursor en ella, presionar la tecla <ENTER> y entonces se desplegará una ventana con información sobre las opciones que se despliegan y uso del sistema. Este método es muy utilizado hoy en día para que los usuarios inexpertos que empiezan a manejar el programa tengan un apoyo con el contenido de esta ventana y guiarlos de la mejor manera posible con su uso y funciones.

- **POR MEDIO DE UNA TECLA ACTIVA :**

  Se le llama tecla activa, a aquella que podemos presionar en cualquier instante y siempre realizará alguna función o tarea que tenga asociada; de esta forma, podemos tener alguna tecla activa en nuestro programa (como F1 o ALT-H) para el

desplegado de una ventana de ayuda para auxiliarlo en el problema o duda que pudiera tener en el programa. Sobre este método de ayuda podemos tener dos opciones o formas de auxiliar al usuario como puede ser :

- **AYUDA GENERALIZADA** : Con esta forma, siempre que se presione la tecla activa, se mostrará la misma ventana de ayuda para todas las opciones del menú; esta ventana contendrá información sobre el manejo del sistema y las funciones de todas las opciones que se presentan en el menú.

- **AYUDA EN LINEA** : Esta forma es de las mas generalizadas hoy en día en muchos de los sistemas comerciales, su característica principal consiste en el desplegado de una ventana de ayuda especial para cada una de las opciones que contiene el menú, de manera que dependiendo de la posición en que se encuentre el cursor o video inverso, se desplegara una ventana con información sobre la función y acción que realizará.

- AYUDA INCRUSTADA :

Se le denomina de esta manera debido a que se le presenta a el usuario una pantalla de ayuda, en la cual aparecen algunas palabras con características diferentes, generalmente se representan con un color diferente, si el usuario coloca el cursor en alguna de estas palabras y presiona ENTER, se despliega otra ventana de ayuda relacionada con la palabra donde posicionó el cursor, por ejemplo :

```
Los métodos de selección en los menús pueden
ser de tres tipos diferentes

-    Con las teclas de flechas
-    Con teclas activas
-    Con el ratón

Esto puede utilizarse para los diferentes tipos
de menús.
```

De esta forma, si el usuario coloca el cursor en la palabra "menús", y presiona <ENTER>, se desplegará otra pantalla de ayuda relacionada con este tópico, facilitándole en gran medida la búsqueda de temas relacionados con ella. Algunos autores, catalogan este tipo de interface dentro de los tipos de menús, llamándolo **"MENÚS INCRUSTADOS"**[13], posiblemente en un principio esta era la función de este elemento de interface, sin embargo, hoy en día su uso mas generalizado puede apreciarse dentro de las pantallas de ayuda de muchos sistemas, las cuales han evolucionado en forma considerable.

Muchos usuarios prefieren usar alguna aplicación la cual tenga características de ayuda en línea debido a las ventajas que puede tener con ella, ya que con este método puede encontrar solución más fácilmente a los problemas que vayan surgiendo conforme va utilizando el sistema, además en muchas ocasiones, los manuales impresos no son concebidos como parte del producto o sistema, teniendo como resultados manuales incompletos o superfluos en los cuales la información esta incompleta, desordenada, o simplemente no se cuenta con los manuales; en cambio, los métodos de ayuda en línea representan una mayor rapidez en la búsqueda de la información o consulta sobre los comandos u opciones del menú, lo cual influye en la productividad, además, los usuarios perciben estos métodos de ayuda en línea como parte del producto o sistema.

Hoy en día un método de ayuda ha dejado de ser una simple presentación de pantallas de texto, se trata de proporcionar al usuario la información que desea en forma rápida y eficiente, así como de algunos temas relacionados con ella; ahora los procedimientos de ayuda suelen ser grandes sistemas manejadores de información, los cuales incluyen todos los tipos de ayuda mencionados anteriormente con el fin de brindar un mejor apoyo al usuario, este concepto lo denominan algunos autores como **HYPERTEXTO**

---

[13] Shneiderman Ben. Designing the User Interface: Stategies for Effective Human-Computer Interaction. Editorial Addison-Wesley Publishing Company. Segunda Edición 1990.

o **HYPERMEDIA**[14], el cual aplican a un sistema manejador de información con capacidad de búsqueda de alta velocidad, y puede contener texto, gráficos, audio, video, ejemplos y otras formas de datos, los cuales pueden ser mostrados al usuario para un mayor entendimiento del sistema, su uso y aplicaciones.

En cuanto al desarrollo del contenido de las pantallas de ayuda, muchos sistemas las generan dentro de la aplicación misma, es decir, cuando se trata de desplegar una ayuda, el programa contiene una serie de sentencias que despliegan un mensaje especifico, de manera que si el programa contiene un total de 10 pantallas de ayuda, estas se encuentran contenidas dentro del programa con sus respectivas sentencias; este método no es muy recomendable, ya que influye en el tamaño del programa ejecutable y además, si se desea modificar alguna de ellas, se tendrá que re-compilar el programa; por esto se sugiere crear una sola rutina que muestre las pantallas de ayuda, y el contenido de estas se coloque en archivos independientes de manera que puedan ser fácilmente modificados se así se desea.

3.7.- IMPLEMENTACIÓN DE LAS RUTINAS.

Para implementar las rutinas que realizaran la función de desplegar los menús, primeramente debemos de definir cuales van a ser las características que incluiremos en él :

- Mostrará todas las opciones que se desean incluir encerradas en un recuadro, el cual se colocara en la posición de pantalla que indique el usuario.

[14] Para una mayor información consultar : Designing & Writing On Line Documentation, Help Files to Hypertext. Horton k. William. Editorial. John Wiley and Son. Primera Edición 1990.

- Colocara una "SOMBRA" alrededor del recuadro, para que el usuario lo identifique como una sola entidad.

- Para detectar cual va a ser la letra que asocie a la selección de cada una de las opciones, utilizaremos el carácter "^", de forma que el siguiente carácter definirá cual tecla va a asociar a esta opción; por ejemplo, la cadena "^Abrir", representara una opción de algún menú, cuya tecla asociada será "A", si presionamos esta tecla, se seleccionara inmediatamente esta opción.

- Los métodos de selección que utilizará serán :
  - Por medio de una tecla asociada a la opción
  - Por medio de las teclas de flechas
  - Con el cursor del ratón

A continuación listaremos una serie de funciones indispensables para crear la rutina de este elemento de interface:

Función : *coloca_caracter*(int x, int y, char *caracter*, int *color1*, int *color2*, int *veces*)

Parámetros :

x,y	:	Columna y renglón donde se desea colocar el carácter
Caracter	:	Carácter a desplegar.
Color1	:	Color del carácter.
Color2	:	Color de fondo.
Veces	:	Numero de veces que se escribirá.

Acción : Coloca un carácter en la posición indicada un numero determinado de veces, al llegar al final del renglón, automáticamente pasa al siguiente; su importancia radica en la colocación del carácter con sus atributos de color tanto del carácter mismo como del fondo; un aspecto importante que hay que recalcar, es que hace uso de una variable global denominada *MODO*, el cual puede tener dos valores:

0 = Indica Modo Texto
1 = Indica Modo Gráfico

En modo texto, esta función utiliza el direccionamiento a la memoria de video para obtener una mayor rapidez en su desplegado; también puede utilizarse la interrupción 10, función 9 (ESCRIBIR CARÁCTER Y ATRIBUTO) para obtener el mismo resultado, con la desventaja de una ser un poco mas lento, esta función necesita los siguientes Parámetros en los registros del microprocesador :

REGISTROS	PARÁMETROS
AH	09h (función a utilizar)
AL	Carácter ASCII a desplegar
BH	Página de presentación Visual
BL	Byte de atributo del carácter
CX	Número de caracteres a desplegar

El byte de atributo del carácter se encuentra formado por el color del carácter y el color de fondo, los códigos en decimal para los colores validos, ya se han definido en el capitulo 2 sección 2.2.

En modo gráfico hace uso de las librería estandard de TURBO C++ para gráficos.

Valor de retorno: Ninguno

Ejemplo: Las siguientes dos instrucciones se pueden utilizar para colocar la pantalla con un color de fondo azul.

```
MODO=0;
coloca_caracter(1,1,' ',0,16,200);
```

Función : *escribe_video*(int *x*, int *y*, char *p, int *atrib*, int *color1*, int *color2*)

Parámetros :

x,y	:	Columna y renglón de la Posición en pantalla.
p	:	Cadena a desplegar
atrib	:	Color de fondo
color1	:	Color del carácter
color2	:	Color del carácter especial

Acción : Desplegara una cadena en la Posición dada por las coordenadas X y Y de la pantalla, también, se usa para desplegar las opciones de los menús, utiliza para su funcionamiento la función *coloca_caracter*(); para desplegar el carácter especial con atributos diferentes se usara el carácter especial "^", de esta forma, una cadena "^Archivo", desplegará la letra "A" con los atributos designados para el carácter especial que asociara a la opción de cada menú.

Valor de retorno : Esta función devolverá la letra que se asociará al menú, es decir, retornará la letra que siga al carácter "^".

Ejemplo: la siguiente instrucción escribe el texto "PRESIONE ENTER" en la posición x,y (5,19), con las siguientes características:
Color de fondo = 48
Color del carácter = 1
Color del carácter especial = 1 (en este caso no hay)

```
escribe_video(5,19,"PRESIONE ENTER",48,1,1);
```

Función : *coloca_sombra*(int *longitud*, int *x*, int *y*)

Parámetros :

longitud	:	Numero de caracteres a modificar.
x,y	:	Columna y Renglón de la pantalla.

Acción : Esta función es utilizada para colocar el efecto de sombra, su funcionamiento se basa en el uso de la interrupción 10 función 8 y en el uso de la función ***coloca_caracter()***:

- La función 8 es utilizada para leer el carácter que se encuentra en la posición X y Y, los Parámetros que necesita son:

REGISTROS	PARÁMETROS
AH	08h (función a utilizar)
BH	Página de presentación Visual

Esta función retorna el carácter y su atributo que se encuentra en la posición actual del cursor.

- Posteriormente, se utiliza la función ***coloca_caracter()*** para colocar nuevamente el carácter en su posición original, pero modificando el atributo del color de fondo, el cual ahora será negro, dando la apariencia de una "sombra".

Valor de retorno : Ninguno.

Función : ***marco***(int *x1*, int *y1*, int *x2*, int *y2*, int *fondo*, int *caracter*, int *sombra*, char *mens*[])

Parámetros :

x1,y1	:	Columna y Renglón de la esquina superior izquierda del marco.
x2,y2	:	Columna y Renglón de la esquina inferior derecha del marco.
fondo	:	Color de fondo del recuadro
caracter	:	Color de los caracteres del limite del recuadro
sombra	:	Bandera para la colocación de la sombra

0 = No se coloca sombra
1 = Si se coloca sombra

mens : Un titulo que será colocado en la parte superior del recuadro

Acción : Colocará un recuadro definido por dos coordenadas, las cuales serán el limite superior izquierdo y el limite inferior derecho, el cual será colocado con los atributos que se definan. Además, tendrá la posibilidad de colocar un título en la parte superior del recuadro, en caso de no desearlo, bastará con colocar una cadena vacía; su funcionamiento de basa en el uso de las funciones anteriores.

Valor de retorno : Ninguno.

Ejemplo: La siguiente instrucción coloca un marco dado por dos coordenadas (10,5) y (70,10), con los atributos de:
color de fondo = 16 (Azul)
Color de carácter = 0 (Negro)
Sombra = 1 (Si se despliega el efecto de sombra)
titulo = "MENÚ PRINCIPAL"

```
marco(10,5,70,10,16,0,1,"MENÚ PRINCIPAL");
```

Con todas estas funciones ya estamos en condiciones para desplegar el menú, sin embargo, todavía nos hace falta la forma de leer la respuesta por parte del usuario, recordemos que se incluirían 3 formas principales de lectura:

- Con el uso de teclas rápidas
- Con el uso de las teclas de flechas
- Con el uso del ratón

Función : *respuesta_hor*(int *x*[], int *y*, int *contador*, char **menu*, char **teclas*, int *norm*, int *inv*, int *c1*, int *c2*, int *raton*[][4]);

Parámetros :

x	:	Arreglo de posiciones en x.
y	:	Posición en Y de las opciones
contador	:	Número de opciones disponibles
menu	:	Un arreglo bidimensional con las opciones
teclas	:	Las teclas activas de cada una de las opciones
norm	:	El fondo de video normal
inv	:	El color del fondo de video inverso, para mostrar la posición del cursor.
c1	:	El color del carácter normal
c2	:	El color del carácter especial
raton	:	Un arreglo bidimensional con las posiciones validas donde puede ser leída una opción con el cursor del ratón.

Acción : Se encarga de leer la decisión del usuario para un menú horizontal, mediante los tres métodos mencionados anteriormente, utiliza la función *bioskey*() de la utilería de TURBO C para obtener el código de 16 bits generado por el teclado, y así detectar fácilmente cual tecla se ha presionado, en caso de ser un carácter del teclado alfabético, checa si se encuentra dentro de la cadena de las teclas asociadas a la opción, si es alguna de las teclas de flechas, mueve el cursor hacia la siguiente posición de acuerdo a la tecla, además, mantiene un control sobre los botones del mouse para detectar cuando se ha presionado alguno de ellos, y verifica si la posición corresponde a alguna opción.

Valor de retorno : Un valor asociado con la decisión del usuario, de la forma siguiente:

-1	->	Si se presiona la Tecla ESC.
0	->	Si se eligió la primera opción
1	->	Para la segunda opción
2	->	Para la tercera opción y así sucesivamente.

Función : *respuesta_ver*(int *x*, int *y*, int *contador*, char **menu*, char **teclas*, int *norm*, int *inv*, int *color1*, int *color2*, int *raton*[][4])

Parámetros :

x	:	Posición en X de la primera opción
y	:	Posición en Y de la primera opción
contador	:	Número de opciones disponibles
menu	:	Un arreglo bidimensional con las opciones
teclas	:	Las teclas activas de cada una de las opciones
norm	:	El fondo de video normal
inv	:	El color del fondo de video inverso, para mostrar la posición del cursor.
color1	:	El color del carácter normal
color2	:	El color del carácter especial
raton	:	un arreglo bidimensional con las posiciones validas donde puede ser leída una opción con el cursor del ratón.

Acción : Obtiene la decisión del usuario para un menú vertical, mediante los tres métodos mencionados anteriormente, su funcionamiento es muy similar a la función anterior, la diferencia radica en el tipo de menú en que es utilizada, ya que en la función anterior, el cursor se mueve en un solo renglón, y en ésta su movimiento es en varios de ellos.

Valor de retorno : Un valor asociado con la decisión del usuario, de la forma siguiente:

-1	->	Si se presiona la Tecla ESC.
0	->	si se eligió la primera opción
1	->	Para la segunda opción
2	->	Para la tercera opción y así sucesivamente.

Función : *menu_hor*(char **menu*, int *x*, int *y*, int *fondo*, int *inverso*, int *color1*, int *color2*, char *titulo*[], int *sombra*)

Parámetros :

menu	:	arreglo bidimensional con cada una de las opciones del menú, la ultima opción deberá de ser el carácter nulo ('\0').
x	:	Posición en X de la primera opción
y	:	Posición en Y de la primera opción
fondo	:	Color del fondo
inverso	:	Color de fondo para el cursor
color1	:	Color del carácter normal
color2	:	Color del carácter asociado a la opción.
titulo	:	Titulo que puede llevar el menú
sombra	:	Desplegado del efecto de sombra 0 = No se despliega sombra 1 = Si se despliega sombra

Acción : Hace uso de todas las funciones anteriores, su tarea principal consiste en la colocación del menú en la posición que se defina y leer la decisión del usuario.

Valor de retorno : Un valor asociado con la decisión del usuario, de la forma siguiente:

-1	->	Si se presiona la Tecla ESC.
0	->	Si se eligió la primera opción
1	->	Para la segunda opción
2	->	Para la tercera opción y así sucesivamente.

Función : *menu_ver*(char **menu*, int *x*, int *y*, int *fondo*, int *inverso*, int *c*, int *c2*, char *titulo*[], int *sombra*)

Parámetros :

menu	:	arreglo bidimensional con cada una de las opciones del menú, la ultima opción deberá de ser el carácter nulo ('\0').
x	:	Posición en X de la primera opción
y	:	Posición en Y de la primera opción
fondo	:	Color del fondo
inverso	:	Color de fondo para el cursor
c	:	Color del carácter normal
c2	:	Color del carácter asociado a la opción.

titulo : Titulo que puede llevar el menú
sombra : Desplegado del efecto de sombra
0 = No se despliega sombra
1 = Si se despliega sombra

Acción : Su funcionamiento es muy similar a la función menu_hor, su tarea principal consiste en la colocación de un menú en forma vertical y obtener la lectura del usuario mediante los métodos definidos anteriormente, al igual que la función anterior (***menu_hor***()) hace uso de todas las funciones explicadas anteriormente.

Valor de retorno : Un valor asociado con la decisión del usuario, de la forma siguiente:

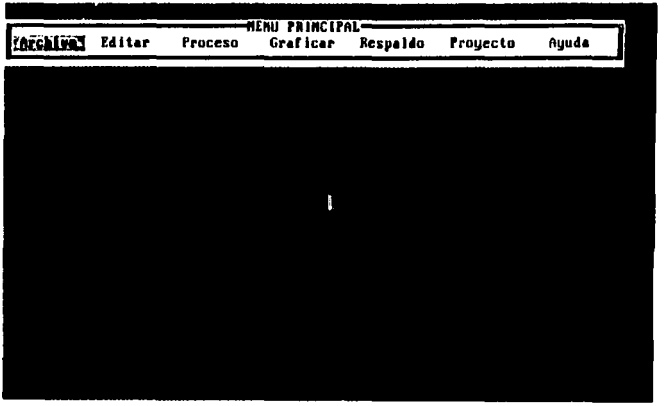
-1 -> Si se presiona la Tecla ESC.
0 -> Si se eligió la primera opción
1 -> Para la segunda opción
2 -> Para la tercera opción y así sucesivamente.

En el Apéndice I puede encontrase el código de todas las funciones definidas en esta tesis colocadas en una librería de código fuente, la cual será llamada "**usuario.c**", esta librería tendrá que ser incluida en nuestros programas mediante la directiva **#include**; a continuación podrán encontrase un ejemplo de como utilizar la función, y al final, en el APÉNDICE B podrá encontrarse un ejemplo de aplicación, con esto se logra la meta que nos habíamos propuesto al inicio de este capítulo.

Ejemplo para un menú HORIZONTAL

```
/*************************************************************************
 *   Ejemplo sobre la utilización de la función menu_hor() para la       *
 *   colocación de un menu en forma Horizontal                           *
 *     Para la selección de  algún modo gráfico  es necesario que se     *
 *     encuentren en el mismo directorio el controlador EGAVGA.BGI.      *
 *************************************************************************/
#include <conio.h>
#include <bios.h>
#include <stdlib.h>
#include <stdio.h>
#include "usuario.c"

void main()
 {
  int eleccion,modo='a';
  char *menup[]={
    " ^Archivo ",  " ^Editar  ",   " ^Proceso  ",  " ^Graficar ",
    " ^Respaldo ", " P^royecto ",  "  ^Ayuda  ",   "\0"};
  clrscr();
  printf("\nMODO DE VIDEO A UTILIZAR : ");
  printf("\n   0=TEXTO ");
  printf("\n   1=GRAFICO RESOLUCION MEDIA");
  printf("\n   2=GRAFICO ALTA RESOLUCION ");
  printf("\n   CUAL ES SU OPCION : ? ");
  while(modo!='0' && modo!='1' && modo!='2') modo=getch();
  switch(modo)
   {
    case '0': coloca_caracter(1,1,176,7,16,2000); raton(); break;
    case '1': inicializa(0); raton_grafico(); break;
    case '2': inicializa(1); raton_grafico(); break;
   }
  muestra_cursor();
  eleccion=menu_hor(menup,1,2,112,32,0,4,"MENU PRINCIPAL",1);
  switch (eleccion)
   {
    case -1:
         escribe_video(20,23,"PRESIONO LA TECLA ESCAPE",16,15,15);
         break;
    case 0:
         escribe_video(20,23,"SELECCIONA LA OPCION 1 ARCHIVO",16,15,15);
         break;
    case 1:
         escribe_video(20,23,"SELECCIONO LA OPCION 2 EDITAR",16,15,15);
```

```
        break;
    case 2:
        escribe_video(20,23,"SELECCIONO LA OPCION 3 PROCESO",16,15,15);
        break;
    case 3:
        escribe_video(20,23,"SELECCIONO LA OPCION 4 GRAFICAR",16,15,15);
        break;
    case 4:
        escribe_video(20,23,"SELECCIONO LA OPCION 5 RESPALDO",16,15,15);
        break;
    case 5:
        escribe_video(20,23,"SELECCIONO LA OPCION 6 PROYECTO",16,15,15);
        break;
    case 6:
        escribe_video(20,23,"SELECCIONO LA OPCION 6 AYUDA",16,15,15);
        break;
   }
 bioskey(0);
 if(modo=='1' || modo=='2') termina();
 clrscr();
}
```

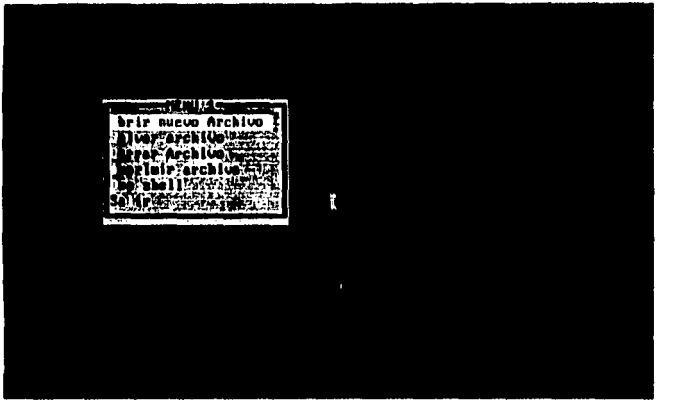

MENU PRINCIPAL
Archivo
Editar
Proceso
Graficar
Respaldo
Proyecto
Ayuda


MENU PRINCIPAL
Archivo
Editar
Proceso
Graficar
Respaldo
Proyecto
Ayuda

Ejemplo para un MENU VERTICAL

```
/*************************************************************************
 *   Ejemplo sobre la utilización de la función menu_ver() para la       *
 *   colocación de un menu en forma Vertical.                            *
 *   Para la seleccion de algún  modo gráfico es necesario que  se       *
 *   encuentren en el mismo directorio el controlador EGAVGA.BGI.        *
 *************************************************************************/
#include <conio.h>
#include <bios.h>
#include <stdlib.h>
#include <stdio.h>
#include "usuario.c"

void main()
 {
  int eleccion,modo;
  char *menu1[]={
     "^Abrir nuevo Archivo ",   "^Salvar archivo      ",
     "^Cerrar Archivo      ",   "^Imprimir archivo    ",
     "^Dos shell           ",   "Sa^lir               ",  "\0") ;
  clrscr();
  printf("\nMODO DE VIDEO A UTILIZAR : ");
  printf("\n   0=TEXTO ");
  printf("\n   1=GRAFICO RESOLUCION MEDIA");
  printf("\n   2=GRAFICO ALTA RESOLUCION ");
  printf("\n   CUAL ES SU OPCION : ? ");
  while(modo!='0' && modo!='1' && modo!='2') modo=getch();
  switch(modo)
   {
    case '0': coloca_caracter(1,1,176,7,16,2000); raton(); break;
    case '1': inicializa(0); raton_grafico(); break;
    case '2': inicializa(1); raton_grafico(); break;
   }
  muestra_cursor();
  eleccion=menu_ver(menu1,13,7,48,112,0,10,"MENU 1",1);
  switch (eleccion)
    {
     case -1:
         escribe_video(20,23,"PRESIONO LA TECLA ESCAPE",16,15,15);
         break;
     case 0:
```

```
            escribe_video(20,23,"SELECCIONA LA OPCION 1 ABRIR ARCHIVO",16,15,15);
            break;
        case 1:
            escribe_video(20,23,"SELECCIONO LA OPCION 2 SALVAR ARCHIVO",16,15,15);
            break;
        case 2:
            escribe_video(20,23,"SELECCIONO LA OPCION 3 CERRAR ARCHIVO",16,15,15);
            break;
        case 3:
            escribe_video(20,23,"SELECCIONO LA OPCION 4 IMPRIMIR ARCHIVO",16,15,15);
            break;
        case 4:
            escribe_video(20,23,"SELECCIONO LA OPCION 5 DOS SHELL",16,15,15);
            break;
        case 5:
            escribe_video(20,23,"SELECCIONO LA OPCION 6 SALIR",16,15,15);
            break;
    }
  bioskey(0);
  if(modo=='1' || modo=='2') termina();
  clrscr();
}
```


Abrir nuevo Archivo


Abrir nuevo Archivo

CAPITULO IV

MENSAJES DEL SISTEMA.

4.1.- OBJETIVOS DEL CAPITULO

Los objetivos principales de este capítulo se pueden sintetizar en los siguientes incisos:

a) Introducir al lector en la definición y características de los tipos de mensajes que se pueden desplegar durante la ejecución de algún programa.

b) Destacar la importancia de su utilización.

c) Crear las funciones que desplieguen los tipos de mensajes mas característicos e importantes.

4.2.- ¿ QUE ES UN MENSAJE DEL SISTEMA ?

Un mensaje representa una forma de establecer una comunicación entre el usuario y el sistema, su función principal consiste en el desplegado de una serie de textos dentro de una caja o recuadro los cuales definen el aviso que se desea comunicar al usuario; existen algunas variaciones en las cuales el sistema solicita una decisión; algunos autores lo llaman **"CAJAS DE DECISIÓN"** y lo toman como otro tipo de elemento de interface, sin embargo, para propósitos de esta tesis lo incluiremos en esta sección; el desplegado de este mensaje es para indicar al usuario algún aviso de manera que este enterado de lo que se esta ejecutando o de alguna acción que va a realizar; puede utilizarse en varios casos como pueden ser:

- Presentar los derechos de autor del programa

- Indicar que esta ejecutando algún proceso o la finalización del mismo.

- Señalar que ha sucedido algún error o la falta de datos.

- Pedir o confirmar alguna decisión al usuario sobre alguna acción que va a tomar el sistema.

La finalidad de un mensaje depende básicamente del uso que se le quiera dar, de esta forma, pueden tenerse dos alternativas :

- Mostrar solamente algún texto, el cual puede ser indiferente para el usuario, y no causa ningún problema en el funcionamiento del programa si se ignora; en este caso, su finalidad es puramente informativa; por ejemplo: los derechos de autor del programa o un mensaje de ayuda.

- Mostrar algún texto, pero en donde se requiere que el usuario ponga su atención, debido a que se ha detectado algún error, falla o anomalía, los cuales pueden causar la detención del proceso; debido a este motivo, el desplegado de estos mensajes son en tonalidades fuertes (como el color rojo), de manera que el usuario pueda identificarlo fácilmente.

Algunos sistemas generan algunos sonidos cuando ocurre un error; los cuales pueden ser útiles cuando el mensaje pueda pasar inadvertido para el operador; sin embargo, también presenta algunas desventajas, ya que en algunas ocasiones resulta embarazoso si hay más personas en el cuarto; y en otras, resulta molesto su uso exagerado, por esto se sugiere que el uso de las señales de audio pueda estar bajo el control del usuario[15], y que este decida si desea que estén activadas o no.

[15] Shneiderman Ben. Designing the User interface. Obra citada.

Para algunos lectores este capítulo puede parecer muy sencillo, ya que este tema puede ser considerado fácil o de poca importancia, ya que muchos programadores utilizan este elemento de interface en su forma más simple, la cual consiste en el desplegado de una pequeño mensaje un una porción de la pantalla donde no contenga información, esperando a que el usuario presione alguna tecla para posteriormente sobre-escribir líneas en blanco; sin embargo, considerando que se van a incluir en este capítulo las "CAJAS DE DIALOGO", este capítulo presenta un poco mayor de complejidad incluso que en el capítulo anterior debido a los siguientes aspectos:

- Debe de tomarse en cuenta que el mensaje se desplegara en alguna parte de la pantalla la cual puede tener información útil para el usuario, teniendo que almacenar su contenido por medio de algún método, evitando que se pierdan estos datos.

- El desplegado de los mensajes puede contener una o mas opciones de las cuales el usuario puede seleccionar alguna de ellas, bajo esta panóramica, puede considerarse que debe de tener integrado elementos idénticos a los de un menú horizontal, ya que estas opciones deberán de ser desplegadas en la pantalla y tener la capacidad de leer la decisión del usuario por los principales métodos de lectura.

- Por último, una vez que el usuario ha tomado alguna decisión, deberá de restaurarse la pantalla original donde se coloco el mensaje.

En cuanto a los aspectos que deben de tomarse en cuenta en el diseño de los mensajes, Shneiderman[16] menciona algunas de estas características, entre las mas importantes podemos citar las siguientes:

---

16 Shneiderman Ben. Designing the User interface. Obra citada. Páginas 308-313.

- **Sea tan especifico y preciso como sea posible:** Mencione el problema que ha sucedido en términos que el usuario entienda, evite la terminología vaga u obscuros códigos internos; use nombres de variables y conceptos conocidos por el usuario.

- **Sea constructivo indicando que se necesita hacer:** Ponga a el usuario en el control de la situación, y proporcione a el usuario la suficiente información para realizar alguna acción o tomar alguna decisión.

- **Use un tono positivo:** Indique que debe hacerse en lugar de condenar el error cometido por el usuario.

4.3.- CARACTERÍSTICAS DE LOS MENSAJES.

Los tipos de mensajes que se pueden desplegar, independientemente del uso que se le asigne, tienen asociadas una serie de características que pueden ser fácilmente identificadas como las siguientes:

- Antes de colocarlo, se almacena la información que hay en ese momento en la posición donde se colocará, evitando con esto eliminar información útil para el usuario.

- La parte principal de este elemento de interface lo constituye una serie de textos los cuales conforman el aviso que se desea presentar a el usuario; para los modos de video gráficos, se suele agregarle algunas figuras o dibujos pequeños, de manera que el usuario pueda asociar de una manera sencilla el tipo y finalidad del mensaje.

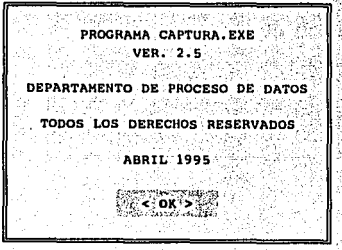
- El mensaje puede estar dentro de un recuadro, de manera que sea identificado como un elemento nuevo e

independiente de la información anterior; este recuadro junto con las cadenas de caracteres que conforman el aviso se representan con colores diferentes al del contenido de la información que se tiene desplegada, esto con el fin de evitar que se confundan los datos y que el usuario identifique rápidamente su aparición.

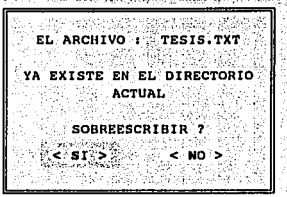
- El recuadro que se despliega puede ir acompañado del efecto de "sombra", es decir, los caracteres que están en la parte inferior y a la derecha de él, se les cambia su atributo del color de fondo, por un color obscuro, dando la apariencia de la proyección de una sombra por parte del mensaje sobre la pantalla, obteniendo con esto una mejor presentación visual y además, el usuario puede asociar más fácilmente esto como un elemento independiente y diferente a la información que tenía desplegada en la pantalla.

- Puede ir acompañado por una serie de opciones en forma de un menú horizontal pequeño, el cual puede tener desde una opción hasta tres o cuatro máximo, de las cuales el usuario puede seleccionar alguna de ellas:

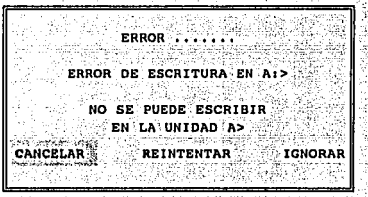
  - Cuando se representa una sola opción, esta suele ser algún mensaje para que el usuario indique que ya ha terminado de leer su contenido, por ejemplo:



- Cuando se representan dos opciones generalmente se utiliza cuando ha sucedido alguna falla, o se ha encontrado algún obstáculo para realizar alguna operación o proceso, por ejemplo:



- Se utilizan tres o cuatro opciones cuando se ha encontrado alguna falla, sin embargo, Se tiene una o dos alternativas para continuar con la tarea que se estaba ejecutando en ese momento, por ejemplo:



Estas opciones se colocan en la parte inferior del mensaje, en modo gráfico, pueden ir acompañadas de un pequeño dibujo que asocie a cada una de ellas.

- Por último, al cumplir su función principal, se elimina de la pantalla y se restaura la información que había donde fue colocado, con el fin de evitar la perdida de la información que tenía el usuario.

En el desarrollo de nuestras rutinas, incluiremos todas estas características con el fin de crear buenos elementos de interface.

4.4.- TIPOS DE MENSAJES.

Existen básicamente tres formas de mensajes, independientemente de las características secundarias que pueden tener y usos o aplicaciones que se les puede dar, estas son :

- **LÍNEAS DE MENSAJES :**

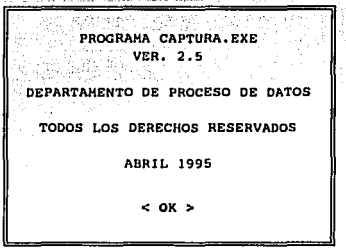
  Se caracterizan por carecer de la mayoría de características que se le pueden asociar a este tipo de elemento de interface; solamente despliegan una cadena de caracteres en una parte de la pantalla especialmente reservada para su uso, y al desaparecer, solamente se coloca una línea de espacios para eliminar la cadena anterior; debido a su fácil implementación, es muy utilizada por los programadores, sin embargo, no es la mejor forma de implementarlos, ya que para un usuario que no este atento a toda la información que se despliega puede pasar inadvertido, por este motivo, el mensaje tiene que resaltar de alguna forma con el fin de llamar la atención del usuario.

```
ERROR EN LA ENTRADA DE DATOS:
  EL VALOR DE LA VARIABLE SOBREPASA EL RANGO
```

- **MENSAJES EN MODO TEXTO :**

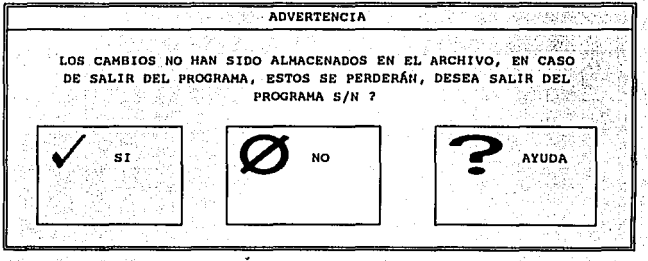
  Su principal característica consiste en el desplegado de un aviso dentro de un recuadro para que el usuario observe su contenido; este recuadro es colocado generalmente en la parte central de la pantalla, no importando si hay información útil para el usuario, este mensaje desaparece una vez que el usuario presione alguna

tecla como señal de que ha terminado de leerlo, una vez que se cumple la tarea, la información que se encontraba es restaurada.



- MENSAJES GRÁFICOS :

Aprovechan las ventajas que se pueden tener en estos modos de video, teniendo la posibilidad de incluir dibujos que asocian al tipo de mensaje que se le esta mostrando a el usuario, además, se pueden incluir algunas figuras que asocien a cada una de las opciones que se despliegan, esto con el fin de que el usuario identifique su finalidad y uso de una manera fácil y rápida.



También pueden clasificarse a los mensajes en cuanto al uso que se le puede dar, entre los mas comunes podemos mencionar los siguientes:

- **MENSAJES INFORMATIVOS :**

  Su principal característica consiste en el desplegado de un aviso puramente informativo para que el usuario observe su contenido, este mensaje desaparece una vez que el usuario presione alguna tecla como señal de que ha terminado de leerlo, el tiempo de permanece este mensaje en la pantalla, es el tiempo desde su desplegado hasta que el usuario presione la tecla, un ejemplo puede ser el desplegado de los derechos de autor.

```
REPORTE TERMINADO, PRESIONE ENTER PARA CONTINUAR
```

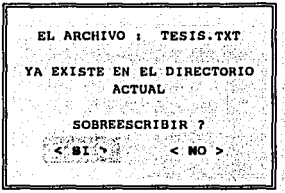
- **MENSAJES DE PROCESO :**

  Se caracteriza por el desplegado de un aviso sobre la ejecución de un proceso, el cual es colocado cuando se inicio el proceso y permanece hasta que termina, su principal función consiste en indicar al usuario que se esta realizando alguna tarea, para evitar que piense que el programa tuvo algún fallo y se interrumpió el sistema.

```
PROCESANDO REGISTRO :      25 DE 250
```

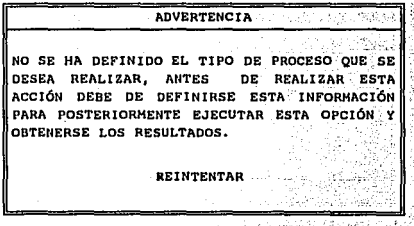
- **MENSAJES DE DECISIÓN :**

  Su característica principal consiste en que además de desplegar un mensaje para que el usuario se entere de alguna anomalía o acción que va a realizar, pide alguna decisión al usuario para poder ejecutarla, interrumpiendo la ejecución del programa hasta que el usuario tome y seleccione alguna de las opciones que se presentan.



- MENSAJES DE ALERTA O ADVERTENCIA :

Se caracteriza por que su en su contenido advierte a el usuario alguna anomalía detectada como puede ser la falta de datos, alguna variable fuera de rango, datos incoherentes, etc., los cuales puede causar un incorrecto funcionamiento del programa, alguna falla, o la no ejecución del proceso.

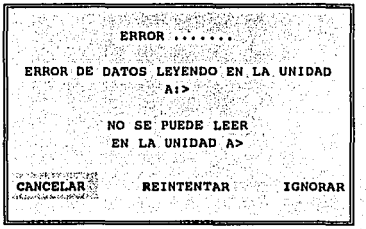


- MENSAJES DE ERROR :

Indica a el usuario que ha sucedido algún error critico, por lo cual no puede terminar el proceso que estaba realizando, se pueden tener dos casos :

- Se interrumpe el proceso de manera inevitable debido a algún error interno del programa, por ejemplo alguna división entre cero, variables no definidas, valores fuera de rango, etc.

- Se espera a que el usuario corrija la falla en caso de que esta sea física, por ejemplo el tener el drive abierto al tratar de almacenar algún archivo en diskette, la imposibilidad de almacenar algún archivo por la falta de espacio, etc.; se puede desplegar dos opciones en la pantalla para que el usuario decida que acción se va a tomar debido a la falla detectada (REINTENTAR o CANCELAR).



Estos son los tipos de mensajes más utilizados dentro de los programas, algunos de ellos pueden parecer muy similares; en efecto, algunos tipos de mensajes se utilizan para varios casos, debido a que el programador define que uso le da; por ejemplo, un mismo tipo de mensaje donde solamente cambia las cadenas de caracteres que se quieren mostrar puede ser utilizado de las siguientes formas.

- Para desplegar Derechos de autor
- Para indicar que ha sucedido algún error fatal que interrumpe la ejecución del programa.
- Para indicar alguna variable fuera de rango
- Indicar el inicio o termino de algún proceso.

4.5. - MÉTODOS DE SELECCIÓN EN LOS MENSAJES.

Tomando en cuenta las cajas de selección, las cuales habíamos mencionado anteriormente, se presentan dentro de esta una serie de opciones para que el usuario seleccione alguna de ellas, la forma de obtener la decisión del usuario es la misma que en los menús horizontales, de hecho, puede considerarse que dentro de una caja de dialogo o mensaje hay un pequeño menú horizontal integrado, el cual puede tener una, dos o mas opciones; como habíamos mencionado en el capítulo anterior, las formas más comunes de obtener la decisión del usuario son :

- Con el uso de una tecla asociada a la opción
- Por medio de las teclas de flechas y la tecla ENTER
- Con algún dispositivo apuntador como el ratón.

Para ofrecer una serie de alternativas para que el usuario utilice aquella que mas la agrade, se incluían las tres opciones de selección, de manera que pudiera utilizar aquella que le facilite la operación del programa, y en este caso no es la excepción; se incluyen estos tres métodos dentro de la selección de un mensaje.

Se presenta el caso de que solamente encierre una sola opción dentro del desplegado del mensaje, como sucede en el caso del mensaje informativo (generalmente esta opción es "OK"); en este caso, podemos considerar que tenemos un menú horizontal representado en su mínima expresión, que es con una sola opción, aún en este caso, podemos obtener la decisión del usuario con los tres métodos anteriores.

Como puede apreciarse, los métodos de selección son los mismos que en un menú horizontal, y de hecho, para el desarrollo de nuestras rutinas, se nos facilita la tarea si consideramos que el mensaje tiene integrado un menú horizontal dentro, el cual puede ser tratado como tal, y las rutinas que realizan esta tarea ya fueron generadas en el capítulo anterior.

4.6.- IMPLEMENTACIÓN DE LAS RUTINAS.

Con el fin de facilitar a los lectores la comprensión de las funciones y tratando de aprovechar aquellas que ya se han generado en el capítulo anterior, solamente se hará referencia a las nuevas funciones, así como el principio de su funcionamiento. Primeramente, para implementar las rutinas que realizaran la función de desplegar los mensajes, necesitamos definir cuales van a ser las características que incluiremos en él:

- Se guardara la información que este desplegada en la pantalla en el lugar donde se colocara el mensaje, con el fin de evitar la perdida de datos.

- El mensaje que se desea mostrar será colocado en la parte central de la pantalla, teniendo como longitud máxima, aquella longitud de la línea mas larga que lo conforma.

- Cada línea del mensaje será centrada dentro de su posición dentro del mismo.

- Todas las líneas que conforman el mensaje se encerraran dentro de un recuadro, con el fin de evitar que se confunda con la información que ya estaba desplegada.

- El recuadro tendrá la posibilidad de proyectar el efecto de sombra.

- Se desplegara una serie de opciones dentro del mensaje con el fin de incluir las cajas de dialogo; para aquellos mensajes que no sea necesario su inclusión, se tendrá que colocarlo de forma obligatorio para estandarizar la forma y parámetros que se necesitarán, en este caso puede incluirse alguna opción de manera que el usuario indique que ha terminado de leer el aviso y desea que este desaparezca, como puede ser el caso de:

- "<Ok>"
- "<Presione ENTER para Continuar>"

- Los métodos de selección que utilizará serán :

  - Por medio de una tecla asociada a la opción
  - Por medio de las teclas de flechas
  - Con el cursor del ratón

- Una vez que se ha cumplido la misión del mensaje, se procederá a colocar la información que había antes de su colocación.

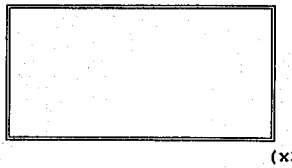
Básicamente, la mayoría de las funciones necesarias para la implementación de este tipo de elemento de interface ya se tienen generadas, solamente nos faltaría la generación de tres funciones fundamentales :

Función : *salva_pantalla*(int *x1*, int *y1*, int *x2*, int *y2*)

Parámetros :

x1	:	Valor X de la primera coordenada
y1	:	Valor Y de la Primera Coordenada
x2	:	Valor X de la segunda coordenada
y2	:	Valor Y de la segunda coordenada

Acción : Almacena la porción de la pantalla definida por el rectángulo que se forma con las dos coordenadas:



Esta función solamente opera en modo texto, estas pantallas van siendo almacenadas en un arreglo dinámico, el cual va cambiando de tamaño conforma se van almacenando mas pantallas o conforma se van extrayendo; para esto, la forma en que se extraen las pantallas utiliza el principio de una **PILA**, es decir, la última pantalla que se almacenó es la primera en salir.

El principio que basa su funcionamiento esta rutina es el acceso directo a memoria para obtener los caracteres y sus atributos que serán almacenados; también puede hacerse uso de la interrupción 10H función 8 (leer carácter y atributo), con la cual se obtienen resultados aceptables, los Parámetros que necesita son los siguientes:

REGISTROS	PARÁMETROS
AH	08h (función a utilizar)
BH	Página de presentación Visual

Esta interrupción regresa en los mismos registros del procesador el carácter y atributo ubicados en la posición donde se encuentra el cursor, de esta manera, se rastrea toda la posición donde se desplegará el mensaje para obtener la información que esta desplegada y se almacena en una arreglo unidimensional para poder regenerar la pantalla nuevamente.

Valor de retorno: Ninguno

Ejemplo : Las siguientes instrucción almacena la parte de la pantalla definida por el rectángulo que forman las coordenadas (10,6) y (70,18):

```
salva_pantalla(10,6,70,18);
```

Función : *coloca_pantalla*()

Parámetros : Ninguno.

Acción : Coloca la última pantalla almacenada con la función *salva_pantalla()*, y la coloca en su posición original; al igual que en el caso anterior, esta función solo trabaja en modo texto, utiliza el acceso directo a la memoria para colocar nuevamente los caracteres y sus atributos de color en sus posiciones originales; también puede utilizarse la interrupción 10H función 9 (escribir carácter y atributo) para obtenerse el mismo resultado, pero con la desventaja de ser un poco mas lenta, los Parámetros que necesita esta interrupción son los siguientes:

REGISTROS	PARÁMETROS
AH	09h (función a utilizar)
AL	Carácter ASCII a desplegar
BH	Página de presentación Visual
BL	Byte de atributo del carácter
CX	Número de caracteres a desplegar

Los datos de las coordenadas donde se colocaran los datos, no son necesarios, ya que estos también son almacenados en el arreglo.

Valor de retorno: Ninguno

Ejemplo : Las siguientes instrucción restaura una parte de la pantalla, la cual se había almacenado anteriormente con la Función *salva_pantalla*():

```
coloca_pantalla();
```

Función : *salva_pan_grafica*(int *x1*, int *y1*, int *x2*, int *y2*)

Parámetros :

x1	:	Valor X de la primera coordenada
y1	:	Valor Y de la Primera Coordenada
x2	:	Valor X de la segunda coordenada
y2	:	Valor Y de la segunda coordenada

Acción : Esta función realiza la misma acción que la función ***salva_pantalla()***, las únicas diferencias radican en que es utilizada para almacenar pantallas en modo gráfico, y en la forma de almacenamiento de la imagen, ya que en este caso no se hace uso de algún arreglo de memoria, debido a que en modo gráfico, no se debe de almacenar el caracter y su atributo de color, sino que se debe de almacenar cada uno de los pixeles que se encuentran dentro del área donde se desplegara el mensaje; para una pantalla de 640 x 480 pixeles, se necesitaría un arreglo de 307,200 posiciones, para una sola pantalla, lo cual es excesivo; por este motivo se hace uso de un archivo temporal, en el cual se van almacenando cada una de las imágenes, con este evitamos esta restricción; al igual que la función ***salva_pantalla()***, utiliza una estructura de **PILA** para su almacenamiento.

Valor de retorno: Ninguno

Ejemplo : Las siguientes instrucción almacena la parte de la pantalla gráfica definida por el rectángulo que forman las coordenadas (10,6) y (70,18):

```
salva_pan_grafica(10,6,70,18);
```

Función : *coloca_pan_grafica*()

Parámetros : Ninguno.

Acción : Coloca la ultima pantalla almacenada con la función ***salva_pan_grafica()***, y la coloca en su posición original; esta función solo trabaja en modo gráfico, utiliza el acceso a un archivo temporal donde fue almacenada la imagen.

Valor de retorno: Ninguno

Ejemplo : Las siguientes instrucción restaura una parte de la pantalla, la cual se había almacenado anteriormente con la Función ***salva_pan_grafica()***:

```
coloca_pan_grafica();
```

Función : *mensaje*(char **men*[],char **elige*[],int *sombra*, int *fondo*, int *inv*, int *c2*, int *c*)

Parámetros :

men	:	Lista de líneas de texto que se quiere desplegar; para finalizar, se introduce una LINEA conteniendo solamente el carácter nulo ("\0").
elige	:	Lista de opciones en el mensaje, en caso de no ser necesario, se deberá de incluir una sola opción, la cual puede ser "<^OK>", recordemos que el símbolo "^" indica que la siguiente letra representará una tecla, la cual puede ser presionada y automáticamente se seleccionara esta opción; al igual que en el parámetro anterior, la ultima opción deberá de ser una cadena con el carácter nulo ("\0").
sombra	:	Una bandera para el desplegado del efecto de sombra: 0 = No se proyecta sombra 1 = Si se proyecta sombra
fondo	:	El color de fondo que se quiere utilizar
inv	:	El color del fondo inverso para indicar la localización del cursor.
c2	:	El color de los caracteres especiales que representaran las teclas que pueden utilizarse para seleccionar alguna opción.
c	:	El color de desplegado de los caracteres y marco que formaran el mensaje.

Acción : Esta función, primeramente calcula la posición donde será colocado el mensaje, almacena la información que esta desplegada en el lugar que se utilizará para colocarlo, mediante la función *marco*() crea un recuadro, posteriormente centra y coloca las cadenas de texto que se necesitan, coloca las opciones o mini-menú horizontal que puede contener, obtienen la decisión del usuario, y por último restaura la información que se tenía en la pantalla.

Valor de retorno : Un valor asociado con la decisión del usuario, de la forma siguiente:
-1 -> Si se presiona la Tecla ESC.
0 -> si se eligió la primera opción
1 -> Para la segunda opción
2 -> Para la tercera opción
y así sucesivamente.

Función : *mensaje2*(char *men*[], int x1, int y1, int fondo, int caracter, int sombra, int opcion)

Parámetros :

men	:	linea del mensaje que se quiere desplegar.
x1,y1	:	Posición donde se quiere desplegar la linea de mensaje.
fondo	:	Color del fondo de la cadena.
caracter	:	Color de la cadena que se desea desplegar.
sombra	:	Una bandera para el desplegado del efecto de sombra: 0 = No se proyecta sombra 1 = Si se proyecta sombra
opcion	:	Representa una bandera, la cual puede ser: 0 = Almacena la imagen que hay donde se colocara el mensaje y lo coloca. 1 = coloca la ultima pantalla que fue almacenada.

Acción : Esta función coloca un mensaje en una sola linea, el cual es encerrada en un recuadro, funciona tanto en modo texto

como en modo gráfico, si la variable opcion es cero, salva la porción de la pantalla donde será almacenada y coloca el mensaje; si es uno, coloca la última pantalla almacenada.

Valor de retorno : Ninguno.

Ejemplo: las siguientes dos líneas colocan un mensaje, ejecuta una función y restaura la pantalla.

```
mensaje2(" REALIZANDO PROCESO ... ",10,10,64,15,1,0);
proceso();
mensaje2(" REALIZANDO PROCESO ... ",10,10,64,15,1,1);
```

Función : *mon_esp*(int *x*, int *y*, char **men*, int *fondo*, int *color*, int *sombra*)

Parámetros :

x,y	:	Posición de la pantalla donde se desea colocar el mensaje
men	:	Línea del mensaje que se quiere desplegar.
fondo	:	El color de fondo que se quiere utilizar
c	:	El color de desplegado de los caracteres y marco que formaran el mensaje.
sombra	:	Una bandera para el desplegado del efecto de sombra: 0 = No se proyecta sombra 1 = Si se proyecta sombra

Acción : Esta función almacena la posición de la pantalla donde será colocado el mensaje, lo coloca con los atributos indicados, y espera a que el usuario presione alguna tecla, cuando esto sucede, restaura la posición de la pantalla donde se había colocado; funciona tanto en modo texto como en modo gráfico.

Valor de retorno : Ninguno.

Ejemplo : La siguiente linea coloca un mensaje en la posición (10,10), con un color de fondo rojo(64) y del caracter blanco(15), con el efecto de sobra alrededor del marco, y espera a que el usuario presione alguna tecla para eliminarlo de la pantalla:

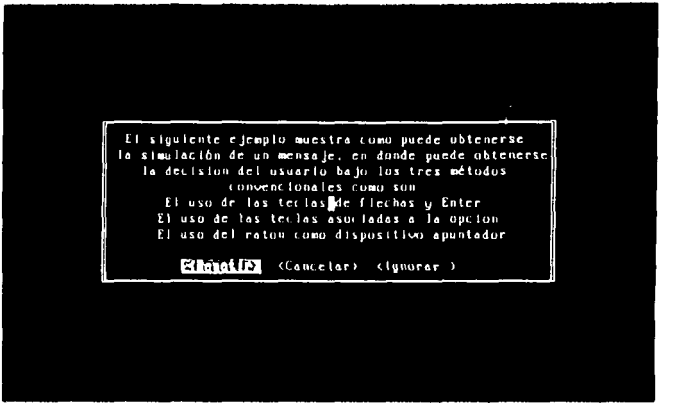
```
men_esp(10,10,"PRESIONE ENTER PARA CONTINUAR ",64,15,1);
```

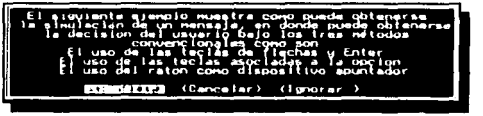
En el Apéndice I puede encontrase el código de todas las funciones definidas en esta sección, a continuación incluimos un ejemplo en donde se ilustra como deben de utilizarse estas últimas funciones, las cuales integran la mayoría de las rutinas generadas:

Ejemplo para el desplegado de un MENSAJE

```
/*****************************************************************
 *   Ejemplo sobre la utilización de la función mensaje() y de la  *
 *   función men_esp() para la colocación de mensajes en la        *
 *     pantalla.                                                   *
 *     Para la selección de  algún modo gráfico es necesario que se *
 *     encuentren en el mismo directorio el controlador EGAVGA.BGI.  *
 *****************************************************************/
#include <conio.h>
#include <stdio.h>
#include <bios.h>
#include "usuario.c"

void main()
 {
  int eleccion,modo;
  char *men[]={
    " El siguiente ejemplo muestra como puede obtenerse ",
    " la simulación de un mensaje, en donde puede obtenerse",
    " la decisión del usuario bajo los tres métodos ",
    " convencionales como son: ",
    " El uso de las teclas de flechas y Enter ",
    " El uso de las teclas asociadas a la opcion ",
    " El uso del raton como dispositivo apuntador",
    " ",
    "\0");
  char *elige[]={" <^Repetir>"," <^Cancelar>"," <^Ignorar >","\0"};
  clrscr();
  printf("\nMODO DE VIDEO A UTILIZAR : ");
  printf("\n   0=TEXTO ");
  printf("\n   1=GRAFICO RESOLUCION MEDIA");
  printf("\n   2=GRAFICO ALTA RESOLUCION ");
  printf("\n   CUAL ES SU OPCION : ? ");
  while(modo!='0' && modo!='1' && modo!='2') modo=getch();
  switch(modo)
   {
    case '0': coloca_caracter(1,1,176,7,16,2000); raton(); break;
    case '1': inicializa(0); raton_grafico(); break;
    case '2': inicializa(1); raton_grafico(); break;
   }
  muestra_cursor();
```

```
eleccion=mensaje(men,elige,1,64,32,14,15);
switch(eleccion)
 {
  case -1: men_esp(20,20," PRESIONO LA TECLA ESCAPE ",64,15,1);     break;
  case 0 : men_esp(20,20," SELECCIONO LA OPCION REPETIR ",64,15,1); break;
  case 1 : men_esp(20,20," SELECCIONO LA OPCION CANCELAR",64,15,1); break;
  case 2 : men_esp(20,20," SELECCIONO LA OPCION IGNORAR ",64,15,1); break;
 }
if(modo=='1' || modo=='2') termina();
clrscr();
}
```


El siguiente ejemplo muestra como puede obtenerse
la simulación de un mensaje, en donde puede obtenerse
la decision del usuario bajo los tres métodos
convencionales como son
El uso de las teclas de flechas y Enter
El uso de las teclas asociadas a la opcion
El uso del raton como dispositivo apuntador
<Cancelar> <Ignorar >


El siguiente ejemplo muestra como puede obtenerse
la simulación de un mensaje, en donde puede obtenerse
la decision del usuario bajo los tres métodos
convencionales como son
El uso de las teclas de flechas y Enter
El uso de las teclas asociadas a la opcion
El uso del raton como dispositivo apuntador
(Cancelar) (Ignorar)

CAPITULO V

FORMAS DE ENTRADA Y SALIDA.

5.1.- OBJETIVOS DEL CAPITULO

Entre los objetivos principales de este capítulo podemos mencionar los siguientes incisos:

a) Introducir al lector en el concepto de **"FORMA DE ENTRADA Y SALIDA DE DATOS"**, así como en los diferentes elementos de interface que la conforman.

b) Destacar la importancia que tienen el uso de las FORMAS dentro de nuestros programas.

c) Crear las funciones de los elementos mas importantes que pueden incluirse dentro de las FORMAS, y generar un ejemplo de como pueden integrarse todos ellos para su generación.

El código fuente de todas las funciones mencionada en este capítulo pueden encontrarse en el final de la tesis, en el APÉNDICE A, el cual contiene el listado del archivo usuario.c.

5.2.- DEFINICIÓN, IMPORTANCIA Y CARACTERÍSTICAS GENERALES.

Una FORMA representa en una oficina un documento con un formato fijo, en el cual se colocan ciertas variables que el usuario debe de introducir de acuerdo a sus características, necesidades, antecedentes o datos personales, para esto, la FORMA

debe de contener una serie de campos o espacios, los cuales debe de llenar el interesado de acuerdo a su información[17], por ejemplo:

- Forma de solicitud de empleo
- Forma de Pago
- Forma de inscripción
- Forma de registro de exámenes extraordinario
- Forma de pedido de material
- Forma de admisión.

La FORMA se utiliza como una herramienta para hacer que el trabajo se realice o introducir datos al sistema; siendo un documento de uso diario e indispensable en cualquier oficina; su formato ya esta definido, y al llenarla se tiene que apegar a las indicaciones y datos que se piden; los usos de esta herramienta pueden ser de diversa índole, siendo los mas comunes:

- Hacer una Solicitud
- Autorizar
- Cancelar
- Notificar
- Hacer pedidos
- Registrar
- Reportar
- Solicitar
- Demandar
- Acordar, etc.

Una "FORMA ELECTRÓNICA DE ENTRADA Y SALIDA DE DATOS" es la representación de este tipo de documento en la computadora; Shneiderman[18] define a la forma como una entidad que agrupa a un conjunto de campos con una interacción entre todos ellos; su implementación tiene dos funciones principales:

---

17 John G. y Burch Gary Grudnitski. Diseño de sistemas de información Teoría y Práctica. Editorial MEGABYTE. Primera Edición 1992 México. Páginas 204-219

18 Shneiderman Ben. Designing the User Interface. Obra citada.

- Obtener los datos de entrada que alimentaran a algún programa para la realización de un proceso o almacenar la información para su uso posterior.

- Presentar los resultados de un proceso, datos almacenados o algún tipo de reporte.

Las FORMAS básicamente se utilizan para obtener datos de cualquier tipo de una manera sistemática, rápida y sencilla, asignándole determinados campos a cada una de las opciones que se presentan y permitiéndole hacer las modificaciones necesarias sin tener que volver a colocar nuevamente todos los datos.

Al diseñar una forma electrónica de datos deben de tomarse en consideración una serie de características, las cuales evitaran conflictos posteriores con los datos del usuario, entre estas características podemos mencionar :

1.- **DEFINICIÓN DE CAMPOS** : Deben de tomarse en cuenta las variables que se van a introducir, así como su tamaño del campo y el tipo de datos que podrá aceptar, todos ellos deben de tener una posición fija dentro de la FORMA, de manera que su localización sea rápida.

2.- **VERIFICACIÓN DE LIMITES** : Se pueden establecer algunos limites para las variables con el fin de evitar la entrada de datos que excedan un determinado valor; en la mayoría de las ocasiones, esto se suele realizar limitando el lugar donde se realiza la lectura de la variable colocando solamente los campos necesarios para ella, de manera que no se pueda exceder en este espacio.

3.- **CÁLCULOS** : Se puede tener la posibilidad, en caso de ser necesario, de realizar algunos cálculos con algunos datos, y presentarlos de manera automática, disminuyendo el número de campos que tiene que llenar el usuario facilitando el trabajo.

4.- **AYUDA EN LÍNEA** : Por supuesto, no faltaba los métodos de ayuda para este elemento de interface para auxiliar a el usuario en el llenado correcto de la FORMA, generalmente se utiliza el método de ayuda en línea; en muchos sistemas combinan los métodos de ayuda en línea y ayuda incrustada, de manera que se presenta a el usuario una pantalla relacionada con la variable donde se encuentra posicionado el cursor, pero además, dentro de esta hay una serie de opciones incrustadas dentro del texto, para facilitar la búsqueda de temas relacionados.

5.- **RECUPERACIÓN O TRANSMISIÓN AUTOMÁTICA DE DATOS** : Se puede tener la capacidad de recuperar datos, en caso de que estos ya existan, para realizar modificaciones, sin tener que recurrir a su llenado total; de esta manera se le presenta a el usuario los datos anteriores para que realice las modificaciones necesarias, reduciendo el tiempo de captura.
También se puede tener la capacidad de transmitir los datos automáticamente al sistema para que estos sean procesados o almacenados, permitiendo con esto reinicializar los valores a su estado o valor original con el fin de solicitar mas datos a el usuario.

6.- **PERMITIR LA CORRECCIÓN DE ERRORES PARA CARACTERES INDIVIDUALES Y EL CAMPO COMPLETO:** Debe de facilitarse la corrección de los datos del campo permitiendo la modificación de caracteres individuales del campo.

7.- **UTILIZAR UN MOVIMIENTO DEL CURSOR ENTRE TODOS LOS CAMPOS:** Debe de utilizarse un mecanismo simple y sencillo para el movimiento del cursor, el cual debe de ser intuitivo para el usuario, esto se puede lograr utilizando las teclas de flechas para mover el cursor entre todos los campos.

8.- **DESPLEGAR MENSAJES DE ERROR PARA VALORES INACEPTABLES:** Este mensaje de error puede aparecer una vez introducido el dato erróneo; este mensaje puede indicar los valores permitidos en el campo.

Como se puede entender, básicamente una FORMA es un conjunto de elementos de interface cada uno de los cuales representa una variable, agrupados y acomodados en posiciones fijas, de manera que todos ellos en su conjunto definen la FORMA; pueden existir, aún dentro de un solo sistema varios tipos de ellas, dependiendo de las necesidades, variables y/o tipos de datos de entrada que se tengan.

5.3 VENTAJAS DE LA UTILIZACIÓN DE LAS FORMAS.

La principal importancia de este elemento de interface radica en la agrupación de una serie de variables las cuales pueden ser de diferente tipo (numéricas, alfabéticas, booleanas, etc.) dentro de un solo elemento para realizar su lectura; el cual, el usuario lo identifica como una sola unidad que tienen comprendidos todas estas variables, y lo asocia con el objeto físico (el documento), facilitándole el entendimiento y uso del programa, además de agregarle una serie de alternativas para facilitar la introducción de los datos, como puede ser la colocación de mensajes de ayuda explicando como llenar correctamente los datos y la facilidad de mover el cursor hacia todas las variables que la constituyen.

La utilización de las formas dentro de nuestros programas, puede traer como consecuencia una serie de ventajas tanto para los usuarios como para el sistema; entre ellas podemos citar:

- Permite la agrupación de varios elementos de interface, diseñados para obtener y/o mostrar datos, dentro de un solo elemento, el cual maneja la interacción entre todos ellos.

- Se facilita la lectura entre todos los datos que la componen, ya que el cursor puede trasladarse de un campo a otro con las teclas de flechas, para posicionarlo en el campo deseado.

- Al limitar o definir el número de campos para cada una de las variables, se disminuye el riesgo de obtener alguna falla en el sistema.

- En caso de existir datos, estos pueden ser colocados nuevamente para se revisión y corrección.

- El usuario asocia este elemento de interface con un documento físico, lo cual ayuda en que su entendimiento sea más fácil, y de esta manera se obtengan resultados mas rápidamente.

- Al facilitar la manera de introducir y corregir los datos que alimentaran a nuestro proceso, se disminuye el tiempo de su captura; lo cual causará un aumento en la productividad

Con esta serie de ventajas, el sistema se hace mas entendible, se facilita su operación y el usuario obtiene una serie de facilidades en la captura de datos, lo cual influye directamente con el tiempo asignado para realizar esta tarea, el cual se disminuye notablemente, obteniendo una mayor rapidez.

5.4 ELEMENTOS DE INTERFACE DENTRO DE LA FORMA.

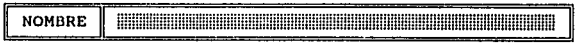
La FORMA DE ENTRADA Y/O SALIDA DE DATOS tiene integrados varios elementos de interface que lo conforman; cada uno de los cuales representa una variable, en el caso de que esta variable tenga un valor inicial, éste deberá de colocarse para que el usuario pueda observarlo y en caso necesario modificarlo.

Entre los principales elementos de interface que podemos agrupar dentro de una FORMA podemos citar a:

- CAMPOS DE TEXTO
- CAMPOS NUMÉRICOS
- BOTONES SIMPLES
- LISTA DE CONTROLES
- DESLIZADORES

De entre todos ellos, los dos primeros son los de mayor uso dentro de todas las formas, y son indispensables para obtener los datos básicos de entrada que se necesitan; los tres últimos se pueden utilizar en casos especiales; a continuación definiremos cada uno de estos elementos de interface :

- **CAMPOS DE TEXTO** : El campo de texto constituye uno de los principales elementos de mayor uso dentro de las FORMAS, se utiliza para obtener aquellos datos que pueden ser caracteres alfabéticos o alfanuméricos, siendo estos muy comunes en cualquier relación de variables que componen una FORMA; un ejemplo de este elemento de interface se muestra a continuación.



En caso de que esta variable ya posea algún valor inicial deberá colocarse dentro de los campos correspondientes; el cursor será posicionado en la primera letra, y el usuario podrá moverlo a través del campo con las teclas de flecha hacia la izquierda (←) y a la derecha (→).

- **CAMPOS NUMÉRICOS** : El campo numérico es muy similar al Campo de texto, la única diferencia consiste en que solamente acepta caracteres numéricos, así como el punto decimal, el signo "-", su uso también es muy generalizado en todas las FORMAS; un ejemplo de este elemento de interface es el siguiente:

Al igual que en el caso anterior, si la variable ya tiene un valor inicial o anterior, este será colocado dentro del campo destinado para su uso en caso de querer realizar alguna modificación; la forma del movimiento del cursor dentro del campo es exactamente la misma que en el caso del CAMPO DE TEXTO, la única diferencia radica en que el valor de la variable es re-escrito con el formato al cual debe de estar apegado; por ejemplo, si se desea que la variable sea de dos números enteros y dos decimales, y solamente teclea el usuario los dos números enteros; al pasar de este campo a otro, el valor de la variable se re-escribe respetando este formato; de esta manera se le facilita el trabajo a el usuario.

- **BOTONES SIMPLES** : Generalmente se utiliza para representar una variable que puede tener uno de dos posibles estados "SI" o "NO", también se les llama variables binarias (1 o 0), las cuales representa una condición o una orden; un ejemplo de este elemento de interface es :

La forma de conmutar o intercambiar de un estado a otro estado se realiza a través de la "**barra espaciadora**", cuando el estado de la variable es "SI" o "ENCENDIDO", se identifica por la presencia de un patrón como puede ser (X), (•), (·), (✓) o (■) , en caso contrario se identifica por su ausencia.

- **LISTA DE CONTROLES O BOTONES :** Este elemento de interface tiene semejanza con los menús y con los campos de selección múltiple, se caracteriza por presentar una serie de valores que puede tener una variable en forma de una lista, de manera que puedan apreciarse todos a la vez y se pueda seleccionar solamente uno de ellos, se diferencia de los menús en que no dan acceso a aplicaciones funcionales, solamente fijan una variable a un determinado valor, por ejemplo :

- **DESLIZADORES (SLIDER'S) :** Su función principal consiste en fijar un valor entre un mínimo y un máximo, el usuario puede modificar su valor inicial mediante el arrastre de un botón o interruptor hacia alguno de los extremos del deslizador, su forma puede ser horizontal o vertical, un ejemplo es el siguiente:

5.5 FUNCIONAMIENTO DE LAS FORMAS

Las FORMAS generalmente utilizan para su funcionamiento dos tipos de cursores, uno que identifique a la variable activa, la cual se despliega en fondo inverso para que el usuario pueda identificarla fácilmente; y dentro de ella se despliega un cursor individual, ya sea en forma de una línea parpadeante, o cualquier otro carácter fácilmente identificable, el cual puede ser cambiado de posición dentro de ella; en el momento que el usuario pasa de

una variable a otra, se tiene que restaurar el video normal de la variable, colocar este video inverso en su nueva posición y colocar el cursor individual al inicio del campo.

Las FORMAS tienen que incluir una interacción entre los distintos elementos de interface que la conforman, de manera que se facilite la captura y corrección de los datos, auxiliando a el usuario en este aspecto; por este motivo, la FORMA debe de relacionar internamente todos y cada uno de los elementos de interface que contiene, los cuales representan a una variable, para que el usuario puede realizar alguna de las siguientes acciones:

- Capturar y/o modificar solamente aquellas variables necesarias, evitando con esto la perdida de tiempo al recorrer una por una todas las variables.

- Modificar solamente algún campo de la variable, sin tener que volver a introducir nuevamente todo su contenido, para esto, se deben de tomar en cuenta los dos modos de captura de datos que se pueden tener:

  - MODO DE INSERCIÓN : En el cual, se colocara el carácter en la posición donde se encuentre el cursor, y aquellos caracteres que se encuentren adelante de esa posición se recorrerán una posición, perdiéndose el último carácter.

```
| NOMBRE    PEREA CLAUDIA ANGÉLICA |
```

MODO DE INSERCIÓN

```
| NOMBRE    GUTIERREZ_PEREA CLAUDIA ANGÉLICA |
```

---------->Los caracteres se recorren conforme se van introduciendo datos.

  - MODO DE SUSTITUCIÓN : En el cual, se van reemplazando los caracteres donde se encuentre el cursor con los nuevos datos que se vayan introduciendo con el teclado, los caracteres que se

van perdiendo, son aquellos que se encuentran en la posición actual del cursor.

NOMBRE GUTIÉRREZ PEREA CLAUDIA ANGÉLICA

MODO DE SUSTITUCIÓN

NOMBRE MEDINAREZ PEREA CLAUDIA ANGÉLICA

------>Se sustituyen los caracteres que se encuentran en la posición actual del cursor.

Estos modos se intercambian de acuerdo a la tecla INSERTAR.

También debe de tomarse en consideración el borrado de caracteres de la variable, el cual puede ser mediante la tecla **backspace** o **suprimir**, sin modificar el resto de la variable.

- Debe de tomarse en cuenta el desplazamiento del cursor a lo largo de todos los campos que conforman la variable mediante las teclas flecha izquierda y flecha derecha; para poder realizar cualquier modificación.

NOMBRE GUTIERREZ PEREA CLAUDIA ANGÉLICA

<-- -->El cursor puede desplazarse a lo largo de todo el campo.

- Además del desplazamiento del cursor entre todos los campos de una variable, debe de considerarse un desplazamiento entre todas las variables de la FORMA, de manera que el usuario pueda posicionar el cursor en cualquier variable de ella, este movimiento debe de estar sujeto a las siguientes reglas:

  - Con la tecla Flecha abajo, se avanza el cursor a la variable que se encuentra en la parte inferior.

- Con la tecla Flecha arriba, se regresa el cursor a la variable que se encuentra en la parte superior.

- Con la tecla <ENTER> se avanza el cursor a la siguiente variable.

- Con la tecla Flecha izquierda y si tenemos el cursor en la posición inicial de la variable, regresamos el cursor a la variable anterior.

- Con la tecla Flecha derecha y si tenemos el cursor en la posición final de la variable, avanzamos el cursor a la siguiente variable.

- Con las teclas INCIO y FIN puede posicionarse el cursor en la primera y última variable de la FORMA.

- Si la FORMA tiene un número muy grande de variables, se pueden desplegar primero algunas de ellas, y posteriormente realizar un desplazamiento (SCROLL) para mostrar las siguientes; en estos casos, se utilizan las teclas RE.-PAG y AV. PAG para pasar a otra lista de datos o variables.

Si consideramos todos estos aspectos dentro de la elaboración de nuestras FORMAS, facilitaremos en gran medida la captura y modificación de datos a el usuario, haciendo que ésta sea una tarea fácil y rápida que cualquier usuario pueda realizarla, sin tener que aprender manejos complicados.

5.6 GENERACIÓN DE FUNCIONES.

Resulta imposible crear una función que coloque cualquier tipo de FORMAS en la pantalla, debido a la inmensa variedad de ellas, cada una con sus características particulares, así como los tipos de las variables, su posición y encabezado, los cuales

difieren de una FORMA a otra; por lo cual, no se puede generar una función que realice esta tarea; por esto, solamente generaremos las funciones de los elementos de interface mas comunes que pueden incluirse dentro de las formas, y al final del capitulo podrá apreciarse un ejemplo de un caso en particular de como pueden interrelacionarse todos ellos para la creación de una FORMA.

Cada uno de estos elementos de interface deben de tomar en cuenta las reglas para el movimiento del cursor dentro de la FORMA, las cuales son:

- La variable activa deberá de desplegarse en video inverso, y dentro de ella se colocará un cursor individual, el cual podrá ser cambiado de posición a lo largo de todos los campos que conforman la variable.

- Cada uno de los elementos de interface tendrá la capacidad de conmutar los dos modos de escritura que se pueden tener, ya sea modo de inserción o de sustitución.

Además deberá de considerarse en el diseño de la forma una serie de características que se relacionan con su desplegado en la pantalla, y que son muy similares a las características de los mensajes, como son:

- Se almacenará la información que este desplegada en el lugar donde se colocara la FORMA, con el fin de evitar la perdida de datos que sean necesarios para el usuario.

- Una vez que se ha cumplido la misión de la FORMA, se procederá a eliminarla de la pantalla y a colocar la información que había antes de su colocación.

Función : *lee_dato*(char *campo*[], int *digitos*, int *x*, int *y*, char *tipo*)

Parámetros :

campo	:	Cadena de caracteres a leer.
digitos	:	Numero máximo de digitos.
x	:	Posición en X donde se desplegará la variable
y	:	Posición en Y donde se desplegara la variable
tipo	:	Carácter que representa el tipo de datos que se pueden admitir en la lectura:

'A' = Aceptara caracteres alfabéticos.
'C' = Aceptara caracteres alfanuméricos.
'I' = Aceptara caracteres numéricos y los signos para la captura de datos enteros.
'N' = Aceptara caracteres numéricos. los signos y el punto decimal, para la captura de datos reales.

Acción : Realiza la lectura de una variable (la cual debe de ser una cadena de caracteres) en la Posición indicada por (*x,y*); los tipos de datos que puede admitir son definidos por la variable *tipo*, de esta forma, controlamos el acceso de los caracteres; utiliza las funciones ***escribe_video()*** para colocar los datos en la pantalla, y ***bioskey()*** para detectar cual tecla ha sido presionada, Funciona tanto para MODO TEXTO como para MODO GRÁFICO.

Valor de retorno : Un valor asociado con la última tecla que presionó el usuario de la forma siguiente:

VALOR	TECLA PRESIONADA
-1	ESCAPE
0	ENTER
1	FLECHA ARRIBA
2	FLECHA ABAJO
3	FINAL DEL TEXTO Y FLECHA DERECHA
4	INICIO DEL TEXTO Y FLECHA IZQUIERDA

VALOR	TECLA PRESIONADA
5	INICIO
6	FIN
7	AV. PAG.
8	RE- PAG.
9	F10

Función : *leo_int*(int *digitos*, int *var, int *x*, int *y*, int *lee*)

Parámetros :

digitos	:	Máximo número de digitos a leer.
var	:	Puntero de la variable donde se almacenara el valor de la lectura del campo.
x	:	Posición en X donde se desplegará la variable
y	:	Posición en Y donde se desplegara la variable
lee	:	Bandera para indicar cual es la acción que se realizara con la variable:
0	=	Solamente mostrara el valor de la variable.
1	=	La variable va a ser leída.

Acción : Realiza dos funciones, dependiendo de la variable *lee*:

- si *lee*=0, solamente colocara el valor de la variable.
- si *lee*=1 además de colocar el valor, realizara su lectura en las coordenadas definidas.

Esta función sirve solamente para campos enteros, utiliza las funciones ***escribe_video()*** para colocar los datos en la pantalla, y ***lee_dato()*** para realizar la lectura, funciona tanto para MODO TEXTO como para MODO GRÁFICO.

Valor de retorno : Un valor asociado con la última tecla que presionó el usuario durante la captura del texto de la forma siguiente (Este valor es el mismo que retorna la función ***lee_dato()***):

VALOR	TECLA PRESIONADA
-1	ESCAPE
0	ENTER
1	FLECHA ARRIBA
2	FLECHA ABAJO
3	FINAL DEL TEXTO Y FLECHA DER.
4	INICIO DEL TEXTO Y FLECHA IZQ.

VALOR	TECLA PRESIONADA
5	INICIO
6	FIN
7	AV. PAG.
8	RE. PAG.
9	F10

Ejemplo : La siguiente instrucción realiza la lectura de una variable entera, en la posición x=12, y=15;

lee_int(4,variable,12,15,1);

Función : *lee_float*(int *digitos*, int *dec*, float **var*, int *x*, int *y*, int *lee*)

Parámetros :

digitos	:	Máximo número de digitos a leer.
dec	:	Numero de decimales.
var	:	Puntero de la variable donde se almacenara el valor de la lectura.
x	:	Posición en X donde se desplegará la variable
y	:	Posición en Y donde se desplegara la variable
lee	:	Bandera para indicar cual es la acción que se realizara con la variable:
0 =		Solamente mostrara el valor de la variable.
1 =		La variable va a ser leída.

Acción : Esta función es muy similar en funcionamiento y estructura a la función anterior, la diferencia radica en que es utilizada para leer campos decimales, por esto, se incluye un parámetro mas, que corresponde precisamente a el número de decimales; la variable digitos, representa el numero total de campos que se colocaran; de esta forma, si tenemos los siguientes datos:

digitos=8 y dec=2

solamente aceptara 5 enteros, el punto y 2 decimales.

Valor de retorno : Un valor idéntico al de la función *lee_dato()*.

Ejemplo : La siguiente instrucción realiza la lectura de una variable de punto flotante, con 6 digitos enteros y 2 decimales (9 digitos en total contando el punto decimal); en la posición x=12, y=15;

lee_float(9,2,variable,12,15,1);

Función : *lee_doble*(int *digitos*, int *dec*, double **var*, int *x*, int *y*, int *lee*)

Parámetros :

digitos	:	Máximo número de digitos a leer.
dec	:	Numero de decimales que contendrá.
var	:	Puntero de la variable donde se almacenara el valor de la lectura.
x	:	Posición en X donde se desplegará la variable
y	:	Posición en Y donde se desplegara la variable
lee	:	Bandera para indicar cual es la acción que se realizara con la variable:
0 =		Solamente mostrara el valor de la variable.
1 =		La variable va a ser leída.

Acción : Su trabajo es el mismo que la función *lee_float()*, la diferencia consiste en el parámetro de lectura, el cual debe de ser del tipo **double**, que es utilizado para variables que necesitan una mayor precisión.

Valor de retorno : Un valor idéntico al de la función *lee_dato()*.

Ejemplo : La siguiente instrucción realiza la lectura de una variable de punto flotante con doble precisión con un formato de 6 posiciones enteras y 2 decimales (9 digitos en total contando el punto decimal), en la posición x=12, y=15;

lee_doble(9,2,variable,12,15,1);

Función : *lee_alfa*(char *campo*[], int *digitos*, int *x*, int *y*, int *lee*)

Parámetros :

campo	:	Cadena donde se almacenara la variable.
digitos	:	Numero de caracteres a leer.
x	:	Posición en X donde se desplegará la variable
y	:	Posición en Y donde se desplegara la variable
lee	:	Bandera para indicar cual es la acción que se realizara con la variable:
0 =		Solamente mostrara el valor de la variable.
1 =		La variable va a ser leída.

Acción : Realiza la lectura de un campo alfabético, el cual debe ser una cadena de caracteres, su estructura es muy similar a las funciones anteriores.

Valor de retorno : Un valor idéntico al de la función *lee_dato()*.

Ejemplo : La siguiente instrucción realiza la lectura de una variable tipo cadena de 30 posiciones, en la posición x=12, y=15;

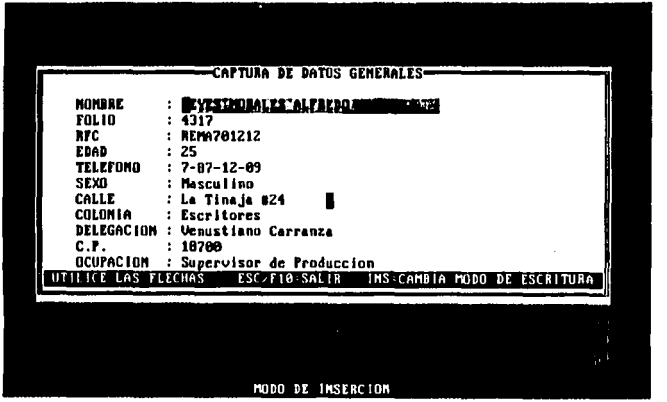
```
lee_alfa(nombre,30,12,15,1);
```

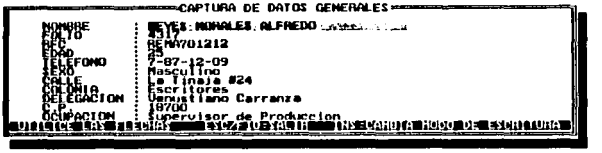
Ejemplo para el desplegado de una FORMA

```
/*************************************************************************
 *    Ejemplo sobre  la utilización de  las funciones de lectura de      *
 *    datos lee_int(), lee_alfa() y de su integración dentro de una      *
 *      forma de datos.                                                  *
 *      Para la selección de algún modo  gráfico es  necesario que se    *
 *      encuentren en el mismo directorio el controlador EGAVGA.BGI.     *
 *************************************************************************/
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <alloc.h>
#include "usuario.c"

void formal(char titulo[],int color,int sombra)
{
 int x1=5,y1=5,x2=75,y2=19,x[50],y[50],elementos,i,sale=0;
// int x1=1,y1=1,x2=50,y2=50,x[50],y[50],elementos,i,sale=0;
 char textos[11][30]={"NOMBRE     : ", "FOLIO      : ","RFC        : ",
                      "EDAD       : ", "TELEFONO   : ", "SEXO       : ",
                      "CALLE      : ", "COLONIA    : ", "DELEGACION : ",
                      "C.P.       : ", "OCUPACION  : "};
 int folio,edad,cp,apunta=0,reg;
 char nombre[35],rfc[12],tel[12],sexo[15],calle[35],colonia[35],deleg[35];
 char ocup[50];
 // NUMERO DE CAMPOS QUE SE COLOCARAN
 elementos=11;
 //INICIALIZACION DE LAS POSICIONES DE CADA UNA DE LAS VARIABLES
 x[0]=x[1]=x[2]=x[3]=x[4]=x[5]=x[6]=x[7]=x[8]=x[9]=x[10]=x1+5;
 for(i=0;i<elementos;i++) y[i]=y1+2+i;
 //INICILIZACION DE LAS VARIABLES, PUEDE INCLUIRSE UNA FUNCION PARA
 //OBTENER LAS VARIABLES POR MEDIO DE ALGUN ARCHIVO O ALGUN OTRO METODO
 strcpy(nombre,"REYES MORALES ALFREDO ");
 folio=4317;
 strcpy(rfc,"REMA701212");              edad=25;
 strcpy(tel,"7-87-12-09");              strcpy(sexo,"Masculino");
 strcpy(calle,"La Tinaja #24");         strcpy(colonia,"Escritores");
 strcpy(deleg,"Venustiano Carranza"); cp=18700;
 strcpy(ocup,"Supervisor de Produccion");
 //ALMACENAMIENTO DE LA PANTALLA DONDE SE COLOCARA LA FORMA
 if(MODO==0) salva_pantalla(x1,y1,x2+1,y2+1);
```

```
 else salva_pan_grafica(x1,y1,x2+1,y2+1);
 //COLOCACION DE LA FORMA
 marco(x1,y1,x2,y2,cfondo1,color,sombra,titulo);
 escribe_video(x1+1,y2-1,  " UTILICE LAS FLECHAS     ESC/F10=SALIR    INS=CAMBIA
MODO DE ESCRITURA ",16,15,15);
 for(i=0;i<elementos;i++) escribe_video(x[i],y[i],textos[i],cfondo1,0,0);
 //colocacion de los datos
 lee_alfa(nombre,32,x[0]+strlen(textos[0]),y[0],0);
 lee_int(4,&folio,x[1]+strlen(textos[1]),y[1],0);
 lee_alfa(rfc,10,x[2]+strlen(textos[2]),y[2],0);
 lee_int(2,&edad,x[3]+strlen(textos[3]),y[3],0);
 lee_alfa(tel,10,x[4]+strlen(textos[4]),y[4],0);
 lee_alfa(sexo,12,x[5]+strlen(textos[5]),y[5],0);
 lee_alfa(calle,32,x[6]+strlen(textos[6]),y[6],0);
 lee_alfa(colonia,32,x[7]+strlen(textos[7]),y[7],0);
 lee_alfa(deleg,32,x[8]+strlen(textos[8]),y[8],0);
 lee_int(5,&cp,x[9]+strlen(textos[9]),y[9],0);
 lee_alfa(ocup,45,x[10]+strlen(textos[10]),y[10],0);
 escribe_video(x[0]+strlen(textos[0]),y[0],nombre,cfondo1,0,0);
 //LECTURA DE LOS DATOS
 while(!sale)
  {
   switch(apunta)
    {
     case 0: reg=lee_alfa(nombre,32,x[0]+strlen(textos[0]),y[0],1);  break;
     case 1: reg=lee_int(4,&folio,x[1]+strlen(textos[1]),y[1],1);    break;
     case 2: reg=lee_alfa(rfc,10,x[2]+strlen(textos[2]),y[2],1);     break;
     case 3: reg=lee_int(2,&edad,x[3]+strlen(textos[3]),y[3],1);     break;
     case 4: reg=lee_alfa(tel,10,x[4]+strlen(textos[4]),y[4],1);     break;
     case 5: reg=lee_alfa(sexo,12,x[5]+strlen(textos[5]),y[5],1);    break;
     case 6: reg=lee_alfa(calle,32,x[6]+strlen(textos[6]),y[6],1);   break;
     case 7: reg=lee_alfa(colonia,32,x[7]+strlen(textos[7]),y[7],1); break;
     case 8: reg=lee_alfa(deleg,32,x[8]+strlen(textos[8]),y[8],1);   break;
     case 9: reg=lee_int(5,&cp,x[9]+strlen(textos[9]),y[9],1);       break;
     case 10:reg=lee_alfa(ocup,45,x[10]+strlen(textos[10]),y[10],1); break;
    }
   switch(reg)
    {
     case -1: case 9: sale=1; break;  //SE PRESIONO ESC o F10
     case 0: case 2: case 3:
      ((apunta>=elementos-1) ? apunta=0 : apunta++); break;
     case 1: case 4: ((apunta<=0) ? apunta=elementos-1 : apunta--); break;
     case 5: case 8: apunta=0; break;
     case 6: case 7: apunta=elementos-1; break;
    }
  }
 if(MODO==0) coloca_pantalla();
```

```
 else coloca_pan_grafica();
}

void  main()
 {
   char modo;
   clrscr();
   printf("\nMODO DE VIDEO A UTILIZAR : ");
   printf("\n    0=TEXTO ");
   printf("\n    1=GRAFICO RESOLUCION MEDIA");
   printf("\n    2=GRAFICO ALTA RESOLUCION ");
   printf("\n    CUAL ES SU OPCION : ? ");
   while(modo!='0' && modo!='1' && modo!='2') modo=getch();
   switch(modo)
    {
     case '0': coloca_caracter(1,1,176,7,16,2000); raton(); break;
     case '1': inicializa(0); raton_grafico(); break;
     case '2': inicializa(1); raton_grafico(); break;
    }
   muestra_cursor();
  indica_modo(insert);
  formal("CAPTURA DE DATOS GENERALES",0,1);
  if(modo=='1' || modo=='2') termina();
  clrscr();
}
```


CAPTURA DE DATOS GENERALES
NOMBRE : REYES MORALES ALFREDO
FOLIO : 4317
RFC : REMA701212
EDAD : 25
TELEFONO : 7-87-12-09
SEXO : Masculino
CALLE : La Tinaja #24
COLONIA : Escritores
DELEGACION : Venustiano Carranza
C.P. : 10700
OCUPACION : Supervisor de Produccion
UTILICE LAS FLECHAS ESC/F10-SALIR INS-CAMBIA MODO DE ESCRITURA
MODO DE INSERCION


CAPTURA DE DATOS GENERALES
NOMBRE : REYES MORALES ALFREDO
FOLIO : 4317
RFC : REMA701212
EDAD : 25
TELEFONO : 7-87-12-09
SEXO : Masculino
CALLE : La Tinaja #24
COLONIA : Escritores
DELEGACION : Venustiano Carranza
C.P. : 10700
OCUPACION : Supervisor de Produccion

CAPITULO VI

USO DE ICONOS EN MODO GRÁFICO.

6.1.- OBJETIVOS DEL CAPITULO

Entre los objetivos principales de este capítulo podemos mencionar los siguientes incisos:

a) Introducir al lector en uno de los mas importantes elementos de interface bajo ambiente gráfico (GUI).

b) Analizar sus características, ventajas, desventajas e importancia que pueden tener su uso dentro de nuestros programas.

c) Explicar las formas principales de implementar y almacenar las imágenes que constituyen el ICONO.

d) Crear las funciones que se encarguen de la colocación y lectura de los ICONOS en la pantalla.

6.2.- EL AMBIENTE GRÁFICO.

El uso de las imágenes gráficas dentro de los elementos de interface y en general dentro de todo el sistema se ha visto influenciado en los últimos años debido a dos motivos fundamentales, los cuales son:

- El ser humano ha desarrollado mecanismos de reconocimiento de imágenes a través del binomio ojos-cerebro, con los cuales percibe y procesa muchos tipos de datos en forma eficiente y rápida; a través de estas imágenes el cerebro asocia fácilmente una serie de ideas y conceptos relacionadas con ella, siendo este un método

muy eficiente de aprendizaje, ya que resulta ser intuitivo sin tener que aprender manejos complicados.

- En un principio el equipo necesario para procesar imágenes y tener buenos resultados era extremadamente caro; y en computadoras sin estas características, las imágenes tenían una definición muy mala, y la velocidad de desplegado era bastante lenta, lo cual causaba un gran aumento en el tiempo de respuesta; sin embargo, la tecnología ha superado todas estas desventajas, ofreciéndonos cada día dispositivos más rápidos y eficientes a un precio accesible, lo cual ha venido a fortalecer y fomentar el uso de gráficas e imágenes dentro de la mayoría de los sistemas.

El modo de video gráfico ha tenido un auge muy grande en todos los sistemas en los últimos años, ya que en él pueden obtenerse mejores resultados visuales que en modo texto, además puede incluirse en él tanto texto como imágenes gráficas, lo cual influye en gran medida en el aprendizaje del usuario; este auge se ha debido a varios aspectos como pueden ser:

- El uso de imágenes influyen en un rápido aprendizaje por parte de los usuarios y representa otro tipo de información que se puede incluir, lo cual influye en poder integrar mayores elementos en la información, teniendo datos más completos.

- Al poder incluir elementos gráficos en el sistema, este puede reproducir la información a semejanza del resultado final (en todos los aspectos como son imágenes, textos, tipos de letras, tamaño, etc.), Algunos autores denominan este concepto como WYSIWYG[19]. De esta forma observa resultados en forma rápida y se identifican fácilmente errores.

---

19 Tomado de las sílabas del Inglés What You See Is What You Get (LO QUE VES ES LO QUE OBTIENES).

- Puede integrar figuras gráficas a semejanza de objetos reales, lo cual influye en gran medida en el aprendizaje del usuario, el cual, bastará con que se familiarice con ellas para que comprenda su aplicación.

En los elementos de interface Gráficos, el usuario ve un tipo de medio ambiente artificial en abstracción de uno físico, el medio ambiente gráfico mas común dentro de los sistemas es de **MICROSOFT WINDOWS** y el **APPLE MACINTOSH**, en los cuales se despliega la información básica a semejanza de objetos reales como pueden ser: escritorios, papeles, archivos, botes de basura, gabinetes, etc. Este tipo de interacción con el usuario resulta ser intuitivo y puede ser usado por usuarios novatos y expertos.

Como puede observarse, las ventajas que nos proporciona el medio ambiente gráfico son bastantes y de gran utilidad, sin embargo, también existen muchas desventajas que podemos encontrar al utilizarlo, como pueden ser:

- Al utilizar el modo de video gráfico, la cantidad de memoria que utiliza para el almacenamiento de la pantalla es mucho mayor que en modo texto.

- La velocidad de respuesta en modo gráfico en mucho mas lenta que en modo texto.

- EL almacenamiento de información gráfica requiere de gran espacio

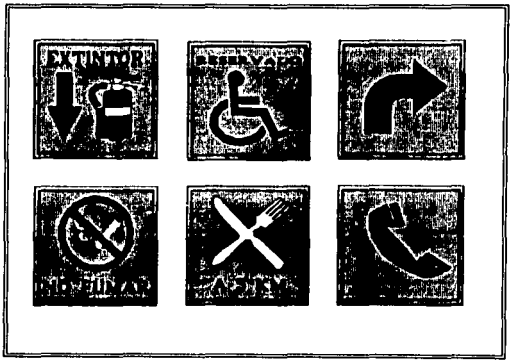
Debido a estos motivos, deberá de analizarse una serie de factores con el fin de determinar si conviene implementar el sistema en modo gráfico, como pueden ser:

- Deberá de tomarse en cuenta la aplicación del programa, con el fin de determinar si este, realmente necesita incluir imágenes para que su función sea mas completa.

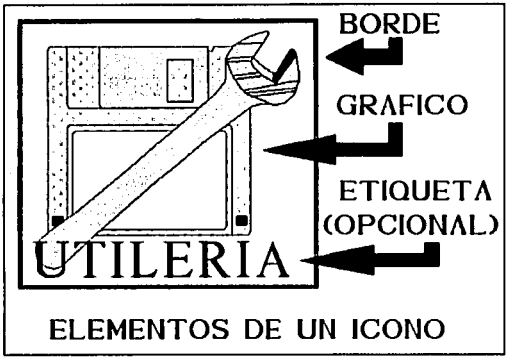
- Además, deberá de tomarse en cuenta el tipo de equipo en el cual se implementará.

6.3.- ICONOS, DEFINICIÓN, CARACTERÍSTICAS GENERALES E IMPORTANCIA.

Básicamente puede definirse a un ICONO como una pequeña figura, la cual representa una idea, un mensaje o un módulo, estos no solamente pueden encontrarse en los programas de computadora t ién podemos encontrarlos en la vida real como puede ser en se les de carreteras, en construcciones públicas, en las oficinas y en todos los lugares donde una idea no necesita estar formada por palabras; por ejemplo:



En un sistema de computo, un ICONO es una representación gráfica de una aplicación, es el método ideal para comunicar ideas gráficamente y de manera sencilla hacia los usuarios



ELEMENTOS DE UN ICONO

Un Icono, básicamente esta compuesto de los siguientes elementos principales[20] :

- **Un Borde** : el cual determina los limites del icono, puede considerarse como opcional. ya que en la mayoría de los casos , los limites son establecidos con la figura misma.

- **Una imagen** : Representa el elemento principal del ICONO, generalmente, es una figura pequeña, con un numero de colores no mayor a 16; la imagen debe de estar relacionado con el módulo, tarea o función que representa.

- **Una etiqueta de texto** : la cual ayuda aun más a identificar la función que representa el icono; esta etiqueta es opcional, y generalmente se incluye pero en forma separada a el ICONO.

---

20 Horton William. The Icon Book: Visual Symbols for Computer Systems and Documentation. Editorial John Wiley & Sons Inc. Primera Edición 1994.

Estos son los tres principales elementos que conforman un ICONO, aunque algunos de ellos pueden considerarse opcionales, sin embargo, la figura o imagen, representa el elemento principal del ICONO, y siempre deberá de incluirse.

6.4.- ESTRUCTURA DE LOS ICONOS.

Un ICONO, es una pequeña figura que representa una aplicación, de cualquier tipo; la cual puede estar codificada dentro del mismo programa o encontrarse en un archivo independiente, el cual contienen exclusivamente la definición del ICONO; cada uno de estos métodos de almacenamiento de la imagen tiene sus ventajas, desventajas, y formas de implementación.

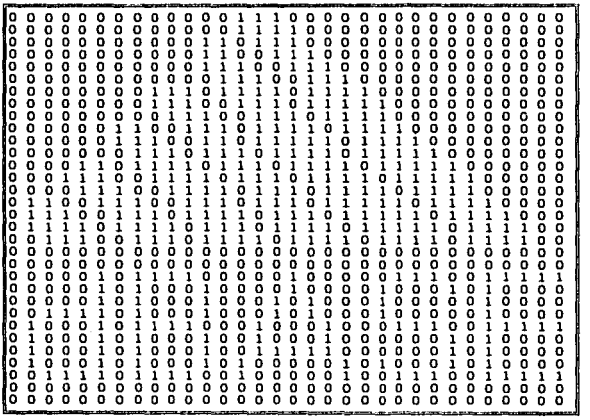
ICONOS DENTRO DE UNA APLICACIÓN

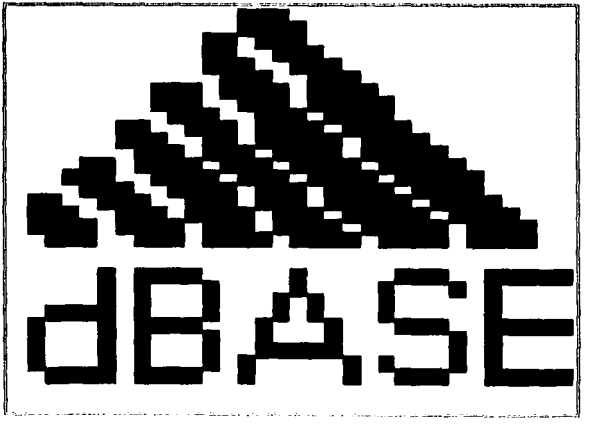
Para este caso, la definición de la imagen esta codificada dentro de la aplicación misma, por lo cual no puede tenerse acceso a ella si no se tiene el código fuente del programa; Básicamente existen dos formas de implementar los iconos dentro del programa; las cuales consisten en:

- La imagen que conforma el icono es codificada dentro de un arreglo bidimensional, cuyos limites son los mismos que el tamaño de la figura, así, por ejemplo, si el ICONO es de 32 x 32 pixeles deberá de tenerse un arreglo de 32 x 32 elementos, los cuales representa a un pixel, el cual pueden adquirir 2 valores, los cuales son :

  0 - Si el pixel que representa el elemento es color negro (Se encuentra apagado).
  1 - Si el pixel que representa el elemento es de color diferente (El punto esta encendido) .

  Así por ejemplo, la siguiente figura puede representarse en un arreglo bidimensional de la siguiente forma:




dBASE

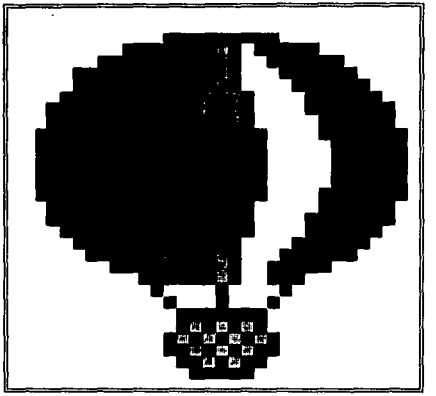
La principal ventaja de este método, es que resulta relativamente fácil su implementación; pero con la principal desventaja que poder representar solamente contornos o siluetas, ya que solo se tiene dos colores para todos los elementos que representaran la imagen del ICONO, los cuales son el blanco y el negro.

La segunda alternativa surge debido a la desventaja que tiene el método anterior en poder representar solamente siluetas o contornos; en este método se utiliza un índice para cada uno de los colores que se utilicen en la figura del ICONO, al cual se le denomina "MAPA DE COLORES" o "PALETA DE COLORES"; por ejemplo:

ÍNDICE	COLOR
0	NEGRO
1	CAFE
2	VERDE
3	VERDE OBSCURO
4	AZUL MARINO
5	MORADO
6	AZUL OBSCURO
7	GRIS

ÍNDICE	COLOR
8	GRIS CLARO
9	ROJO INTENSO
10	VERDE INTENSO
11	AMARILLO INTENSO
12	AZUL MARINO INTENSO
13	MORADO INTENSO
14	AZUL CLARO INTENSO
15	BLANCO INTENSO

En base a esta paleta o mapa de colores, se llenan los campos de un arreglo bidimensional, pero con la diferencia fundamental, que ahora sus elementos ya no son solamente ceros y unos, sino que ahora variaran de acuerdo al tamaño del mapa de colores, el cual también puede variar dependiendo de la cantidad que necesitemos, ya que podemos tener un mapa de 2,4,8,16,32 o hasta 256 colores; generalmente para figuras pequeñas, basta con tener 16 colores para obtener buenos resultados; por ejemplo, de acuerdo al mapa anterior, podemos representa la siguiente imagen como sigue:



Representando la cuarta parte de la imagen anterior, tendremos el siguiente arreglo, para este ejemplo tomaremos empezando de la esquina superior izquierda, tenemos los siguientes valores:

```
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 15 15 15 15 15 15 15 15  0  0  0  0
15 15 15 15 15 15 15 15 15  0  0  0 12  0  9  0
15 15 15 15 15 15 15 15  0  0 12 12 12  0  9  0
15 15 15 15 15  0  0 12 12 12 12  0  9  9  9  0
15 15 15 15 15  0 12 12 12 12  0  9  9  9  9  0
15 15 15 15  0 12 12 12 12  0  9  9  9  9  0  2
15 15 15  0 12 12 12 12 12  0  9  9  9  9  0  2
15 15 15  0 12 12 12 12  0  9  9  9  9  9  0  2
15 15  0 12 12 12 12 12  0  9  9  9  9  0  2  2
15 15  0 12 12 12 12  0  9  9  9  9  9  0  2  2
15 15  0 12 12 12 12  0  9  9  9  9  9  0  2  2
15 15  0 12 12 12 12  0  9  9  9  9  9  0  2  2
15 15  0 12 12 12 12 12  0  9  9  9  9  0  2  2
```

La principal ventaja de este método, consiste en los mejores resultados que pueden obtenerse al utilizar un numero mayor de colores para la colocación de la imagen.

Las desventajas de estos métodos de almacenar y representar los iconos dentro del programa de aplicación son:

- Por estar contenidos dentro del programa mismo, influyen en su tamaño.

- EN caso de tener la necesidad de realizar alguna modificación a algún ICONO, deberá de tenerse el código fuente, realizar las modificación y recompilar el programa, lo cual en muchas ocasiones no se tienen.

ICONOS EN ARCHIVOS INDEPENDIENTES.

Para este caso, los iconos son almacenados en archivos externos, los cuales contienen la definición de la imagen; para esta alternativa, nosotros podemos generar nuestra propia estructura del archivo que contendrá al icono, o podemos utilizar alguno de los formatos que podemos encontrar para almacenar imágenes gráficas[21], como pueden ser formatos BMP, PCX, GIF, PIC, etc.; para este caso, tratándose de iconos, ya existe un formato especial muy utilizado por muchas utilerías generadas para MICROSOFT WINDOWS, los cuales pueden generar archivos que contienen el ICONO, generalmente son archivos con extensión ICO.

Con el fin de aprovechar todos aquellos programas que pueden generar archivos de iconos, y de una inmensidad de iconos que podemos encontrar en este formato, nos basaremos en estos archivos para representar al ICONO que vayamos a utilizar.

---

[21] Para una mayor información sobre los formatos gráficos consultar: Kay C. David & Levine R. John. Graphics File Formats. Editorial Nindcrest/McGraw-Hill. Segunda edición 1995.

FORMATO DE LOS ARCHIVOS DE ICONOS (.ICO)

Los Iconos son comúnmente almacenados como archivos binarios con la extensión ICO; estos iconos son imágenes con un formato BITMAP (BMP) especializado. Un Archivo con extensión ICO contiene una imagen de 32 x 32 pixeles, además del mapa de colores utilizado; esta imagen contenida en el archivo es utilizada como un icono dentro de MICROSOFT WINDOWS; El archivo es de 766 bytes de longitud.

La forma en que esta compuesto el archivo es de la siguiente forma:

- Primeramente contiene una estructura de encabezado con los siguientes datos:

Posición (byte)	Tamaño (byte)	Nombre	Descripción
0	2	ICO_RESERVED 1	Valor reservado, siempre es 1.
2	2	ICO_RESOUCE_TYPE	Su valor siempre es 1.
4	2	ICO_RESOUCE COUNT	Especifica el numero de imágenes en el archivo, en archivos ICO siempre es 1.
6	1	ICO_WIDTH	Altura de la imagen (32).
7	1	ICO_HEIGHT	Ancho de la imagen (32).
8	1	ICO_COLOR_COUNT	Numero de colores en el icono (16).
9	1	ICO RESERVED 2	Valor Reservado.
10	2	ICO_RESERVED 3	Valor Reservado.
12	2	ICO_RESERVED 4	Valor Reservado.
14	4	ICO_DBL_SIZE	Especifica el número de bytes del "BITMAP" independiente contenido en el archivo ICO, este valor es de 744.
18	4	ICO_DBL_OFFSET	Define la posición en el archivo ICO donde empieza el "BITMAP" independiente, su valor es 22.

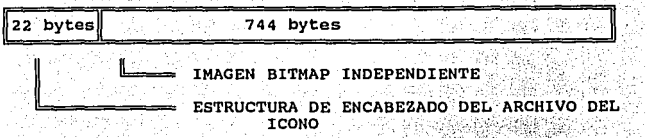
Es decir, el archivo ICO esta dividido en 2 grandes partes:

- La primera contiene una estructura donde se definen los

datos fundamentales del archivo (ICON_FILE_HEADER), el cual ocupa un total de 22 bytes.

- La segunda parte es idéntica a un archivo BITMAP (MAPA DE BITS), a excepción de su encabezado, para los archivos BITMAP, su encabezado (BITMAP_FILE_HEADER) es de un total de 16 bytes.

Esto se puede apreciar mejor en la siguiente figura:



- Posteriormente a partir de la Posición 22 del archivo, la estructura es idéntica a la de un archivo con formato BMP, la única deferencia radica en la Posición de los datos, ya que en el archivo ICO empieza en la dirección 22, mientras que en el archivo BITMAP (BMP) empieza en la dirección 14.

La estructura que continua a partir de la dirección 22 (BITMAP_INFO_HEADER) es la siguiente:

Posición (byte)	Tamaño (byte)	Nombre	Descripción
22	4	Bi_Size	Define el tamaño de este Encabezado, el cual es de 40 Bytes
26	4	Bi_Width	Altura de la Imagen en pixeles
30	4	Bi_Height	Ancho de la Imagen en pixeles
34	2	Bi_Planes	Numero de planos de imagen, debe de ser 1.
36	2	Bi_Bi_Count	Bits por pixel, puede ser 1,4,8 o 24
38	4	Bi_Compression	Tipo de Compresión Utilizada.

Posición (byte)	Tamaño (byte)	Nombre	Descripción
42	4	Bi_SIze_Image	Tamaño en Bytes de la imagen comprimida o cero.
46	4	Bi_X_Pels_per_Meter	Resolución Horizontal, en pixeles/metro.
50	4	Bi_Y_Pels_per_Meter	Resolución Vertical en pixeles/metro.
54	4	Bi_Clr_used	Numero de colores utilizados
58	4	Bi_Clr_Important	Numero de colores "importantes".
62	4*N	Bmi_Colors	Continua el mapa de colores.

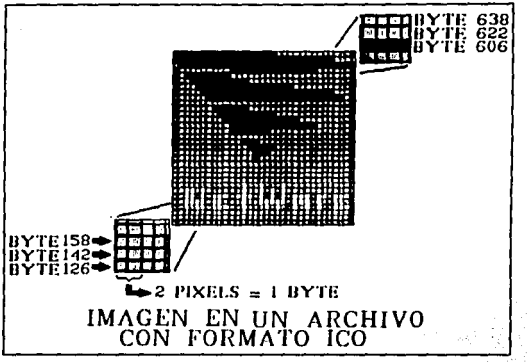
- El mapa de colores utilizado por los archivos de iconos es de 16 elementos, cada uno de los cuales ocupa un total de 4 bytes del archivo y se encuentra a partir del byte 62; su finalidad consiste en definir una paleta de colores, o mapa de los colores utilizados por la figura, basadas en intensidades relativas de azul, verde y rojo; el formato de cada uno de los elementos que lo integran es de la siguiente forma:

BYTE	NOMBRE	COLOR	RANGO
1	RGB-AZUL	INTENSIDAD AZUL PARA LA ENTIDAD	0 A 255
2	RGB-VERDE	INTENSIDAD VERDE PARA LA ENTIDAD	
3	RGB-ROJO	INTENSIDAD ROJO PARA LA ENTIDAD	
4	RGB-RESERVADO	VALOR IGUAL A CERO	BYTE NO USADO, SU VALOR PUEDE SER CERO

Cada entidad es expresada como una longitud de 8 bits (1 byte), y todas ellas en su conjunto definen uno de los colores; De esta forma esta constituido cada uno de los 16 elementos que integran el mapa de colores.

- La figura en si comienza en la byte 126, y se encuentra almacenada en 2 pixeles por byte, es decir, por cada byte (8 bits), 4 bits corresponden a un pixel, y los 4 restantes corresponden a otro, la forma que cada posición del archivo

corresponde a los pixeles puede observarse en la siguiente figura, la cual muestra el contenido de un archivo:



Al obtener la lectura de los elementos de un archivo con extensión ICO, se obtendrá primero un arreglo de 16x32 elementos, debido a que cada elemento representará dos pixeles, para el caso de la figura anterior, representando la cuarta parte de la matriz, tenemos lo siguiente:

153	153	153	153	153	153	119	121
153	153	153	153	153	153	151	119
153	153	153	153	153	153	153	119
153	153	153	153	153	153	153	121
153	153	153	153	153	153	153	153
153	153	153	153	153	153	153	153
153	153	153	153	153	153	153	153
153	153	153	153	153	153	153	153
153	153	153	153	153	153	249	159
153	255	159	153	153	153	249	159
153	255	159	159	255	159	255	159
153	255	255	159	249	153	249	159
153	249	255	159	255	153	249	153
153	153	153	153	153	153	153	153
153	153	153	153	153	153	153	153
153	153	153	153	153	153	153	153

Sin embargo, debido a que cada elemento representa dos pixeles, tenemos que separarlos; para esto, la forma mas sencilla de entenderlo es colocarlos en su forma binaria, por ejemplo, los primeros elementos de la esquina inferior izquierda tenemos:

153->10011001	249->11111001	255->11111111
153->10011001	153->10011001	153->10011001
153->10011001	153->10011001	153->10011001
153->10011001	153->10011001	153->10011001

Cada uno de los elementos anteriores representa dos pixeles; los primeros 4 bits del elemento representa al pixel a la izquierda, y los 4 últimos bits representa al pixel a la derecha, es decir, se separan a la mitad; si separamos los elementos anteriores en sus grupos de 4 bytes tendremos el siguiente arreglo:

1001->9	1001->9	1111->15	1001->9	1111->15	1111->15
1001->9	1001->9	1001->9	1001->9	1001->9	1001->9
1001->9	1001->9	1001->9	1001->9	1001->9	1001->9
1001->9	1001->9	1001->9	1001->9	1001->9	1001->9

De esta forma se constituye el arreglo de 32 x 32 elementos, cada uno de los cuales representa a un pixel, sin embargo, un arreglo en esta forma binaria pueda causar un poco de confusión al tratar de interpretarlo, si convertimos los datos binarios a su equivalente decimal se tiene lo siguiente:

9	9	9	9	9	9	9	9	9	9	9	7	7	7	9	
9	9	9	9	9	9	9	9	9	9	9	9	7	7	7	
9	9	9	9	9	9	9	9	9	9	9	9	9	7	7	
9	9	9	9	9	9	9	9	9	9	9	9	9	7	9	
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	9	9	15	9	9	15	
9	9	15	15	9	15	9	9	9	9	9	9	15	9	9	15
9	9	15	15	9	15	9	15	15	15	9	15	15	15	9	15
9	9	15	15	15	15	9	15	15	9	9	9	15	9	9	15
9	9	15	9	15	15	9	15	15	15	9	9	15	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	

Cada uno de los elementos anteriores representa un índice, el cual hace referencia a los colores contenidos en el MAPA DE COLORES, cada uno de los elementos anteriores representa al siguiente color:

7 - Representa al color gris
9 - Representa al color Rojo
15 - Representa al color Blanco

De esta forma, se obtiene la figura en base a el mapa de colores; si sustituimos estos colores en el arreglo anterior, obtendremos la imagen que se tiene almacenada en el archivo.

---

cualquiera de los dos métodos puede utilizarse, sin embargo, se recomienda, y generalmente se utiliza en todos los sistemas el segundo método, es decir, los ICONOS se encuentran en archivos independientes, esto es debido a los siguientes aspectos:

- Al tener los iconos en archivos independientes, se disminuye el tamaño del programa ejecutable, ya que eliminamos código del programa principal, y se direcciona a estos archivos.

- En caso de que se necesite modificar algún icono, bastara con modificar el archivo que lo contiene, si los iconos se encontraran en el programa principal, habría que modificarlo y recompilarlo, lo cual puede ser mas tedioso y difícil.

- Al estar los iconos en archivos independientes, pueden actualizarse más fácilmente su contenido, cambiando, estos archivos por otros.

6.5 GENERACIÓN DE FUNCIONES.

Hoy en día, el estandard dentro de las computadoras **IBM PC** y compatibles en lo que se refiere a medio ambiente gráfico, lo ha establecido **MICROSOFT WINDOWS**, en el cual existen una gran variedad de programas dentro de los cuales pueden generarse archivos de ICONOS, también pueden encontrase una diversidad de archivos con este formato, los cuales pueden utilizarse fácilmente, trataremos de aprovechar estas ventajas.

Por esto, para la generación de las rutinas, deberá de tomarse en cuenta, que la definición de las imágenes se establecerá en archivos independientes, los cuales tendrán el formato gráfico ICO, el cual ya se definió anteriormente, para efecto del mapa de colores, solamente deberá de obtenerse de un archivo de un ICONO, con el fin de ahorra tiempo sin tener que leerlos de todos ellos, ya que en todos ellos el mapa de colores es el mismo.

La estructura que contendrá el nombre de los archivos de ICONOS que queremos representar será muy similar a la que se uso en la parte de menús; es decir, será un arreglo de cadenas; además, se utilizaran otros dos arreglos:

- Uno para la definición de las leyendas o texto que se le incluirá en la parte inferior de cada icono.

- Otro de enteros, el cual deberá de contener las coordenada "X" y "Y" donde se colocaran cada uno de los iconos.

También debe de tomarse en cuenta, que para la colocación de modo gráfico, TURBO C incluye una serie de archivos necesarios para poder realizar esta tarea; en este caso, será necesario incluir con el programa el archivo EGAVGA.BGI para poder establecer el modo gráfico y poder representar los ICONOS.

Función : *inicializa(int modo)*

Parámetros : *modo* = Indica el modo de video gráfico que se quiere utilizar
0 = GRAFICO MEDIA RESOLUCIÓN
1 = GRAFICO ALTA RESOLUCIÓN.

Acción : Su función principal consiste en inicializar el modo gráfico, para lo cual requiere que se tenga presenta una tarjeta de video VGA, y se encuentren presentes con el programa los dispositivos para su manejo, el cual es el archivo "EGAVGA.BGI", el cual es parte de la libreria gráfica de TURBO C.

Valor de retorno : Ninguno, en caso de error la función lo detecta y es abortado el programa.

Función : *termina()*

Parámetros : Ninguno.

Acción : Se encarga de cerrar el modo de video gráfico, y regresar al modo de video que se tenía antes de ejecutar el programa.

Valor de retorno : Ninguno.

Función : *lee_paleta*(char *archivo1*[],struct *mapa_colores color*[])

Parámetros : *archivo1*: Nombre del archivo que contiene el icono.
color : Arreglo de estructura, dentro de la cual se va a

almacenar cada uno de los componentes que forman el mapa de los 16 colores.

Acción : Se encarga de realizar la lectura exclusivamente del mapa de colores del archivo que contiene el ICONO, esta es almacenada dentro de un arreglo de estructuras, la cual contiene cada uno de los componentes de los colores primarios que forman cada uno de los elementos del mapa de colores.

Valor de retorno : Ninguno, pero los valores que contenía el arreglo que definirá los colores que se utilizarán serán modificados por aquellos valores que sean obtenidos de la lectura del archivo.

Función : *recupera_icono*(char *archivo1*[],unsigned char *grafico*[32][32])

Parámetros :
- *archivo1*: Nombre del archivo que contiene el icono.
- *grafico* : Arreglo bidimensional donde se almacenara el contenido de la imagen en base a los índices de los colores que representa cada uno de los puntos.

Acción : Se encarga de realizar la lectura de los puntos que conforma la imagen que representa a el ICONO, los cuales son almacenados dentro del arreglo bidimensional "*grafico*".

Valor de retorno : Ninguno, pero los valores que contenía el arreglo que definirá la imagen serán modificados por aquellos valores que sean obtenidos de la lectura del archivo.

Función : *fija_paleta*(struct *mapa_colores color*[])

Parámetros : *color* : Arreglo de estructuras que contienen la definición del mapa de colores en base a tonalidades.

Acción : Se encarga de generar y fijar el mapa de colores necesario para colocar el ICONO con sus respectivos colores; en caso de no realizarse esta función antes de colocar el ICONO, este será dibujado de acuerdo a la paleta de colores que se tenga, lo cual causará que los colores no sean los adecuados, aunque la figura sea la correcta.

Valor de retorno : Ninguno.

Función : *muestra_icono*(unsigned char **grafico**[32][32],int *x*,int *y*)

Parámetros : *grafico* : Arreglo de estructuras que contienen la definición de la imagen del ICONO.
x : Posición en X de la esquina superior izquierda donde será colocado el icono.
y : Posición en Y de la esquina superior izquierda donde será colocado el icono.

Acción : Se encarga de colocar el ICONO, de acuerdo a los datos contenidos dentro del arreglo bidimensional destinado para esa tarea, el cual es llamado "*grafico*", la esquina superior izquierda del ICONO será colocada en la Posición que indique el usuario a través de las variables "*x*" y "*y*", para esto, la pantalla, ya debe de encontrase inicializada en modo gráfico.

Valor de retorno : Ninguno.

Función : *resp_icono*(int *elem*,int *raton*[][4])

Parámetros : *elem* : numero de ICONOS que se tienen representados, por lo tanto, también representa uno de los limites de la variable *raton*.

raton : Arreglo bidimensional, el cual contiene las coordenadas de la esquina superior izquierda y la esquina inferior derecha, necesarias para obtener la lectura del usuario con el uso del ratón.

Acción : Su función principal consiste en obtener la lectura sobre una serie de iconos que son mostrados en la pantalla hacia el usuario, esta lectura se realiza checando en que partes de la pantalla es presionado el botón izquierdo del ratón, y checandolas con las todas las posiciones de los iconos que se tienen, para verificar si concuerda con alguna de ellas, en caso de que esto no suceda, se sigue esperando hasta obtener otra lectura del ratón.

Valor de retorno : Retorna un valor de acuerdo con el icono que haya seleccionado el usuario, este valor estará relacionado con la posición que ocupa el ICONO dentro del arreglo bidimensional que define los nombres de los iconos; por ejemplo, si se tiene el siguiente arreglo:

```
char *menu1[]={ "disk","bc","123j","amipro","cprompt2","\0") ;
```

Los cuales pueden estar representados en orden diferente, de acuerdo a las posiciones X y Y que se le definan a cada uno. Si se selecciona el ICONO que fue leído del archivo "bc.ico", se retornará un valor de 0, debido a que es el elemento cero (0) del arreglo, y así sucesivamente.

Función : *menu_ver_iconos*(char *menu[],char *etiq[],int *pos*[][2])

Parámetros : *menu* : Arreglo bidimensional, el cual contiene el nombre de los archivos de los ICONOS que se quieren observar en la pantalla.

etiq : Arreglo bidimensional, el cual contiene las etiquetas que se colocarán debajo del ICONO para una mejor identificación.

pos : Arreglo bidimensional de enteros, el cual contiene las posiciones "X" y "Y" donde serán colocados cada uno de los iconos.

Acción : Realiza la función de obtener el mapa de colores de los archivos de iconos contenidos dentro de la variable "*menu*", los coloca en la pantalla en las posiciones que defina el usuario a través de la variable "*pos*" y con la leyenda o etiqueta definida en el arreglo "*etiq*", utiliza para su funcionamiento las funciones definidas anteriormente para poder realizar estas tareas.

Valor de retorno : Retorna un valor de acuerdo con el icono que haya seleccionado el usuario, este valor es el mismo que retorna la función anterior "resp_icono".

EJEMPLO PARA EL DESPLEGADO DE ICONOS

```
/*************************************************************************
 *     Ejemplo sobre la utilización de la función menu_iconos() para     *
 *     la colocación de iconos en la pantalla; en este programa es       *
 *      indispensable que se tenga activado el ratón, ya que esta es la  *
 *      única interacción con el usuario, también es indispensable que   *
 *      se encuentre en el mismo directorio el controlador EGAVGA.BGI,   *
 *      el cual es utilizado para el manejo de gráficos.                 *
 *************************************************************************/

#include <stdlib.h>
#include <graphics.h>
#include <dos.h>
#include "usuario.c"

void main()
 {
   int eleccion,c;
   char *menu1[]={
     "bc",       "disk",    "123j",   "amipro",
     "balloon2", "clipper", "hp",     "password",
     "fox",      "ibm3",    "word1",  "logitec2",
     "netware",  "print",   "tools1", "cprompt2",
     "\0"} ;
   char *etiq1[]={
     " C++ ",    "JUEGOS",    "Lotus",   "AMIPRO",
     "COREL",    "CLIPPER",   "Utilerias", "Password",
     "FOX PRO",  "IBM", "Word",        "Scanner",
     "Redes",    "IMPRIMIR",  "Utilerias",  "Salir2",
     "\0"} ;
   int pos[][2]={20 ,30 ,100,30 ,180,30 ,260,30 ,340,30 ,420,30 ,500,
                 30 ,580,30 , 20,100,100,100,180,100,260,100,340,100,
                 420,100,500,100,580,100};
   char archivo1[15],mens[60];
    inicializa(1);
    raton_grafico();
    muestra_cursor();
    eleccion=menu_iconos(menu1,etiq1,pos);
    switch (eleccion)
     {
      case -1: sprintf(mens,
            "PRESIONO LA TECLA ESC O EL BOTON DERECHO DEL RATON"); break;
      default:
      sprintf(mens,"SELECCIONO EL ICONO No %2d : %s.ICO ",eleccion+1,
              menu1[eleccion]); break;
```

```
    }
   men_esp(20,13,mens,64,15,1);
   men_esp(20,20,"PRESIONE ENTER PARA TERMINAR ",64,15,1);
   bioskey(0);
   termina();
   clrscr();
}
```

USO DE ICONOS EN MODO GRÁFICO.

CONCLUSIONES

Los elementos de interface con el usuario, no constituyen el sistema en su totalidad y su uso no garantiza que vaya a ser del agrado de los usuarios, ni influyen en sus buenas o malas herramientas de trabajo que pudieran tenerse; el éxito de un programa se basa en combinar en forma eficiente varios aspectos que influyen de manera notable sobre la decisión del usuario hacia algún producto en especial, los cuales son:

- **LAS HERRAMIENTAS DE TRABAJO:** Las cuales deben de ser claras, eficientes y rápidas, de manera que resuelvan un problema, o realicen una tarea en su totalidad, tratando de disminuir su tiempo de ejecución.

- **LOS ELEMENTOS DE INTERFACE:** Los cuales se encargaran de dar acceso a todas las herramientas que contenga el programa, y de auxiliar a el usuario y facilitar el uso del mismo.

- **LA COMBINACIÓN ELEMENTOS DE INTERFACE - HERRAMIENTAS:** Sin duda alguna, de nada sirve un programa sin herramientas, y un programa sin elementos de interface resulta ser complicado e ineficiente; por este motivo, se debe de buscar la relación óptima, de manera que el usuario pueda buscar de una manera sencilla todas las herramientas y pueda utilizarlas; para esto, debe de tomarse en cuenta la clasificación de todas las opciones del sistema, de manera que su búsqueda sea rápida y sencilla.

- **LOS MÉTODOS DE AYUDA:** Hoy en día todo usuario exigente, requiere que el programa contenga los métodos de ayuda mas modernos y eficientes, ya que con ellos, tendrán un gran apoyo en el uso del programa y en la tarea que realizan las herramientas que contiene, disminuyendo con esto el tiempo de búsqueda y consulta en los manuales.

- **EL COSTO:** Este aspecto es el que mas influye sobre la decisión del usuario, ya que, si un programa es muy costoso, no importando que sea la octava maravilla del mundo, y prácticamente resuelva los problemas sin la interacción del usuario, pero el costo esta fuera del alcance de los posibles usuarios, será imposible que este sea adquirido, y su compra estará limitada a un pequeño grupo de posibles usuarios con los recursos necesarios, limitando con esto, las posibles ganancias.

Como se puede observar, Los elementos de interface representan la forma en que un sistema interactua con el usuario de una manera flexible, fácil y rápida, lo cual influye directamente en la competitividad del programa con otros que tratan o realizan la misma función, solamente si se combina en forma eficiente con todos los aspectos anteriores, de manera que la mayor cantidad de usuarios se decidan por la utilización de nuestro producto.

En esta tesis, se hizo incapie en el diseño de una librería de funciones, la cual contenga una serie de herramientas que auxilien al programador y/o ingeniero a realizar la interface con el usuario y a desarrollar sus programas en forma más rápida y eficiente; estos aspectos pueden generar una serie de beneficios tangibles como pueden ser:

- Se tiene integradas una serie de funciones que nos auxiliaran y nos facilitaran la creación del programa.

- En el diseño de algún programa, el uso de la librería puede ahorra bastante tiempo en el diseño del mismo, en lugar de empezar sin prácticamente ninguna ayuda.

- El sistema resulta mas confiable, debido a que se controla y limita el acceso de los datos al sistema, disminuyendo la probabilidad de error y fallas en el sistema.

Otros beneficios son para el usuario, entre los principales podemos mencionar los siguientes:

- Con el uso de los elementos de interface auxiliando a el usuario en la elección de tareas y manejo del sistema, el usuario queda satisfecho y con un buen concepto del sistema, obteniéndose una buena disposición por parte de los usuarios a usar el sistema.

- Usando elementos de interface en forma eficiente, provoca que el uso del sistema sea más fácil, rápido y entendible, lo cual influirá directamente en el tiempo de aprendizaje sobre el uso del sistema, el cual, en la mayoría de los casos resulta ser intuitivo y fácil.

- Con el uso de los elementos de interface auxiliandonos en el manejo del programa, el tiempo que el usuario utiliza para realizar su trabajo se disminuirá notablemente, causando un incremento en la productividad, ya que se podrá realizar el trabajo en un tiempo menor.

Si un usuario, se siente satisfecho con las ventajas que le proporciona nuestro programa con el uso de las herramientas o funciones que pudiera realizar, además de la forma que interactua con él por medio de los elementos de interface, facilitando su uso y acceso a estas herramientas, este programa, será preferido por los usuarios, ya que cumple con la tarea principal de todo programa: satisfacer una necesidad en forma rápida, eficiente y sencilla. De nada sirve generar un sistema cuyo uso sea complicado, y sus herramientas de trabajo estén ocultas, o sea difícil acceder a ellas; esto solamente causará una mala impresión y todo usuario exigente tratará evitar su uso.

Sin embargo, incluir los aspectos anteriores influye notablemente en el costo del producto, y en un aumento de tiempo en

el desarrollo del sistema, ante esta posible desventaja, puede surgir algunas interrogantes:

¿ Que tan factible resulta incluir los elementos de interface en los programas ?
¿ Los beneficios serán mayores que los costos ?
¿ En verdad valdrá la pena gastar tiempo y dinero tratando de incluir elementos de interface rápidos y eficientes ?

Para contestar las preguntas anteriores, tenemos que tomar en cuenta dos aspectos fundamentales en el diseño del sistema, los cuales son:

- El ciclo de vida del programa.
- La Re-Usabilidad del Código.

Si no tomamos en cuenta las necesidades del usuario al operar el sistema, y la competencia por parte de otras compañías, el ciclo de vida del programa será muy reducido, y por otra parte el usuario optará por otros sistemas que en verdad le faciliten el trabajo, disminuyendo con esto, desde el diseño de nuestro programa las posibles ganancias que podamos obtener con él, ya que será superado ampliamente por la competencia, y quedara obsoleto en muy poco tiempo; por otro lado, si tomamos en cuenta a el usuario en el proceso de construcción de nuestro programa, este permanecerá más tiempo en el mercado, y si se combina eficientemente con las herramientas de trabajo, podrá contar con la popularidad entre los usuarios, obteniendo mayores ganancias, y teniendo la posibilidad de crear nuevas actualizaciones, con las cuales aumentara el ciclo de vida de nuestro sistema; Además, seria ilógico no tomar en cuenta las necesidades del usuario, si precisamente es a este último a quien va orientado el sistema.

Por otra parte, la Re-Usabilidad del Código es otro aspecto muy importante, el cual debe de tomarse en cuenta para el desarrollo de nuevos programas, si tomamos en cuenta este punto al diseñar algún sistema, se nos facilitará notablemente el diseño de nuevos programas, debido a que ya contaremos con una base, la cual nos auxiliara notablemente en este aspecto, a tal grado de **NO INFLUIR** notablemente en el tiempo de diseño y en el costo del

SOFTWARE, debido a que ya se tiene generadas una serie de funciones y rutinas que nos servirán y facilitaran el desarrollo del nuevo programa; sin embargo, esta ventaja de crear código re-utilizable, solamente se observara hasta contar con una librería muy completa y eficiente, la cual se pueda ir actualizando.

Si tomamos en cuenta todos estos aspectos el resultado final resultará de mejor calidad y de mayor aceptación por parte de los usuarios; los cuales preferirán algún SOFTWARE que le facilite el trabajo; siendo nuestro sistema más competitivo en el mercado de ventas, y teniendo la posibilidad de obtener un número mayor de ganancias, y teniendo la posibilidad de incrementar el ciclo de vida del sistema si es que combina en forma eficiente elementos de interface y herramientas, con lo cual, podemos obtener mayores ganancias con un esfuerzo menor.

Por este motivo, se ha hecho incapie dentro de esta tesis en que el usuario, tenga una librería de funciones, las cuales le auxiliaran en la programación y en la solución de sus problemas; una vez que se tengan estas librerías, el uso de los elementos de interface influirán en un porcentaje mínimo en el costo del programa y en el tiempo de su desarrollo; los cuales estarán determinados en su mayor porcentaje por la cantidad y calidad de tareas que se quiera que el sistema realice; el punto de mayor peso radica directamente en los beneficios que se obtendrán, ya que si se combinan de manera eficiente las herramientas de trabajo y los elementos de interface, el sistema será de fácil utilización y se podrán obtener resultados más rápidamente, lo cual influye en forma determinante sobre la decisión de un usuario hacia algún programa en particular.

APÉNDICE A

En esta sección se incluye el listado de todas las funciones desarrolladas a lo largo de esta tesis, en caso de querer utilizarlas inmediatamente, sin tener la necesidad de teclas todas estas líneas, se incluye con esta tesis un diskette con todos los ejemplos generados, cada uno de ellos con el archivo que contiene todo el código de este APÉNDICE, este archivo se llama usuario.c

```
/*****************DEFINICIÓN DE LIBRERÍAS UTILIZADAS ********************/
#include <conio.h>
#include <bios.h>
#include <dos.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <io.h>
#include <graphics.h>
#include <alloc.h>
#include <fcntl.h>
#include <math.h>

/************************DEFINICION DE CONSTANTES*************************/
#define ctexto 0
#define cfondo1 112
#define cfondo2 32
#define cfondo3 64
#define columnas 80
#define renglon 25
#define mem_vid (char far *)0xB8000000
#define arch_temporal "TEMPO01.TMP"


/***************** DECLARACION DE VARIABLES DE USO GLOBAL *****************/
struct mapa_colores {unsigned char azul,verde,rojo,reserv;} color[16];
union REGS r;
int insert=1,factorx,factory,ANCHO=1,ALTO=1,MODO=0;
unsigned long int TAMANO=0,TAM_GRAF=0;
short int *pantalla;
long *pan_grafica;
FILE *a_tempo;

/*******************DEFINICION DE LAS FUNCIONES ***************************/

void inicializa(int modo);
  /*************************************************************************
   *  FUNCIÓN : Inicializa el modo de video gráfico, el parámetro modo     *
   *            define la resolución que se desea utilizar                 *
   *            modo = 0  -> Resolución media                              *
   *            modo = 0  -> Resolución alta                               *
   *************************************************************************/
```

```
#define termina() { closegraph(); MODO=0;}
  /*************************************************************************
   *  FUNCIÓN : termina el uso del video en modo gráfico y regresa         *
   *            el video a modo texto.                                     *
   *************************************************************************/

void raton(void);
  /*************************************************************************
   *  FUNCIÓN:  ACTIVA EL MANEJADOR DEL RATON EN MODO TEXTO                *
   *************************************************************************/

void raton_grafico(void);
  /*************************************************************************
   *           ACTIVA EL MANEJADOR DEL RATON EN MODO GRÁFICO               *
   *************************************************************************/

#define muestra_cursor() { r.x.ax=1;  int86(0x33,&r,&r); }
  /*************************************************************************
   *  FUNCIÓN:  Muestra el cursor del raton, tanto para modo texto como    *
   *            en modo gráfico                                            *
   *************************************************************************/


#define oculta_cursor()  { r.x.ax=2;  int86(0x33,&r,&r); }
  /*************************************************************************
   *  FUNCIÓN:  Oculta el cursor del raton, tanto para modo texto como     *
   *            en modo gráfico                                            *
   *************************************************************************/

void coloca_caracter(int pos_x, int pos_y, char caracter, int color_caracter,
     int color_fondo, int veces);
  /*************************************************************************
   *  COLOCA UN CARACTER EN EL MONITOR CON LOS ATRIBUTOS ESPECIFICADOS     *
   *       posx,posy       = POSICIÓN DONDE SE COLOCARA EL CARACTER        *
   *       caracter        = CARACTER A SER DESPLEGADO                    *
   *       color_caracter  = COLOR DEL CARACTER                           *
   *       color_fondo     = COLOR DE FONDO                               *
   *       veces           = NUMERO DE CARACTERES A DESPLEGAR.            *
   *************************************************************************/

int  escribe_video(int x, int y, char *p, int atrib, int color1, int color2);
  /*************************************************************************
   *RUTINA QUE ESCRIBE UNA CADENA DE CARACTERES EN LA PANTALLA             *
   *  x,y    = POSICIÓN EN LA PANTALLA                                     *
   *  p      = MENSAJE A DESPLEGAR                                         *
   *  atrib  = COLOR DE FONDO                                              *
   *  color1 = COLOR CON EL QUE SERÁN DESPLEGADOS LOS CARACTERES NORMALES  *
   *  color2 = COLOR CON EL QUE SERÁN DESPLEGADOS LOS CARACTERES ESPECIALES*
   *************************************************************************/

void coloca_sombra(int longitud, int x, int y);
  /*************************************************************************
   *    COLOCA UNA SOMBRA (FONDO NEGRO) EN LA PANTALLA, LOS PARÁMETROS     *
   *    QUE NECESITA SON :                                                 *
   *    longitud = NUMERO DE CARACTERES A CAMBIAR DE FONDO                 *
   *    x,y      = POSICIÓN DE LA PANTALLA                                 *
   *************************************************************************/

void margen(int longitud, int fondo, int x, int y, int color, int c1, int c2,
     int c3);
  /*************************************************************************
   *  RUTINA QUE COLOCA LA COLUMNA SUPERIOR O INFERIOR DE UN RECUADRO EN   *
   *  LA POSICIÓN INDICADA Y CON LOS ATRIBUTOS INDICADOS:                  *
```

```
    *      longitud = LONGITUD EN EL EJE X DEL CUADRO                    *
    *      fondo    = COLOR DE FONDO DEL CUADRO                          *
    *      x,y      = ESQUINA SUPERIOR IZQUIERDA DONDE SE COLOCARA EL CUADRO*
    *      color    = COLOR DEL CARACTER                                 *
    *      c1,c2,c3 = CÓDIGOS ASCII DE LOS CARACTERES QUE SERÁN DESPLEGADOS *
    *                 PARA EL MARGEN                                    *
    ********************************************************************/

void marco(int x1, int y1, int x2, int y2, int color_fondo, int caracter,
     int sombra, char titulo[]);
  /*******************************************************************
    *    RUTINA QUE COLOCA UN MARCO EN LA POSICIÓN INDICADA Y CON LOS  *
    *    ATRIBUTOS INDICADOS, LOS PARÁMETROS NECESARIOS SON:           *
    *      x1,y1 = ESQUINA SUPERIOR IZQUIERDA DEL CUADRO               *
    *      x2,y2 = ESQUINA INFERIOR DERECHA DEL CUADRO                 *
    *      color_fondo = COLOR DE FONDO DEL CUADRO                     *
    *      caracter    = COLOR DEL CARACTER                            *
    *      sombra      = INDICA SI SE DESEA DAR LA APARIENCIA DEL QUE EL *
    *                    CUADRO  PROYECTA UNA SOMBRA SOBRE LA PANTALLA *
    *                            0 = NO PROYECTA SOMBRA                *
    *                            1 = SI PROYECTA SOMBRA                *
    *      titulo      = TITULO DEL RECUADRO                           *
    ********************************************************************/

int  esta_en(char *a, char c);
  /*******************************************************************
    * RUTINA QUE COMPRUEBA SI UN CARACTER ESTA EN UNA CADENA, SI ESTA  *
    *  RETORNA SU POSICIÓN, DE LO CONTRARIO, RETORNA 0                 *
    ********************************************************************/

int  boton(int c[][4], int numero);
  /*******************************************************************
    * Función que comprueba si es presionado el botón izquierdo del raton *
    *  y si la posición donde se presionó corresponde en un rango      *
    *  comprendido en la variable ratón                                *
    ********************************************************************/


void llena_raton(int raton[][4], int x, int y, int i, int longitud);
  /*******************************************************************
    * llena un arreglo con los rangos de las posiones dónde se puede   *
    *  presionar el boton izquierdo del ratón.                         *
    ********************************************************************/

int  respuesta_hor(int x[], int y, int contador, char *menu[], char *teclas,
     int norm, int inv, int c1, int c2, int raton[][4]);
  /*******************************************************************
    *  Función que obtiene la respuesta de un menu Horizontal, los     *
    *  parámetros necesario son :                                      *
    *  x,y      = Posición de la primera opción del menu               *
    *  contador= Numero de opciones en el menú                         *
    *  menu     = Arreglo de cadena que contiene las opciones del menú *
    *  teclas   = Teclas asociadas a cada una de las opciones del menú *
    *  norm     = Color del fondo para el video Normal                 *
    *  inv      = Color de fondo para el video inverso                 *
    *  c1       = Color del caracter en video normal                   *
    *  c2       = Color de los caracteres especiales que representa a  *
    *                las opciones                                      *
    *  raton    = Un arreglo con las posiciones de pantalla virtual de *
    *               cada una de las opciones del menú                  *
    ********************************************************************/
```

```
int  localiza_longitud(char *menu[], int elementos, int tipo);
  /*************************************************************************
   * encuentra la longitud mayor de un arreglo de cadenas                  *
   *************************************************************************/

void obten_posicion(char *menu[], int pos[], int elementos, int x);

int  menu_hor(char *menu[], int x, int y, int fondo, int inverso, int color1,
       int color2, char titulo[], int sombra);
  /*************************************************************************
   *    Función que despliega un menú Horizontal, los parámetros           *
   *        que necesita son los siguientes:                               *
   *    menu  = es un arreglo bidimensional que contiene los datos del menu *
   *    x,y   = coordenadas a colocar el menu                              *
   *    fondo = Color del fondo en video Normal                            *
   *    inverso = Color de fondo del video inverso                         *
   *    color1  = Color de las letras                                      *
   *    color2  = Color de las letras asociadas a las opciones             *
   *    titulo  = Titulo que se colocara en la parte superior del menú     *
   *    sombra= 0 no coloca sombra                                         *
   *          1 si coloca sombra                                           *
   *                                                                       *
   *            los claves de los colores son las siguientes:              *
   *                  colores de fondo válidos                             *
   *    16 = fondo azul              64  = fondo rojo                      *
   *    32 = fondo verde             80  = fondo rosa                      *
   *    48 = fondo azul claro        112 = fondo blanco                    *
   *                                                                       *
   *            colores para los caracteres                                *
   * 1 = azul            7 = blanco              13 = rosa intenso         *
   * 2 = verde           8 = gris                14 = amarillo intenso     *
   * 3 = azul claro      9 = azul intenso       15 = blanco intenso        *
   * 4 = rojo fuerte    10 = verde intenso     +128 = caracter parpadeante*
   * 5 = rosa           11 = azul claro intenso                           *
   * 6 = rojo claro     12 = rojo intenso                                 *
   *************************************************************************/

int  respuesta_ver(int x, int y, int contador, char *menu[], char *teclas,
       int norm_video, int inv_video, int color1, int color2, int raton[][4]);
  /*************************************************************************
   *  Función que obtiene la respuesta de un menu Vertical, los            *
   *  parámetros necesario son :                                           *
   *  x,y      = Posición de la primera opción del menu                    *
   *  contador= Numero de opciones en el menú                              *
   *  menu     = Arreglo de cadena que contiene las opciones del menú      *
   *  teclas   = Teclas asociadas a cada una de las opciones del menú      *
   *  norm_video = Color del fondo para el video Normal                    *
   *  inv_video  = Color de fondo para el video inverso                    *
   *  color1     = Color del caracter en video normal                      *
   *  color2     = Color de los caracteres especiales que representa a     *
   *               las opciones                                            *
   *  raton      = Un arreglo con las posiciones de pantalla virtual de    *
   *               cada una de las opciones del menú                       *
   *************************************************************************/

int  menu_ver(char *menu[], int x, int y, int fondo, int inverso, int c,
       int c2, char titulo[], int sombra);
  /*************************************************************************
   *    Función que despliega un menú Vertical, los parámetros que         *
   *        necesita son los siguientes:                                   *
   *    menu  = es un arreglo bidimensional que contiene los datos del menu *
   *    x,y   = coordenadas a colocar el menu                              *
   *    fondo = Color del fondo en video Normal                            *
```

```
   *    inverso = Color de fondo del video inverso                          *
   *    c       = Color de las letras                                       *
   *    c2      = Color de las letras asociadas a las opciones              *
   *    titulo  = Titulo que se colocara en la parte superior del menú      *
   *    sombra= 0 no coloca sombra                                          *
   *          1 si coloca sombra                                            *
   *                                                                        *
   *            los claves de los colores son las siguientes:               *
   *                    colores de fondo válidos                            *
   *    16 = fondo azul             64  = fondo rojo                        *
   *    32 = fondo verde            80  = fondo rosa                        *
   *    48 = fondo azul claro       112 = fondo blanco                      *
   *                                                                        *
   *              colores para los caracteres                               *
   * 1 = azul              7 = blanco              13 = rosa intenso        *
   * 2 = verde             8 = gris                14 = amarillo intenso    *
   * 3 = azul claro        9 = azul intenso        15 = blanco intenso      *
   * 4 = rojo fuerte      10 = verde intenso      +128 = caracter parpadeante*
   * 5 = rosa             11 = azul claro intenso                           *
   * 6 = rojo claro       12 = rojo intenso                                 *
   **************************************************************************/

void salva_pantalla(int x1, int y1, int x2, int y2);
  /*************************************************************************
   *  Almacena la información contenido en el recuadro formado por        *
   *  las coordenadas de la esquina superior izquierda (x1,y1) y la        *
   *  esquina inferior derecha (x2,y2).                                    *
   *         SOLAMENTE FUNCIONA PARA MODO TEXTO                            *
   *************************************************************************/

void coloca_pantalla(void);
  /*************************************************************************
   *  Coloca la última pantalla almacenada por la función salva_pantalla() *
   *         SOLAMENTE FUNCIONA PARA MODO TEXTO                            *
   *************************************************************************/

void salva_pan_grafica(int x1, int y1, int x2, int y2);
  /*************************************************************************
   *  Almacena la información contenido en el recuadro formado por        *
   *  las coordenadas de la esquina superior izquierda (x1,y1) y la        *
   *  esquina inferior derecha (x2,y2).                                    *
   *         SOLAMENTE FUNCIONA PARA MODO GRÁFICO                          *
   *************************************************************************/

void coloca_pan_grafica(void);
  /*************************************************************************
   *Coloca la última pantalla almacenada por la función salva_pan_grafica()*
   *         SOLAMENTE FUNCIONA PARA MODO GRÁFICO                          *
   *************************************************************************/

int  centra(char linea[], char men[], int longitud);

void mensaje2(char men[], int x1, int y1, int color_fondo, int caracter,
     int sombra, int opción);
  /*************************************************************************
   * Rutina que despliega un mensaje en la pantalla, los parámetros son:   *
   *    men          = Mensaje que se desea desplegar                      *
   *    x1,y1        = Esquina superior izquierda del marco del mensaje    *
   *    color_fondo= Color de fondo con el que se desplegara el mensaje    *
   *    caracter     = Color con el que serán desplegados los caracteres   *
   *    sombra       = Bandera que representa si se desea proyectar sombra *
   *          SOMBRA =0  NO      SOMBRA =1  SI                             *
   *    opción       = puede tener dos posibles valores                    *
```

```
*                0 -> almacena la pantalla donde se va a colocar el     *
*              mensaje y lo despliega                                    *
*               1 -> Restaura la pantalla donde se había colocado el     *
*                    mensaje                                             *
*************************************************************************/


void men_esp(int x, int y, char *men, int fon, int color, int sombra);
 /************************************************************************
  * Despliega un mensaje en la pantalla y espera a que el usuario pulse *
  * alguna tecla, los parámetros necesarios son:                         *
  *   x,y     = Posición del mensaje                                     *
  *   men     = mensaje a desplegar                                      *
  *   fon     = Color de fondo                                           *
  *   color   = Color de los caracteres                                  *
  *   sombra = Bandera que indica si se desea desplegar la sombra        *
  ************************************************************************/

int  mensaje(char *men[], char *elige[], int sombra, int fondo, int inv,
     int c2, int c);
 /************************************************************************
  * Rutina que despliega un mensaje en la pantalla, los parámetros son: *
  *    men          = Mensaje que se desea desplegar                     *
  *    elige        = opciones que se desean colocar en el mensaje       *
  *    sombra       = Bandera que representa si se desea proyectar sombra *
  *          SOMBRA =0  NO      SOMBRA =1  SI                            *
  *    fondo        = Color de fondo con el que se desplegara el mensaje *
  *    inv          = Color de fondo para el video inverso               *
  *    c2           = Color con el que serán desplegados los caracteres  *
  *    c            = Color de las letras asociadas a las opciones       *
  ************************************************************************/

void indica_modo(int insert);

void quita(int *pos, char var[], int x1, int y1, int opción);

int  lee_dato(char campo[], int digitos, int x, int y, char tipo);
 /************************************************************************
  *  FUNCIÓN QUE LEE UN CAMPO ALFABÉTICO Y REGRESA ALGUNO DE LOS         *
  *  SIGUIENTES VALORES :                                                *
  * -1   SE PRESIONÓ ESC                                                 *
  *  0   SE PRESIONÓ ENTER                                               *
  *  1   SE PRESIONÓ FLECHA ARRIBA                                       *
  *  2   SE PRESIONÓ FLECHA ABAJO                                        *
  *  3   AL FINAL DEL TEXTO Y FLECHA DERECHA                             *
  *  4   AL EMPIEZO DEL TEXTO Y FLECHA IZQUIERDA                         *
  *  5   HOME                                                            *
  *  6   END                                                             *
  *  7   PG_UP                                                           *
  *  8   PG_DOWN                                                         *
  *  9   tecla F10                                                       *
  ************************************************************************/

int  lee_int(int digitos, int *var, int x, int y, int lee);
 /************************************************************************
  * Se encarga de la lectura y/o desplegado de una variable entera      *
  * los parámetros necesarios son:                                      *
  *    digitos = total de digitos a leer                                *
  *    var     = variable tipo entera donde se almacenara el valor      *
  *    x,y     = Posición de la pantalla donde se realizara la lectura  *
  *    lee     = Bandera que indica si solamente despliega el valor o   *
  *              realiza la lectura                                     *
  *              0 = Solamente coloca el valor de la variable           *
```

```
*              1 = Coloca el valor de la variable y realiza su lectura*
 ***************************************************************************/

int  lee_doble(int digitos, int dec, double *var, int x, int y, int lee);
 /***************************************************************************
  * Se encarga de la lectura y/o desplegado de una variable de punto       *
  * flotante con doble precisión los parámetros necesarios son:            *
  *    digitos = total de digitos a leer, incluyendo al punto decimal      *
  *    dec     = numero de decimales a leer                                *
  *    var     = variable tipo entera donde se almacenara el valor         *
  *    x,y     = Posición de la pantalla donde se realizara la lectura     *
  *    lee     = Bandera que indica si solamente despliega el valor o      *
  *              realiza la lectura                                        *
  *              0 = Solamente coloca el valor de la variable              *
  *              1 = Coloca el valor de la variable y realiza su lectura*
  ***************************************************************************/

int  lee_float(int digitos, int dec, float *num, int x, int y, int lee);
 /***************************************************************************
  * Se encarga de la lectura y/o desplegado de una variable de punto       *
  * flotante los parámetros necesarios son:                                *
  *    digitos = total de digitos a leer, incluyendo al punto decimal      *
  *    dec     = numero de decimales a leer                                *
  *    var     = variable tipo entera donde se almacenara el valor         *
  *    x,y     = Posición de la pantalla donde se realizara la lectura     *
  *    lee     = Bandera que indica si solamente despliega el valor o      *
  *              realiza la lectura                                        *
  *              0 = Solamente coloca el valor de la variable              *
  *              1 = Coloca el valor de la variable y realiza su lectura*
  ***************************************************************************/

int  lee_alfa(char campo[], int digitos, int x, int y, int lee);
 /***************************************************************************
  * Se encarga de la lectura y/o desplegado de una variable tipo           *
  * cadena los parámetros necesarios son:                                  *
  *    campo   = variable tipo cadena donde se almacenara la lectura       *
  *    digitos = total de digitos a leer, incluyendo al punto decimal      *
  *    x,y     = Posición de la pantalla donde se realizara la lectura     *
  *    lee     = Bandera que indica si solamente despliega el valor o      *
  *              realiza la lectura                                        *
  *              0 = Solamente coloca el valor de la variable              *
  *              1 = Coloca el valor de la variable y realiza su lectura*
  ***************************************************************************/

void fija_paleta(struct mapa_colores color[]);
 /***************************************************************************
  * fija la paleta de colores para el desplegado de iconos                 *
  ***************************************************************************/

void muestra_icono(short char grafico[32][32], int x, int y);
 /***************************************************************************
  * coloca un icono contenido en la variable gráfico en la posición x,y    *
  ***************************************************************************/

void lee_paleta(char archivo1[], struct mapa_colores color[]);
 /***************************************************************************
  * Realiza la lectura de la paleta de colores de un archivo de icono      *
  ***************************************************************************/

void recupera_icono(char archivo1[], short char grafico[32][32]);
 /***************************************************************************
  * Realiza la lectura de la figura del icono de un archivo de *.ICO       *
  ***************************************************************************/
```

```
int  resp_icono(int elem, int raton[][4]);
  /*************************************************************************
   * Se encarga de verificar cuando se presiona el boton izquierdo del    *
   *   raton y la posición donde se realizo esta contenida dentro de      *
   *   algún rango establecido en el arreglo llamado raton                *
   *************************************************************************/

int  menu_iconos(char *menu[], char *etiq[], int pos[][2]);
  /*************************************************************************
   * Se encarga de colocar una serie de iconos en la pantalla definidos   *
   *  en el arreglo "menu", el cual contiene los nombres de los archivos, *
   *  la etiqueta o leyenda de cada icono esta definida en la variable    *
   *  "etiq", y las posiciones de cada uno de los iconos se encuentra     *
   *  definida en la variable "pos".                                      *
   *************************************************************************/

/************************CÓDIGO DE LAS FUNCIONES************************/

void raton(void)
  {
    r.x.ax=0;  int86(0x33,&r,&r);
    r.x.ax=38; int86(0x33,&r,&r);
    factorx=(r.x.cx+1)/columnas;
    factory=(r.x.dx+1)/renglon;
  }

void raton_grafico(void)
  {
    r.x.ax=0;  int86(0x33,&r,&r);
    r.x.ax=38; int86(0x33,&r,&r);
    factorx=(r.x.cx+1)/getmaxx();
    factory=(r.x.dx+1)/getmaxy();
  }

void coloca_caracter(int pos_x, int pos_y, char caracter, int color_caracter,
                     int color_fondo, int veces)
  {
   int i;
   char cadena[2];
   char far *mem;
   if(MODO==0)
    {
     mem=mem_vid+((pos_y-1)*80*2)+((pos_x-1)*2);
     for(i=0;i<veces;i++)
      {
       *(mem++)=caracter;
       *(mem++)=color_fondo+color_caracter;
      }
    }
   else
    {
     cadena[0]=caracter; cadena[1]='\0';
     setfillstyle(SOLID_FILL,color_fondo/16);
     bar((pos_x-1)*ANCHO,(pos_y-1)*ALTO,(pos_x+veces-1)*ANCHO,(
        pos_y)*ALTO-1);
     setcolor(color_caracter);
     for(i=0;i<veces;i++) outtextxy((pos_x-1+i)*ANCHO,(pos_y-1)*ALTO,cadena);
    }
 }

int escribe_video(int x, int y, char *p, int atrib, int color1, int color2)
   {
    register int i,j;
```

```
    oculta_cursor();
    for(i=x;*p; i++)
      {
       if (*p=='^')
        {
         colcca_caracter(i,y,*++p,color2,atrib,1);
         j=*p++;
        }
       else  coloca_caracter(i,y,*p++,color1,atrib,1);
      }
    muestra_cursor();
    return(j);
   }

void coloca_sombra(int longitud, int x, int y)
 {
   register int i;
   char caracter;
   for (i=0;i<longitud;i++)
    {
      if(MODO==0)
       {
            gotoxy(x,y);
            r.h.ah=8;  r.h.bh=0;
            caracter=int86(0x10,&r,&r);
            coloca_caracter(x++,y,caracter,0,7,1);
       }
      else  coloca_caracter(x+i,y,' ',0,7,1);
    }
 }

void margen(int longitud, int fondo, int x, int y, int color, int c1, int c2,
          int c3)
  {
    coloca_caracter(x,y,c1,color,fondo,1);
    coloca_caracter(x+1,y,c2,color,fondo,longitud-2);
    coloca_caracter(x+(longitud-1),y,c3,color,fondo,1);
  }

void marco(int x1, int y1, int x2, int y2, int color_fondo, int caracter,
     int sombra,char titulo[])
    {
      int longitud=x2-x1+1,i;
      oculta_cursor();
      margen(longitud,color_fondo,x1,y1,caracter,201,205,187);
      for(i=0;i<y2-y1-1;i++)
       {
         coloca_caracter(x1,y1+i+1,186,caracter,color_fondo,1);
         coloca_caracter(x1+1,y1+i+1,' ',caracter,color_fondo,x2-x1-1);
         coloca_caracter(x2,y1+i+1,186,caracter,color_fondo,1);
         if(sombra) coloca_sombra(1,x2+1,y1+i+1);
       }
      margen(longitud,color_fondo,x1,y2,caracter,200,205,188);
      if(sombra)
       {
         coloca_sombra(1,x1+longitud,y2);
         coloca_sombra(longitud,x1+1,y2+1);
       }
      longitud=strlen(titulo);
      i=(((x2-x1)-longitud)/2)+x1;
      escribe_video(i,y1,titulo,color_fondo,caracter,caracter);
      muestra_cursor();
    }
```

```
int esta_en(char *a,char c)
   {
     register int i;
     for(i=0;*a;i++)  if (toupper(*a++)==toupper(c)) return (i+1);
     return 0;
   }

int boton(int c[][4],int numero)
 {
  int retorna=-1,i=0;
  r.x.ax=5; r.x.bx=0; int86(0x33,&r,&r);
  if(r.x.ax==1)
   {
    while(i<numero && retorna==-1)
     {
      if((r.x.cx>=c[i][0] && r.x.cx<=c[i][2]) && (r.x.dx>=c[i][1]
         && r.x.dx<=c[i][3])) retorna=i;
      i++;
     }
    r.x.ax=6; r.x.bx=0; int86(0x33,&r,&r);
    while(r.x.ax!=1) int86(0x33,&r,&r);
   }
  return(retorna);
 }

void llena_raton(int raton[][4],int x,int y,int i,int longitud)
 {
   if(MODO==0)
     {
       raton[i][0]=x*factorx;
       raton[i][1]=y*factory;
       raton[i][2]=raton[i][0]+longitud*factorx;
       raton[i][3]=y*factory;
     }
   else
     {
       raton[i][0]=(x)*ANCHO*factorx;
       raton[i][1]=(y)*ALTO*factory;
       raton[i][2]=raton[i][0]+longitud*ANCHO*factorx;
       raton[i][3]=(y+1)*ALTO*factory;
     }
 }

int respuesta_hor(int x[], int y, int contador, char *menu[], char *teclas,
              int norm, int inv, int c1, int c2, int raton[][4])
    {
      union inkey {char ch[2]; int i; } c;
      int  tecla_elegida,flecha=0,b=-1,fl_ant;
      escribe_video(x[flecha],y,menu[flecha],inv,c1,c2);
      for(;;)
      {
        while(!bioskey(1) && (b=boton(raton,contador))==-1);
        fl_ant=flecha;
        if(b==-1)
         {
          c.i=bioskey(0);
          if(c.ch[0])
           {
            tecla_elegida=esta_en(teclas,toupper(c.ch[0]));
            if(tecla_elegida)
              {
                flecha=tecla_elegida-1;
                escribe_video(x[fl_ant],y,menu[fl_ant],norm,c1,c2);
```

```
                escribe_video(x[flecha],y,menu[flecha],inv,c1,c2);
                return(flecha);
               }
              switch(c.ch[0])
               {
                 case '\r':
                    escribe_video(x[flecha],y,menu[flecha],inv,c1,c2);
                    return flecha;
                 case ' ' : flecha++;  break;
                 case 27  : return -1;
               }
             }
            else
             {
              switch(c.ch[1])
              {
                case 75: flecha--; break;
                case 77: flecha++; break;
                case 80:
                  escribe_video(x[flecha],y,menu[flecha],inv,c1,c2);
                  return flecha;
              }
             }
            if(flecha==contador) flecha=0;
            if(flecha<0) flecha=contador-1;
            if(fl_ant!=flecha)
             {
              escribe_video(x[fl_ant],y,menu[fl_ant],norm,c1,c2);
              escribe_video(x[flecha],y,menu[flecha],inv,c1,c2);
             }
           }
          else
           {
            escribe_video(x[fl_ant],y,menu[fl_ant],norm,c1,c2);
            escribe_video(x[b],y,menu[b],inv,c1,c2);
            return (flecha=b);
           }
        }
      }

int localiza_longitud(char *menu[],int elementos,int tipo)
 {
   int c=0,mayor=0;
   for(c=0;c<elementos;c++)
    {
     if(!tipo) {if(strlen(menu[c])>mayor) mayor=strlen(menu[c]);}
     else mayor=mayor+strlen(menu[c]);
    }
   return(mayor);
 }

void obten_posicion(char *menu[],int pos[],int elementos,int x)
 {
   register int i;
   pos[0]=x;
   for(i=1;i<elementos;i++) pos[i]=pos[i-1]+strlen(menu[i-1]);
 }

int menu_hor(char *menu[], int x, int y, int fondo, int inverso, int color1,
             int color2, char titulo[], int sombra)
  {
    int n,pos[30],raton[30][4],elem=0;
    char teclas[30];
```

```
    while(strcmp(menu[elem],"\0")!=0) elem++;
    obten_posicion(menu,pos,elem,x+1);
    marco(x,y,pos[elem-1]+strlen(menu[elem-1]),y+2,fondo,color1,
          sombra,titulo);
    for (n=0;n<elem;n++)
     {
      teclas[n]=escribe_video(pos[n],y+1,menu[n],fondo,color1,color2);
      llena_raton(raton,pos[n]-1,y,n,strlen(menu[n])-1);
     }
    teclas[n+1]='\0';
    return(respuesta_hor(pos,y+1,elem,menu,teclas,fondo,inverso,color1,
           color2,raton));
  }

int respuesta_ver(int x, int y, int reg, char *menu[], char *teclas, int
      norm_video,int inv_video, int color1, int color2, int raton[][4])
    {
     union inkey { char ch[2];    int i;  } c;
     int tecla_elegida,b=-1,fl=0,fl_ant;
     escribe_video(x,y+fl,menu[fl],inv_video,color1,color2);
     for(;;)
     {
      while(!bioskey(1) && (b=boton(raton,reg))==-1);
      fl_ant=fl;
      if(b==-1)
       {
        c.i=bioskey(0);
        if(c.ch[0])
         {
          tecla_elegida=esta_en(teclas,toupper(c.ch[0]));
          if(tecla_elegida)
           {
            fl=tecla_elegida-1;
            escribe_video(x,y+fl_ant,menu[fl_ant],norm_video,color1,color2);
            escribe_video(x,y+(fl),menu[fl],inv_video,color1,color2);
            return(tecla_elegida-1);
           }
          switch(c.ch[0])
           {
            case '\r':
              escribe_video(x,y+fl,menu[fl],inv_video,color1,color2);
              return fl;
            case ' ' : fl++; break;
            case 27  : return -1;
           }
         }
        else
         {
          switch(c.ch[1])
           {
            case 72: fl--; break;
            case 80: fl++; break;
           }
         }
        if(fl==reg) fl=0;
        if(fl<0) fl=reg-1;
        if(fl!=fl_ant)
         {
          escribe_video(x,y+fl_ant,menu[fl_ant],norm_video,color1,color2);
          escribe_video(x,y+fl,menu[fl],inv_video,color1,color2);
         }
       }
      else
```

```
          {
           fl=b;
           escribe_video(x,y+fl_ant,menu[fl_ant],norm_video,color1,color2);
           escribe_video(x,y+fl,menu[fl],inv_video,color1,color2);
           return fl;
          }
        }
      }

int menu_ver(char *menu[], int x, int y, int fondo, int inverso, int c,
             int c2, char titulo[], int sombra)
  {
   int i,raton[30][4],longitud,elem=0;
   char teclas[30];
   while(strcmp(menu[elem],"\0")!=0) elem++;
   longitud=localiza_longitud(menu,elem,0);
   marco(x,y,x+longitud+1,y+elem+1,fondo,c,sombra,titulo);
   for (i=0;i<elem;i++)
    {
     teclas[i]=escribe_video(x+1,y+i+1,menu[i],fondo,c,c2);
     llena_raton(raton,x,y+i,i,longitud-1);
    }
   teclas[i]='\0';
   return(respuesta_ver(x+1,y+1,elem,menu,teclas,fondo,inverso,c,c2,raton));
  }

void salva_pan_grafica(int x1,int y1,int x2,int y2)
  {
   int i,ymed;
   void *pant;
   unsigned long int tam_ant=TAM_GRAF,tamano;
   if(MODO==1)
    {
     oculta_cursor();
     x1=(x1-1)*ANCHO;  x2=x2*ANCHO;
     y1=(y1-1)*ALTO;  y2=y2*ALTO;
     if(TAM_GRAF==0) a_tempo=fopen(arch_temporal,"wb");
     else a_tempo=fopen(arch_temporal,"a+b");
     TAM_GRAF=TAM_GRAF+5;
     pan_grafica=realloc(pan_grafica,TAM_GRAF*sizeof(long));
     ymed=ceil((y2-y1)/4);
     tamano=imagesize(x1,y1,x2,y1+ymed);
     pan_grafica[tam_ant++]=x1;
     pan_grafica[tam_ant++]=x2;
     pan_grafica[tam_ant++]=y1;
     pan_grafica[tam_ant++]=y2;
     pan_grafica[tam_ant++]=tamano;
     pant=malloc(tamano);
     fseek(a_tempo,0,2);
     for(i=0;i<4;i++)
      {
       getimage(x1,y1+i*ymed,x2,y1+(i+1)*ymed,pant);
       fwrite(pant,tamano,1,a_tempo);
      }
     free(pant);
     fclose(a_tempo);
     muestra_cursor();
    }
  }

void salva_pantalla(int x1,int y1,int x2,int y2)
  {
   int y,x;
```

```
    char far *mem;
    unsigned long int tam_ant=TAMANO,tamano;
    if(MODO==0)
     {
      oculta_cursor();
      tamano=(x2-x1+1)*(y2-y1+1)*2;
      TAMANO=TAMANO+tamano+5;
      pantalla=(short int*)realloc(pantalla,TAMANO*sizeof(short int)+5);
      pantalla[tam_ant++]=255;
      pantalla[tam_ant++]=x1;  pantalla[tam_ant++]=x2;
      pantalla[tam_ant++]=y1;  pantalla[tam_ant++]=y2;
      for(y=y1;y<=y2;y++)
       {
         mem=mem_vid+((y-1)*80*2)+((x1-1)*2);
         for (x=x1;x<=x2;x++)
          {
           pantalla[tam_ant++]=*(mem++);
           pantalla[tam_ant++]=*(mem++);
          }
       }
      muestra_cursor();
     }
   }

void coloca_pan_grafica(void)
 {
   int x1,y1,y2,y,x,i,ymed,maneja;
   char far *mem;
   void *pant;
   unsigned long tam_ant=TAM_GRAF,tam_grafico,tamano,t;
   unsigned posicion;
   if(MODO==1 && TAM_GRAF>0)
    {
     oculta_cursor();
     a_tempo=fopen(arch_temporal,"rb");
     tam_ant=TAM_GRAF=TAM_GRAF-5;
     x1=pan_grafica[tam_ant++];
     tam_ant++;
     y1=pan_grafica[tam_ant++];
     y2=pan_grafica[tam_ant++];
     tamano=pan_grafica[tam_ant++];
     pan_grafica=realloc(pan_grafica,TAM_GRAF*sizeof(long));
     ymed=(y2-y1)/4;
     fseek(a_tempo,0,2);
     t=ftell(a_tempo)-tamano*4;
     if(t<0.0) t=0;
     fseek(a_tempo,t,0);
     pant=malloc(tamano);
     for(i=0;i<4;i++)
      {
       fread(pant,tamano,1,a_tempo);
       putimage(x1,y1+i*ymed,pant,0);
      }
     free(pant);
     fclose(a_tempo);
     if(TAM_GRAF<=0) unlink(arch_temporal);
     else
      {
       maneja=open(arch_temporal,O_RDWR);
       chsize(maneja,t);
       close(maneja);
      }
     muestra_cursor();
```

```
    }
 }

void coloca_pantalla(void)
 {
   int x,y,x1,y1,x2,y2;
   unsigned long int i,j;
   char far *mem;
   if(MODO==0)
     {
       oculta_cursor();
       for(i=TAMANO;pantalla[i]!=255;i--);
       j=i;
       pantalla[i]=0;
       x1=pantalla[++i];
       x2=pantalla[++i];
       y1=pantalla[++i];
       y2=pantalla[++i];
       for(y=y1;y<=y2;y++)
        {
         mem=mem_vid+((y-1)*80*2)+((x1-1)*2);
         for (x=x1;x<=x2;x++)
          {
            *(mem++)=pantalla[++i];
            *(mem++)=pantalla[++i];
          }
        }
       muestra_cursor();
       TAMANO=j;
       pantalla=realloc(pantalla,TAMANO);
     }
 }

int centra(char linea[], char men[],int longitud)
 {
  int l=strlen(men),espacios=0;
  char cadena[50]="                                                  ";
  if(l<longitud);
   {
    espacios=(longitud-l)/2;
    cadena[espacios]='\0';
    sprintf(linea,"%s%s",cadena,men);
   }
  return(espacios);
 }

void mensaje2(char men[], int x1, int y1, int color_fondo, int caracter,
              int sombra, int opcion)
   {
    static int x2,y2;
    int tamano,tot_pan;
    if(opcion==0)
     {
      x2=x1+strlen(men)+2;
      y2=y1+2;
      if(MODO==0) salva_pantalla(x1,y1,x2+1,y2+1);
      else salva_pan_grafica(x1,y1,x2+1,y2+1);
      marco(x1,y1,x2,y2,color_fondo,caracter,sombra,"");
      escribe_video(x1+1,y1+1,men,color_fondo,caracter,caracter);
     }
    else
     {
      if(MODO==0) coloca_pantalla();
```

```
      else coloca_pan_grafica();
     }
   }

void men_esp(int x, int y, char *men, int fon, int color, int sombra)
 {
  mensaje2(men,x,y,fon,color,sombra,0);
  bioskey(0);
  mensaje2(men,x,y,fon,color,sombra,1);
 }

int mensaje(char *men[],char *elige[],int sombra,int fondo,int inv,int c2,
            int c)
 {
   int i,raton[30][4],longitud,elem=0,x,y,tamano,espacios,pos[5],elem2=0,n;
   char teclas[8]=" ",linea[80];
   unsigned size;
   while(strcmp(men[elem],"\0")!=0) elem++;
   longitud=localiza_longitud(men,elem,0);
   x=(80-longitud)/2;
   y=(25-elem)/2;
   if(MODO==0) salva_pantalla(x,y,x+longitud+2,y+elem+3);
   else  salva_pan_grafica(x,y,x+longitud+2,y+elem+3);
   marco(x,y,x+longitud+1,y+elem+2,fondo,c,sombra,"");
   for (i=0;i<elem;i++)
    {
     centra(linea,men[i],longitud);
     escribe_video(x+1,y+i+1,linea,fondo,c,c2);
    }
   while(strcmp(elige[elem2],"\0")!=0) elem2++;
   obten_posicion(elige,pos,elem2,x+1);
   linea[0]='\0';
   for (n=0;n<elem2;n++) strcat(linea,elige[n]);
   espacios=centra(linea,linea,longitud);
   for (n=0;n<elem2;n++)
    {
     pos[n]=pos[n]+espacios;
     teclas[n]=escribe_video(pos[n],y+i+1,elige[n],fondo,c,c2);
     llena_raton(raton,pos[n]-1,y+i,n,strlen(elige[n])-1);
    }
   teclas[n+1]='\0';
   i=respuesta_hor(pos,y+i+1,3,elige,teclas,fondo,inv,c,c2,raton);
   if(MODO==0) coloca_pantalla();
   else coloca_pan_grafica();
   return(i);
 }

void indica_modo(int insert)
 {
  if(insert==0) escribe_video(30,25," MODO DE SUSTITUCION ",64,15,15);
  else escribe_video(30,25,"  MODO DE INSERCION  ",64,15,15);
 }

void quita(int *pos,char var[],int x1,int y1,int opcion)
 {
  int i;
  if(*pos>0 || opcion==1)
   {
    *pos=((opcion==0)? *pos-1 : *pos);
    for(i=*pos;i<strlen(var)-1;i++) var[i]=var[i+1];
    var[i]=' ';
   }
  escribe_video(x1,y1,var,cfondo2,ctexto,ctexto);
```

```
}
int lee_dato(char campo[],int digitos,int x,int y, char tipo)
{
  int posl=0,tecla,k;
  char a[]={"ABCDEFGHIJKLMNOPQRSTUVWXYZ?¿.-_abcdefghijklmnopqrstuvwxyz ");
  char n[]={"1234567890.-+ ");
  char i[]={"1234567890-+ ");
  char t[]={"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
            7890?_-+*<>\\|@!$%&/()=?¿'\":;,. ");
  union inkey { char ch[2]; int i; } c;
  escribe_video(x,y,campo,cfondo2,ctexto,ctexto);
  for(;;)
   {
    coloca_caracter(x+posl,y,campo[posl],ctexto,cfondo3,1);
    while(!bioskey(1));
    coloca_caracter(x+posl,y,campo[posl],ctexto,cfondo2,1);
    c.i=bioskey(0);
    if(c.ch[0])
     {
      switch(tipo)
       {
        case 'A' : tecla=esta_en(a,c.ch[0]); break;
        case 'N' : tecla=esta_en(n,c.ch[0]); break;
        case 'C' : tecla=esta_en(t,c.ch[0]); break;
        case 'I' : tecla=esta_en(i,c.ch[0]); break;
       }
      if(tecla)
       {
        if(insert)
         {
          for(k=digitos-1;k!=posl;k--) campo[k]=campo[k-1];
          escribe_video(x,y,campo,cfondo2,ctexto,ctexto);
         }
        campo[posl]=(c.ch[0]);
        coloca_caracter(x+posl,y,c.ch[0],ctexto,cfondo2,1);
        if(posl<digitos-1) posl++;
       }
      switch(c.ch[0])
       {
        case '\b': quita(&posl,campo,x,y,0); break; //Tecla Backspace
        case '\r': return(0);                       //Tecla ENTER
        case 27  : return(-1);                      //Tecla ESCAPE
       }
     }
    else
     {
      switch(c.ch[1])
       {
        case 68: return(9);                         //tecla F10
        case 71: return(5);                         //tecla HOME
        case 72: return(1);                         //tecla flecha arriba
        case 73: return(8);                         //tecla PG_UP
        case 75:                                    //tecla flecha derecha
          if(posl>0) posl--;
          else return(4);                           //tecla home
          break;
        case 77:                                    //tecla flecha izquierda
          if(posl<digitos-1) posl++;
          else return(3);
          break;
        case 79: return(6);                         //tecla END
        case 81: return(7);                         //tecla PG_DOWN
```

```
        case 80: return(2);                          //tecla flecha abajo
        case 82: insert=((insert==0)? 1 : 0);        //tecla insert
           indica_modo(insert); break;
        case 83: quita(&pos1,campo,x,y,1); break; //tecla SUPRIMIR
       }
     }
   }
 }

int lee_int(int digitos,int *var,int x, int y,int lee)
 {
  char campo1[25],formato[10];
  int regresa;
  sprintf(formato,"%%%dd",digitos);
  sprintf(campo1,formato,*var);
  if(strlen(campo1)>digitos) { *var=0;  sprintf(campo1,formato,*var);}
  escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
  if(lee)
   {
    regresa=lee_dato(campo1,digitos,x,y,'I');
    *var=atoi(campo1);
    sprintf(campo1,formato,*var);
    if(strlen(campo1)>digitos) { *var=0; sprintf(campo1,formato,*var); }
    escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
   }
  return(regresa);
 }

int lee_doble(int digitos,int dec,double *var,int x, int y,int lee)
 {
  char campo1[25],formato[15];
  int regresa;
  sprintf(formato,"%%%d.%dlf",digitos,dec);
  sprintf(campo1,formato,*var);
  if(strlen(campo1)>digitos) { *var=0; sprintf(campo1,formato,*var);}
  escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
  if(lee)
   {
    regresa=lee_dato(campo1,digitos,x,y,'N');
    *var=atof(campo1);
    sprintf(campo1,formato,*var);
    if(strlen(campo1)>digitos) {*var=0; sprintf(campo1,formato,*var);}
    escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
   }
  return(regresa);
 }

int lee_float(int digitos,int dec,float *num,int x, int y,int lee)
 {
  char campo1[25],formato[10];
  int regresa;
  sprintf(formato,"%%%d.%dlf",digitos,dec);
  sprintf(campo1,formato,*num);
  if(strlen(campo1)>digitos) { *num=0;  sprintf(campo1,formato,*num);}
  escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
  if(lee)
   {
    regresa=lee_dato(campo1,digitos,x,y,'N');
    *num=atof(campo1);
    sprintf(campo1,formato,*num);
    if(strlen(campo1)>digitos) { *num=0; sprintf(campo1,formato,*num); }
    escribe_video(x,y,campo1,cfondo1,ctexto,ctexto);
   }
```

```
  return(regresa);
 }

int lee_alfa(char campo[],int digitos,int x, int y,int lee)
 {
  int regresa,i;
  if(strlen(campo)>digitos) campo[digitos]='\0';
  else
   {
    for(i=strlen(campo);i<digitos;i++) campo[i]=' ';
    campo[i]='\0';
   }
  escribe_video(x,y,campo,cfondo1,ctexto,ctexto);
  if(lee)
   {
    regresa=lee_dato(campo,digitos,x,y,'C');
    escribe_video(x,y,campo,cfondo1,ctexto,ctexto);
   }
  return(regresa);
 }

void fija_paleta(struct mapa_colores color[])
{
  struct palettetype pal;
  int i;
  getpalette(&pal);
  for(i=0;i<16;i++)
     setrgbpalette(pal.colors[i],color[i].rojo/4,color[i].verde/4,
                   color[i].azul/4);
 }

void muestra_icono(short char grafico[32][32],int x,int y)
{
  int i,j;
  for(i=0;i<32;i++) { for(j=0;j<32;j++)  putpixel(x+j,y+i,grafico[i][j]); }
 }

void lee_paleta(char archivo1[],struct mapa_colores color[])
 {
   FILE *arch1;
  int i;
  if((arch1=fopen(archivo1,"rb"))==NULL)
    { printf(" ERROR FALTA UN ARCHIVO DE UN ICONO "); exit(1);}
  else
   {
    fseek(arch1,62,0);
    for(i=0;i<16;i++) fread(&color[i],sizeof(color[i]),1,arch1);
   }
  fclose(arch1);
 }

void recupera_icono(char archivo1[],short char grafico[32][32])
 {
   FILE *arch1;
  int i,j,punto,punto1,punto2;
  if((arch1=fopen(archivo1,"rb"))==NULL)
    { printf(" ERROR FALTA UN ARCHIVO DE UN ICONO "); exit(1);}
  else
   {
    fseek(arch1,126,0);
    for(i=31;i>=0;i--)
     {
      for(j=0;j<16;j++)
```

```
        {
          fread(&punto,sizeof(short char),1,archl);
          puntol=punto2=punto;
          puntol <<= 4; puntol>>=4;  punto2 >>= 4;
          grafico[i][j*2]=punto2;  grafico[i][j*2+1]=puntol;
        }
      }
    }
   fclose(archl);
  }

int resp_icono(int elem,int raton[][4])
  {
   int b;
   for(;;)
    {
     while((b=boton(raton,elem))==-1)  (if(bioskey(1)) bioskey(0);)
     return b;
    }
  }

int menu_iconos(char *menu[],char *etiq[],int pos[][2])
  {
   int i,raton[30][4],elem=0;
   char archivol[15];
   short char grafico[32][32];
   struct mapa_colores color[16];
   while(strcmp(menu[elem],"\0")!=0) elem++;
   settextstyle(2,0,0);
   for (i=0;i<elem;i++)
    {
     sprintf(archivol,"%s.ICO",menu[i]);
     recupera_icono(archivol,grafico);
     if(i==0)
      {
       lee_paleta(archivol,color);
       fija_paleta(color);
      }
     muestra_icono(grafico,pos[i][0],pos[i][1]);
     setcolor(0);
     outtextxy(pos[i][0],pos[i][1]+34,etiq[i]);
     raton[i][0]=pos[i][0]*factorx;
     raton[i][1]=pos[i][1]*factory;
     raton[i][2]=(pos[i][0]+32)*factorx;
     raton[i][3]=(pos[i][1]+32)*factory;
    }
   settextstyle(0,0,0);
   return(resp_icono(elem,raton));
  }

void inicializa(int modo)
{
  int gdriver=VGA, gmode,errorcode;
  if(modo==0) gmode=VGAMED;
  else gmode=VGAHI;
  initgraph(&gdriver,&gmode,"");
  errorcode=graphresult();
  if(errorcode!=grOk)
   {
    printf("\nERROR, NO SE ENCUENTRA EL DRIVER GRAFICO EGAVGA.BGI");
    exit(1);
   }
  setfillstyle(SOLID_FILL,15);
```

```
  bar(0,0,getmaxx(),getmaxy());
  ALTO=textheight("");
  ANCHO=textwidth("");
  MODO=1;
}
```

APÉNDICE B

EJEMPLO

En esta sección se ejemplificar el uso de los elementos de interface y de las rutinas generadas a lo largo de la tesis aplicado a la solución de una parte de un problema; solamente se analizará una parte debido a que es un problema muy complejo; y analizarlo a fondo y proponer soluciones demandaría bastante tiempo y personal, además de que no es este el tema ni parte fundamental de los objetivos de la tesis, solamente se utilizara como ejemplo de la forma de implementar los elementos de interface en los programas, y obtener mejores resultados y programas mas competitivos, los cuales sean del agrado del usuario.

La Dirección General de Carreteras Federales (DGCF), tiene una metodología para la construcción de una nueva carretera; para lo cual debe de seguirse una serie de pasos o etapas para llegar al resultado final, entre ellos podemos mencionar los siguientes:

1.- estudios preliminares
2.- Salección de Ruta
3.- Levantamiento topográfico
4.- Creación de Anteproyecto
5.- Proyecto

A continuación explicaremos brevemente, de una manera muy superficial y sencilla cada una de las etapas anteriores, sin entrar mucho en los detalles, ya que se puede ahondar bastante en cada uno de estos puntos con formulas, ejemplo, reglas, lineamentos, procedimientos, etc., lo cual no es el objetivo principal de esta sección; posteriormente, se seleccionara una parte de alguno de estos puntos y se plantearan soluciones usando los elementos de interface en su diseño, con el fin de que el usuario pueda apreciar la forma de implementarlo.

1.- ESTUDIOS PRELIMINARES

Comprende los estudios socioeconómicos que justifican la construcción de nuevos caminos y/o las mejoras de los existentes; además, debe de tomarse en cuenta los puntos obligados por los cuales debe de pasar la carretera, entendiéndose por puntos obligados aquellos sitios por los que necesariamente debe de pasar el camino, ya sea por razones económicas, sociales o políticas; con los cuales debe de justificarse la construcción de la carretera.

2.- SELECCIÓN DE RUTA

En la construcción de una carretera influyen una serie de aspectos, los cuales deben de tomarse en cuenta; entre ellos podemos mencionar la topografía, la geología, la hidrología, el drenaje, el uso de la tierra, etc.; todos estos datos, en conjunto con los datos de tránsito, constituyen la información básica para el proyecto de la obra y la selección de la ruta; además, debe recopilarse información sobre las obras existentes.

También se puede basar en el uso de cartas geográficas, para darse una idea de la topografía de la zona; otra forma de reconocer el terreno por donde se piensa construir el proyecto es a base de lo que se conoce como "reconocimiento aéreo", y la mas utilizada de todas ellas, es a base de fotografía aérea, las cuales facilitan el estudio del terreno desde los puntos de vista topográficos y geológicos, permitiendo determinar la elección de una mejor ruta; este tipo de fotografía, utiliza equipo especial tanto de película, como de papel para la impresión.

En base a todos estos elementos, y con un grupo de expertos, los cuales interpretan los datos anteriores, se determinan algunas alternativas para la construcción del proyecto, se evalúan todas ellas y se elige solamente una de ellas.

3.- LEVANTAMIENTO TOPOGRÁFICO

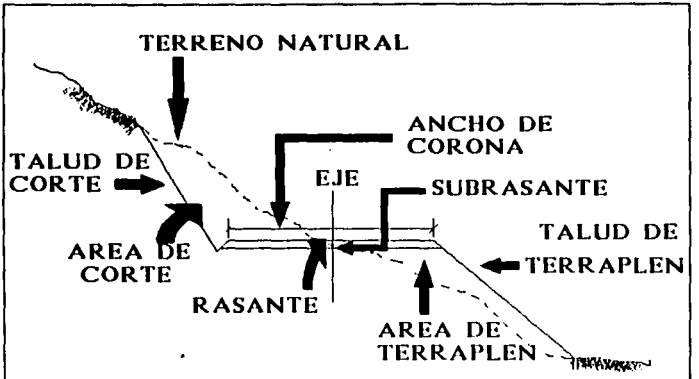
Una vez establecida la ruta por donde se construirá la carretera, se procede a complementar y definir los datos sobre el terreno; esto se realiza mediante un levantamiento topográfico para obtener la geometría exacta del terreno por donde se construirá la carretera; también pueden emplearse equipo fotogrametrico para obtener estos datos, si lo permite la topografía y vegetación de la zona.

4.- ANTEPROYECTO

Con base a todos los datos anteriores, se inicia el estudio para el trazo del camino, para lo cual, el proyectista, debe de observar la geometría del terreno, para obtener el eje que seguirá el camino, al cual se le denomina "ALINEAMIENTO HORIZONTAL", el cual es una proyección sobre el eje horizontal del eje del camino, y al cual esta compuesto por tangentes, curvas circulares y curvas de transición; siendo las tangentes las rectas que unen dos curvas; una vez establecido este alineamiento horizontal, se continua con la definición del "ALINEAMIENTO VERTICAL", el cual es la proyección del eje de la carretera sobre un plano vertical; al eje del proyecto en alineamiento vertical se le denomina "SUBRASANTE"; al igual que el alineamiento horizontal, el alineamiento vertical esta compuesto de tangentes y curvas. A cada punto dentro del alineamiento vertical donde se unen 2 tangentes se le denomina PIV, al punto donde se une el primer punto de la curva con la tangente se le denomina PCV, y al punto de unión entre el ultimo punto de la curva y la tangente se le denomina PTV.

5.- PROYECTO DEFINITIVO

Una vez establecidas las líneas que conforman los alineamientos horizontal y vertical, el siguiente paso consiste en definir las características geométricas del camino y las propiedades de los materiales que lo formarán, dentro de las características geométricas del camino se tiene que definir el tamaño de la corona y la inclinación de los taludes de cortes y terraplén.



SECCION TRANSVERSAL TIPICA
GEOMETRIA DEL PROYECTO

Una vez definidos tanto la geometría del terreno como la geometría del proyecto, se procede a realizar cálculos relacionados con los volúmenes de construcción tanto en corte como en terraplén. También se incluye en esta parte el cálculo de movimientos de terracerias por medio del diagrama denominado "curva masa".

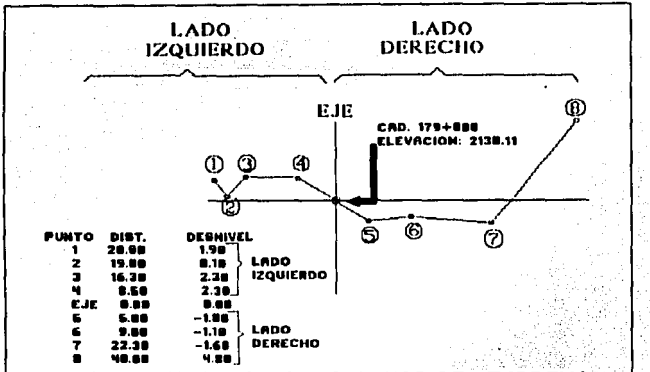
PARTE A ANALIZAR.

Vamos a analizar, una parte correspondiente al anteproyecto; el cual representa el comienzo de la entrada de datos hacia la computadora para su proceso y verificación; primeramente, tenemos los datos obtenidos del levantamiento topográfico, lo cuales se obtienen por "brigadas", las cuales se encargan de realizar las mediciones correspondientes en los lugares indicados, con el fin de

obtener la geometría exacta del terreno; con lo cual los cálculos resultarán mucho mas precisos; estos datos son almacenados en una FORMA como la siguiente:

LADO IZQUIERDO				CAD. ELEV.	LADO DERECHO				
	20.00 1.90	19.00 0.10	16.30 2.30	8.50 2.30	179000.00 2138.11	5.00 -1.80	9.00 -1.10	22.30 -1.60	40.00 4.80	
	20.00 3.40	17.60 1.00	16.20 3.30	4.40 1.60	179009.00 2131.90	4.00 0.50	14.40 0.10	25.40 -2.30	37.80 -6.80	40.00 -6.50
	20.00 2.40	13.70 1.70	11.00 -0.10	9.70 1.00	179020.00 2135.37	6.40 0.00	16.00 -1.90	40.00 -3.40		
	.	20.00 2.20	5.40 0.60	1.70 -0.40	179020.00 2134.38	7.10 -1.30	20.00 -1.60	40.00 -2.90		
			20.00 5.00	6.00 1.70	179029.00 2133.49	20.00 -0.80	40.00 -2.10			
			20.00 0.40	16.00 1.50	179031.50 2134.08	1.70 0.30	4.40 -1.90	7.50 -1.30	20.00 -1.60	40.00 -3.20
			20.00 1.70	3.50 0.20	179040.00 2133.13	16.00 -1.60	29.80 -1.80	40.00 -3.20		
	20.00 1.80	5.00 0.40	3.30 -1.00	1.10 -1.00	179060.00 2131.02	7.10 -0.10	24.60 -4.00	40.00 -5.30		
			20.00 2.80	3.20 1.10	179063.00 2129.58	2.50 0.00	5.80 0.50	22.20 -2.70	40.00 4.00	

Los puntos de lado izquierdo, son tomados como puntos negativos al eje del proyecto, los puntos del lado derecho son tomados como puntos positivos; el punto en donde se intersecta el eje con el terreno, define el centro de nuestro eje; ahora, solamente tienen que graficarse estos puntos en plano coordenado; los cuales definirán la forma exacta del terreno por donde se proyectara la construcción de la nueva carretera.



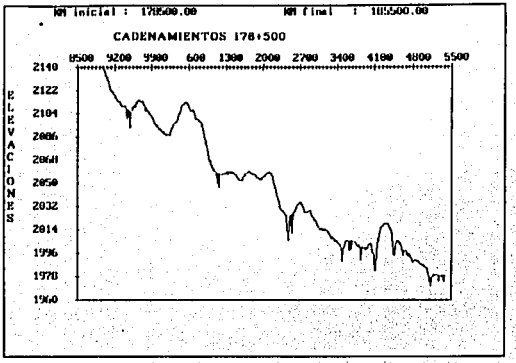
REPRESENTACION DE LOS DATOS DEL TERRENO

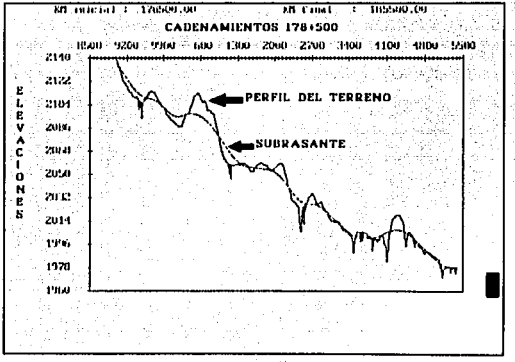
Estos datos tiene que ser capturados o pueden ser obtenidos mediante el uso de equipo de fotogrametría, si es posible; además, tiene que ser acompañados de un archivo de datos generales, los cuales identifiquen a que camino pertenecen los datos anteriores.

Al terminar de capturar los datos del terreno, estos tienen que ser mostrados gráficamente en la pantalla para que el proyectista pueda observar la geometría del terreno, y pueda hacer una mejor estimación de la mejor alternativa para la construcción de la carretera.

Otro aspecto importante que hay que resaltar, es que con los mismos datos anteriores, puede obtenerse también lo que se conoce como "PERFIL DEL TERRENO", el cual esta definido por los puntos donde se intersecta el terreno con el eje del proyecto, con ellos, puede observarse la forma que tiene el terreno a lo largo del eje del proyecto.

Una vez establecidos todos estos aspectos, se debe de proponer una línea, la cual fijara el eje de la construcción de la carretera, la cual se le llama "SUBRASANTE", y esta constituida por curvas parabólicas.


KM inicial : 178500.00
KM final : 185500.00
CADENAMIENTOS 178+500
8500 9200 9900 600 1300 2000 2700 3400 4100 4800 5500
2140
2122
2104
2086
2068
2050
2032
2014
1996
1978
1960
ELEVACIONES


CADENAMIENTOS 178+500
PERFIL DEL TERRENO
SUBRASANTE
ELEVACIONES

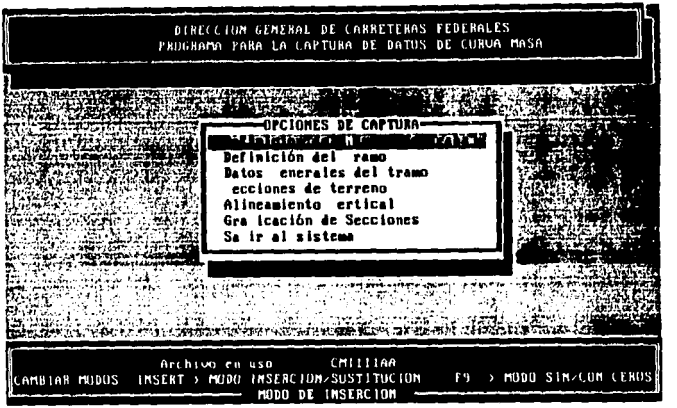
Los datos que deben de capturarse para poder establecer esta línea son los siguientes:

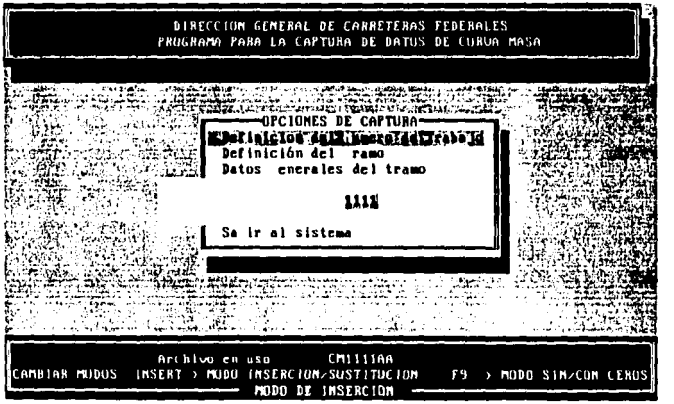
CADENAMIENTO	ELEVACIÓN	LONGITUD
179000.00	2135.66	0.00
179420.00	2110.46	340.00
179750.00	2108.15	260.00
180130.00	2092.95	280.00
180570.00	2101.75	600.00
181320.00	2056.75	340.00
182060.00	2052.55	400.00
182480.00	2027.35	260.00
182820.00	2026.84	260.00
183460.00	2002.84	140.00
183880.00	2000.74	120.00
184360.00	2010.10	440.00
185380.00	1978.54	0.00

Hasta aquí ilustraremos la forma de resolver este problema, utilizando las rutinas de los elementos de interface creados en esta tesis. Como inicio del programa, podemos representar los derechos de autor y/o función del programa con la ayuda de un mensaje; posteriormente, procederemos a desplegar un menú con todas las opciones disponibles que se podrán utilizar durante la sesión; los archivos que se utilizarán deberán de apegarse a la siguiente nomenclatura:

```
INICIALES          = CM (INDICAN TIPO DE PROCESO)
NUMERO DE TRABAJO  = 4 DÍGITOS NUMÉRICOS, los cuales identifican a
                     la carretera.
TRAMO              = 2 DÍGITOS, los cuales identifican el tramo.
EXTENSIÓN          = .GRA -> DATOS GENERALES
                     .VF  -> SECCIONES DE TERRENO
                     .VER -> ALINEAMIENTO VERTICAL
```

De esta forma, los únicos datos que necesitamos para definir los archivos son el "NUMERO DE TRABAJO" (Identificador de la carretera) y el "TRAMO", que se quiere utilizar.


DIRECCION GENERAL DE CARRETERAS FEDERALES
PROGRAMA PARA LA CAPTURA DE DATOS DE CURVA MASA
OPCIONES DE CAPTURA
Definición del ramo
Datos enerales del tramo
ecciones de terreno
Alineamiento ertical
Gra icación de Secciones
Sa ir al sistema
Archivo en uso CM1111AA
CAMBIAR MODOS INSERT > MODO INSERCION/SUSTITUCION F9 > MODO SIN/CON CEROS
MODO DE INSERCION


DIRECCION GENERAL DE CARRETERAS FEDERALES
PROGRAMA PARA LA CAPTURA DE DATOS DE CURVA MASA
OPCIONES DE CAPTURA
Definición del ramo
Datos enerales del tramo
Sa ir al sistema
Archivo en uso CM1111AA
CAMBIAR MODOS INSERT > MODO INSERCION/SUSTITUCION F9 > MODO SIN/CON CEROS
MODO DE INSERCION

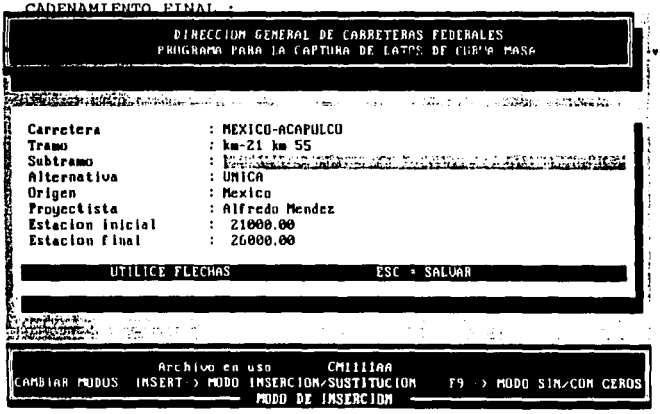
Para realizar la función de las primeras dos opciones del menú, que corresponden a la lectura del Numero de trabajo y del tramo, podemos auxiliarnos en las funciones de lectura que generamos en el capitulo correspondiente a las formas.

Una vez definidos estos dos parámetros, se definen con ellos los archivos que se van a utilizar; las siguientes tres opciones del menú, corresponden a dos opciones de captura.

CAPTURA DE DATOS GENERALES.

Para la captura de datos utilizaremos la FORMA DE ENTRADA Y SALIDA DE DATOS, la cual contendrá los siguientes campos

CARRETERA:
TRAMO:
SUBTRAMO:
ALTERNATIVA:
ORIGEN:
PROYECTISTA:
CADENAMIENTO INICIAL:
CADENAMIENTO FINAL :



CAPTURA DE SECCIONES DE TERRENO.

Para la captura de lo puntos que definirán la geometría de las secciones del terreno, se basan en distancias y desniveles, se tendrá un limite de 50 puntos para definir una sección y un limite de 500 secciones, generalmente, los puntos de una sección no llega al limite, por esta razón, los puntos cuya distancia al eje este definida como cero, no serán tomados en cuenta; al terminar de capturar todos los datos, se tendrá que generar un archivo con el siguiente formato, el cual se ha venido utilizando en la dependancia ya mencionada anteriormente:

240381	179000.00	2138.11	1.90	-20.000	0.10	-19.000	2.30	-16.300	2.30	-8.500	-1.80	5.000
240382	179000.00	2138.11	-1.10	9.000	-1.60	22.300	4.80	40.000	0.00	0.000	0.00	0.000
240381	179009.00	2135.90	3.40	-20.000	1.00	-17.600	3.30	-16.200	1.50	-4.400	0.50	4.000
240382	179009.00	2135.90	0.10	14.400	-2.30	25.400	-6.80	37.800	-5.50	40.000	0.00	0.000
240381	179020.00	2135.37	2.40	-20.000	1.70	-13.700	-0.10	-11.000	1.00	-9.700	0.00	8.400
240382	179020.00	2135.37	-1.90	15.000	-3.40	40.000	0.00	0.000	0.00	0.000	0.00	0.000
240381	179028.00	2134.38	2.20	-20.000	0.60	-5.400	0.40	-1.700	-1.30	7.100	-1.60	20.000
240382	179028.00	2134.38	-2.90	40.000	0.00	0.000	0.00	0.000	0.00	0.000	0.00	0.000
240381	179029.00	2133.49	5.00	-20.000	1.70	-6.000	-0.80	20.000	-2.10	40.000	0.00	0.000
240381	179031.50	2134.08	0.40	-20.000	1.50	-16.000	-0.30	1.700	-1.90	4.400	-1.30	7.500
240382	179031.50	2134.08	-1.60	20.000	-3.20	40.000	0.00	0.000	0.00	0.000	0.00	0.000
240381	179040.00	2133.13	1.70	-20.000	0.20	3.500	-1.60	16.000	-1.80	29.800	-3.20	40.000
240381	179060.00	2131.02	1.80	-20.000	0.40	-5.000	-1.00	3.300	-1.00	-1.100	-0.10	7.100
240382	179060.00	2131.02	-4.00	24.600	-5.30	40.000	0.00	0.000	0.00	0.000	0.00	0.000
240381	179063.00	2129.58	2.80	-20.000	1.10	-3.200	0.00	2.500	0.50	5.800	-2.70	22.200
240382	179063.00	2129.58	4.00	40.000	0.00	0.000	0.00	0.000	0.00	0.000	0.00	0.000

En donde la primera columna corresponde a la tarjeta, la cual esta constituida por el Numero de trabajo y un numero consecutivo, el cual comienza en 81, indicando que es la primera linea de puntos, 82 que es la segunda linea, y así sucesivamente hasta encontrara otra tarjeta con 81, indicando el inicio de otra sección de terreno; la siguiente dos columnas corresponde al cadenamiento y la elevación; las últimas 10 columnas contienen un total de 5 puntos del terreno, dados por desnivel, distancia; las distancias negativas indican lado izquierdo del eje, y las positivas lado derecho; las distancias de cero no son tomados en cuenta.

Para resolver este problema, utilizaremos el concepto de FORMAS DE ENTRADA Y SALIDA DE DATOS, auxiliandonos de una archivo binario para la lectura rápida de los datos; al finalizar de realizar la captura, se tendrá que generar el archivo correspondiente con su extensión respectiva (VF). Un aspecto fundamental que debemos de tomar en cuenta, es que el usuario, pueda observar gráficamente la distribución del terreno mediante los puntos que ha introducido; es decir desea obtener la graficación de los puntos; otro punto importante que debe de tomarse en cuenta es el borrado de secciones y desplegar algún listado de todos los cadenamientos o secciones que se han capturado, para que el usuario pueda determinar cuantas secciones ha capturado.

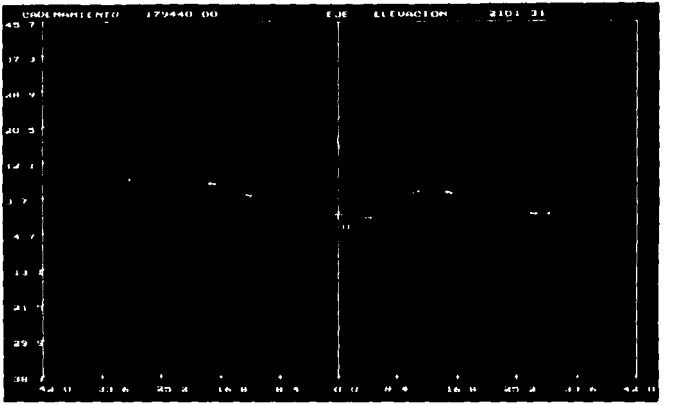
DIRECCION GENERAL DE CARRETERAS FEDERALES
PROGRAMA PARA LA CAPTURA DE DATOS DE CURVA MASA

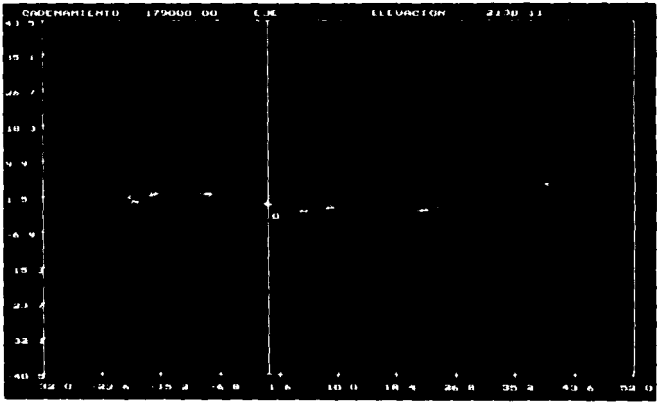
CADENAMIENTO 179000.00 ELEVACION 2138.11

DIST	DESN	DIST	DESN	DIST	DESN	DIST	DESN	DIST	DESN
-20.00	1.90	-19.00	0.10	-16.30	2.30	-0.50	2.30	5.00	-1.80
9.00	-1.10	22.30	-1.60	40.00	4.80	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

UTILICE FLECHAS ESC=SALVAR PG_DOWN=CAD ANTERIOR PG_UP=CAD SIGUIENTE
F4 >LISTA DE CADENAMIENTOS F5->ELIMINAR CAD F6 > OBSERVAR TERRENO

Archivo en uso CM1111AA
CAMBIAR MODOS INSERT-> MODO INSERCION/SUSTITUCION F9 -> MODO SIN/CON CEROS
MODO DE INSERCION


CADENAMIENTO 179440.00
EJE ELEVACION 2101.31


CADENAMIENTO 179000.00
EJE ELEVACION 2130.31

CAPTURA DE DATOS DE ALINEAMIENTO VERTICAL.

Para la captura de los datos de alineamiento vertical, debe de tomarse en cuenta un factor muy importante, el numero de datos que se pretende presentar en la pantalla mediante el uso de una FORMA, ya que por lo general, es mucho mayor que el tamaño de la pantalla; por esto, tenemos que tomar en cuenta el uso de desplazamiento (SCROLL) de una ventana, cuando estemos en los extremos de ella para poder representar una cantidad mayor de datos.

GRAFICACION DE SECCIÓN.

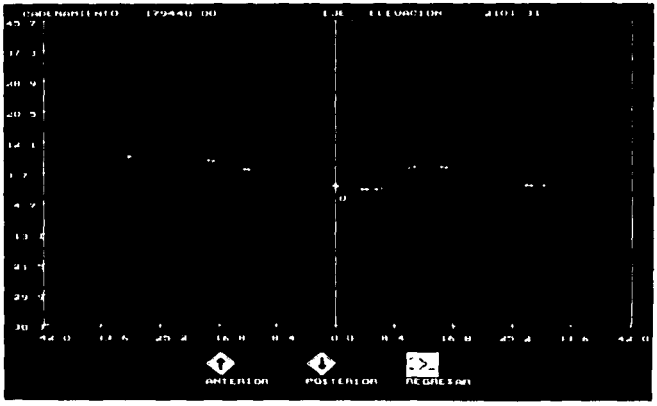
La última parte del menú, corresponde a la graficación de secciones; con esta opción, el usuario podrá observar gráficamente todas las secciones que ya capturo, con el fin de observar la geometría del terreno y poder definir de una mejor forma la geometría del proyecto; para poder optimizar la velocidad de respuesta del equipo, nos auxiliaremos de un archivo binario de acceso directo, con el fin de que la lectura sea mucho más rápida; para cambiarse de una sección a otra, podemos utilizar las teclas de FLECHA IZQUIERDA Y RE. PAG. para regresar a la sección anterior; y de FLECHA DERECHA Y AV. PAG: para avanzar a la siguiente sección; también, podemos aprovechar el uso de los iconos para facilitar la selección al usuario; en fin; este es solamente una idea de como podemos solucionar el problema, pero, como se ha mencionado en loa capítulos anteriores, cada persona puede realizar un diseño diferente de solución, utilizando diferentes estructuras, estilos de programación y elementos de interface; sin embargo, todos deben de encaminarse hacia un mismo fin, facilitar la labor al usuario de una manera sencilla, fácil, rápida y eficiente.

Esta, es una mínima parte de todo un problema complejo y largo, el cual se ha resuelto utilizando elementos de interface, mediante los cuales se facilita la utilización del sistema; para este problema en especifico falta incluir una serie de procesos fundamentales para definir solucionar totalmente el problema, como

pueden ser:

- Definir la geometría del proyecto
- Realizar algún traslado de ejes
- Graficar terreno con proyecto
- Graficacion hacia plotter
- etc.

Sin embargo, solamente utilizamos este problema para ilustrar el uso de los elementos de interface, esperemos que con él queden mas claros y puedan observarse mas fácilmente sus ventajas.



GLOSARIO

ALMACENAMIENTO :

Retención de instrucciones de programa, datos iniciales e información dentro de la computadora para que la información este disponible para efectos de procesamiento.

ARGUMENTO :

Valor aplicado a un procedimiento o subrutina, necesario para evaluar u obtener algún resultado.

BIOS :

Del inglés Basic Input/Output System (Sistema básico de entrada y salida). Conjunto de programas codificados en memoria de solo lectura (ROM) en las computadoras IBM y compatibles. Estos programas facilitan la transferencia de información e instrucciones de control entre la computadora y los periféricos.

BIT :

Unidad básica de información en un sistema de numeración binaria. Las computadoras trabajan con números binarios, los cuales están representados por "0" y "1", a los cuales se les denomina bit.

BUFFER :

Unidad de memoria destinada a la tarea de retener información de forma temporal, en especial cuando se requiere que el almacenamiento temporal compense las diferencias de velocidad entre los componentes de un sistema de cómputo.

BYTE :

Es el número de bit's que una computadora necesita para representar un solo carácter.

CAMPO :

Espacio para una parte específica de información en un registro de datos.

CARÁCTER :

Cualquier letra, número, signo de puntuación o símbolo que pueda generarse en la pantalla.

COMPILADOR :
Programa que sirve para traducir instrucciones de un lenguaje de alto nivel a un lenguaje de máquina que sea comprensible por el procesador.

CONMUTAR :
Pasar de un modo o estado a otro.

CURSOR :
Carácter parpadeante en la pantalla, el cual indica una posición en la misma. También puede definirse como un punto móvil de luz en la pantalla y que tiene la posibilidad de desplazamiento vertical y horizontal.

DATO :
Representa una unidad de información, es decir, son elementos básicos de la información que pueden ser procesados o producidos por una computadora; los datos solamente representan la información.

FORMA :
Es un Documento, el cual contienen un formato fijo y espacios en blanco para alojar datos; por ejemplo una forma del Seguro Social.

FUNCIÓN :
Procedimiento diseñado, el cual regresa un valor calculado cada vez que es llamada.

HARDWARE :
Se le llama así a los componentes electrónicos, tarjetas, periféricos y en general cualquier parte física del sistema de cómputo, incluyendo circuitos integrados, terminales de video, impresoras, y dispositivos auxiliares.

HIPERMEDIA :
Aplicación asistida por computadora capaz de agregar texto, gráficas, sonido, video y voz a las capacidades de un sistema

de información.

HIPERTEXTO :
Sistema de información no secuencial, el cual contiene ligas entre archivos, con lo cual se logra procesar o hacer referencia a otro tipo de información relacionada con la primera a través de las ligas.

ICONO :
Es una interface gráfica, el cual es un símbolo o figura en la pantalla que representa a un archivo, función, módulo o alguna otra entidad.

INTERRUPCIÓN :
Instrucción del microprocesador que detiene momentáneamente el procesamiento para que se lleve a efecto una aplicación de entrada, salida o de otro tipo; cuando termina la operación el proceso continua.

MICROPROCESADOR :
Circuito integrado que contiene la unidad Lógica-Aritmética (ALU) y la unidad de control de la unidad central de procesamiento (CPU) de la computadora.

PARÁMETRO :
Variable que se transfiere a un módulo, procedimiento o función para su posterior utilización con el fin de realizar algún proceso, esta variable puede ser modificada si así se requiere durante el proceso.

PERIFÉRICO (EQUIPO) :
Equipo que se conecta a la computadora, la cual se encarga de controlarlo, por ejemplo : impresoras, plotters, scanners, etc.

PIXEL :
Elemento gráfico mínimo que exhibe un dispositivo en la pantalla, y a partir del cual se constituye una imagen. A cada punto de la pantalla se le denomina pixel, y representa una unidad lógica de exhibición de video.

PROGRAMA :

Conjunto de instrucciones, procedimientos y funciones de computadora para realizar algún trabajo.

PROCESO :

Indica cualquier acción que se realiza con la información en el interior de la computadora. El procesamiento puede modificar el contenido o la forma de la información; o solo puede transferirla de una fuente a otra sin modificarla.

RATON (MOUSE) :

Dispositivo de entrada que cuenta con uno o mas botones de control; esta alojado en un estuche cuyo tamaño cabe en la mano y cuyo diseño permite deslizarlo sobre la mesa. A medida que se mueve el ratón, sus circuitos transmiten señales que mueven el apuntador en la pantalla.

REPORTE :

Representa el desplegado de datos e información.

RESALTAR :

Marcar caracteres o nombres de comandos en la pantalla en video inverso; el proceso sirve para que el usuario identifique el texto o comando sobre el que se desea se ejecute la siguiente operación.

RESOLUCIÓN :

Medida de nitidez de una imagen generada por un dispositivo de salida como un monitor o una impresora. En un monitor, la resolución se expresa como el número de pixeles que aparecen en la pantalla en posición horizontal y el de líneas que aparecen en posición vertical.

RUTINA :

Procedimiento o función realizadas en un lenguaje determinado, y que realizan alguna función especifica como pueden ser: recuperar datos o controlar un sistema; en cuanto a su función, las rutinas pueden ser especificas o generales.

SCROLL :

Método de observación de la información en la pantalla,

implica el movimiento continuo de información ya sea en forma vertical u horizontal.

SISTEMA DE INFORMACIÓN:

Es el conjunto total de procedimientos, operaciones y funciones dedicadas a la generación, recolección, almacenamiento, recuperación y difusión de datos e información.

SOFTWARE:

Programas y rutinas que indican a la computadora que hacer y cuando hacerlo; incluye compiladores ensambladores, traductores, interpretes y programas de aplicación.

USUARIO:

Se le denomina a aquella persona que utiliza y se beneficia directa o indirectamente de las capacidades de un sistema de computo y usa estas para realizar alguna tarea.

VARIABLE:

Es una área designada en memoria que guarda un valor o cadena asignada a dicha variable; es decir, representa un elemento, dato o área especifica en la memoria principal que puede asumir cualquier valor.

VIDEO INVERSO:

Es un medio utilizado para resaltar el texto en la pantalla, para que los caracteres normales aparezcan brillantes sobre la pantalla, o para que algunos caracteres resalten del resto de la información.

BIBLIOGRAFÍA

EBERTS E. RAY.
USER INTERFACE DESIGN.
EDITORIAL PRENTICE HALL.
PRIMERA EDICIÓN 1994.

FAISON EDMUND W.
GRAPHICAL USER INTERFACE WITH TURBO C++.
EDITORIAL SAMS.
PRIMERA EDICIÓN 1991.

HORTON K. WILLIAM.
DESIGNING & WRITING ON-LINE DOCUMENTATION.
HELP FILES TO HYPERTEXT.
EDITORIAL JOHN WILEY & SONS
PRIMERA EDICIÓN 1990.

HORTON WILLIAM
THE ICON BOOK :
VISUAL SYMBOLS FOR COMPUTER SISTEMS AND DOCUMENTATION
EDITORIAL JOHK WILEY & SONS INC.
PRIMERA EDICIÓN 1994.

JOHN G., BURCH GARY GRUDNITSKY.
DISEÑO DE SISTEMAS DE INFORMACION.
EDITORIAL MEGABYTE.
PRIMERA EDICIÓN 1992.
PAGINAS 204-219.

KAY C. DAVID & LEVINE R. JOHN.
GRAPHICS FILE FORMATS.
EDITORIAL NINDCREST/McGRAW-HILL.
SEGUNDA EDICIÓN 1995.

LAURER, BRENDA.
THE ART OF HUMAN-COMPUTER INTERFACE DESIGN.
EDITORIAL ADDISON WESLEY PUBLISHING COMPANY INC.
PRIMERA EDICIÓN 1990.

MICROSOFT INC.
MICROSOFT MOUSE: PROGRAMMER'S REFERENCE.
ED. MICROSOFT PRESS.
SEGUNDA EDICIÓN 1991.

NABAJYOTI BARKAKAT.
X WINDOW : SYSTEM PROGRAMMING.
EDITORIAL SAMS PUBLISHING.
SEGUNDA EDICIÓN 1994.

NEXTSTEP.
USER INTERFACE GUIDELINES.
NEXTSTEP DEVELOPER's LIBRARY.
EDITORIAL ADDISON-WESLEY PUBLISHING COMPANY.
PRIMERA EDICIÓN 1992.

NORTON, PETER.
INSIDE THE IBM PC.
EDITORIAL BRADY.
PRIMERA EDICIÓN 1986.

NORTON, PETER & WILTON RICHARD.
GUIA DEL PROGRAMADOR PARA EL IBM PC Y PS2.
EDITORIAL ANAYA MULTIMEDIA-MICROSOFT.
SEGUNDA EDICIÓN 1988.

PHOENIX TECHNOLOGIES LTD.
SYSTEM BIOS FOR IBM PC's, COMPATIBLES
AND EISA COMPUTERS.
EDITORIAL ADDISON-WESLEY PUBLISHING COMPANY INC.
SEGUNDA EDICIÓN 1991.

POWELL E. JAMES.
DESIGNING USER INTERFACE.
EDITORIAL MICROTREND BOOKS.
PRIMERA EDICIÓN 1990.

SHNEIDERMAN BEN.
DESIGNING USER INTERFACE
STATEGIES FOR EFFECTIVE HUMAN-COMPUTER INTERACTION.
EDITORIAL ADDISON WESLEY PUBLISHING COMPANY.
SEGUNDA EDICIÓN 1989.

SIME M. E. & COMMBS M. J.
DESIGNING FOR HUMAN-COMPUTE COMMUNICACTION.
EDITORIAL ACADEMIN PRESS.
PRIMERA EDICIÓN 1983.

SUN MICROSYSTEMS INC.
OPEN LOOK.
GRAPHICAL USER INTERFACE.
EDITORIAL ADDISON-WESLEY PUBLISHING COMPANY INC.
PRIMERA EDICIÓN 1989.

WEISKAMP KEITH & HEINKY LOREN.
POWER GRAPHICS USING TUTBO C++.
EDITORIAL JOHN WILEY & SONS.
PRIMERA EDICIÓN 1991.