UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

ESCUELA DE MATEMATICA

TRABAJO DE GRADUACION

"EDITOR DE TEXTO ORIENTADO A COBOL"

PRESENTADO POR:

ROSA CRISTINA LEON MEDINA

FREDYS ARTURO GARCIA PORTILLO

PREVIO A LA OBTENCION DEL TITULO DE:

LICENCIADO EN MATEMATICA

OCTUBRE/1993

SAN SALVADOR EL SALVADOR CENTRO AMERICA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

ESCUELA DE MATEMATICA

TRABAJO DE GRADUACION

"EDITOR DE TEXTO ORIENTADO A COBOL"

PRESENTADO POR:

ROSA CRISTINA LEON MEDINA

FREDYS ARTURO GARCIA PORTILLO

PREVIO A LA OBTENCION DEL TITULO DE:

LICENCIADO EN MATEMATICA

OCTUBRE/1993

SAN SALVADOR EL SALVADOR CENTRO AMERICA

UNIVERSIDAD DE EL SALVADOR

RECTOR: Dr. FABIO CASTILLO FIGUEROA.

SECRETARIO GENERAL: Lic. MIRNA A. PERLA DE AMAYA.

FACULTAD DE CIENCIAS NATURALES Y MATEMATICA

DECANO:Lic. MARINA ESTELA CONTRERAS DE TOBAR.

SECRETARIO: Lic. RODOLFO FERNANDO MENJIVAR.

ESCUELA DE MATEMATICA

DIRECTOR: Ing. JOSE FRANCISCO MARROQUIN.

TRABAJO DE GRADUACION

COORDINADOR: Ing. OSCAR HERNAN LEMUS GOMEZ.

ASESOR: Ing. WILLIAM ERNESTO MARROQUIN.

TRABAJO DE GRADUACION

PRESENTADO POR:

ROSA CRISTINA LEON MEDINA.

FREDYS ARTURO GARCIA PORTILLO.

PREVIO A LA OBTENCION DEL TITULO DE:

LICENCIADO EN MATEMATICA

AGRADECIMIENTOS

Quisiéramos dar las gracias a todas las personas que de una u otra forma contribuyeron con sus aportes a la elaboración de este proyecto, principalmente:

- Al Ing. William Ernesto Marroquín, quien orientó de la mejor manera el desarrollo del proyecto.

- Al Ing. Oscar Hernán Lemus Gómez, por sus valiosas observaciones y recomendaciones que contribuyeron a mejorar nuestro trabajo.

- Al Lic. Salvador Urías, Coordinador del Laboratorio de Cómputo de Ingeniería Eléctrica, de la Universidad Centro Americana "JOSE SIMEON CAÑAS" (UCA), por su aporte técnico en el uso del lenguaje C; y por facilitarnos, en muchas ocasiones, el uso de las computadoras.

- A las empresas en donde realizamos la investigacion, por habernos facilitado informacion valiosa para la elaboración del proyecto.

LOS AUTORES.

DEDICATORIA

Por que mi inteligencia viene de Dios y el fruto de mis esfuerzos es el resultado del sacrificio de los míos, dedico este trabajo:

- A Dios todopoderoso, por que con la luz de su espíritu ilumina mi camino.

- A la memoria de mi padre, por que mis triunfos son también los suyos, y por haberme enseñado a respetar a mis semejantes y a enfrentar la vida con responsabilidad.

- A mi madre, por su infinito amor, comprensión, su confianza en mí y el poder de sus oraciones, me fortalecieron en los momentos más difíciles.

- A mis hermanos, por creer en mí y pensar que con la ayuda de Dios todo es posible.

- A todos mis tíos y tías, principalmente a mi tío Fidel por su ayuda incondicional.

- A mis compañeros de trabajo, en la Facultad Multidisciplinaria de Oriente, de igual forma a mis alumnos, por que de ellos también se aprende.

- A mis profesores, amigos y compañeros de estudio, como un recuerdo.

Fredys Arturo García Portillo.

INDICE GENERAL

páginas

página

CAPITULO III

3. DISEÑO DEL EDITOR

página

página

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

CAPITULO 1

INTRODUCCION

1.1 MARCO TEORICO Y ANTECEDENTES.

Desde su inicio, la ingeniería de software se ha dado a la tarea de diseñar herramientas que den soporte al desarrollo de muchas aplicaciones en los diferentes lenguajes de programación, tal es así, que la evolución de los editores de texto ha sido notable, pues han dejado de ser editores de línea, como se conocían antes, y se han convertido en editores de pantalla, como los que se conocen actualmente, lo cual los convierte en herramientas importantes para la solución de tareas dentro del quehacer de la informática.

El diseño de estas herramientas en el país no ha sido el empeño de las empresas e instituciones que se dedican a la producción de software, aunque si bien es cierto algunas instituciones educativas, como la UCA, han diseñado editores de texto en el desarrollo de cursos de programación, con el propósito de que los alumnos pongan en práctica el uso de un lenguaje en particular.

Considerando la importancia que los editores de texto tienen y la necesidad de ir creando nuestros propios productos de programación, científicamente y con capacidad competitiva, se plantea en este proyecto, el diseño de un editor de texto orientado a COBOL, que de facilidades de edición, compilación y ejecución de programas en éste lenguaje. Es con estos

propósitos que se han realizado investigaciones sobre editores de texto y sobre estudios desarrollados en el área de diseño de software, en base a las cuales se puede decir que la Universidad de El Salvador (UES), hasta el momento no cuenta con ningún tipo de investigación relacionada con el diseño de editores de texto; sin embargo, la Universidad Centroamericana "José Simeón Cañas" (UCA), ha hecho investigaciones de este tipo, contando actualmente con una herramienta de software titulada "Editor de Pantalla para el Sistema Operativo UNIX", cuyo contenido es la presentación de un editor de texto el cual da facilidades de uso en el ambiente UNIX; dentro de otras investigaciones que se relacionan con este proyecto, la UCA también cuenta con la tesis titulada "Paquete Generador de Aplicaciones de COBOL", éste da facilidades en la generación de reportes, menús, mantenimiento de archivos, en ambiente COBOL.

En base a lo anterior se puede decir que el desarrollo de proyectos en el área de diseño de software han sido muy pocos, pero que sin embargo, ponen de manifiesto que en nuestro país se cuenta con los recursos necesarios para el diseño de herramientas de software orientadas a aplicaciones específicas.

En relación a los editores que utilizan las empresas para desarrollar sus aplicaciones no se ha encontrado uno que dé facilidades para editar texto en COBOL, a nivel de PC's,

pero sí, existe a nivel de multiusuario un editor llamado "Editor de WANG", que ofrece ciertas ventajas para editar programas no sólo en lenguaje COBOL, sino también en ENSAMBLADOR, BASIC, C, FORTRAN, RPG II y PL/I. Este editor es usado por INTERDATA (representantes de WANG en El Salvador) y por todas las instituciones a las que esta presta soporte técnico.

1.2 DESCRIPCION DEL PROYECTO.

En este proyecto se desarrolla una herramienta de software que facilite la edición de texto para lenguaje COBOL.

El editor presenta, además, facilidades para procesar texto en otros lenguajes, lo que lo convierte en un editor de propósito general con funciones específicas para lenguaje COBOL.

Para definir las funciones que tiene el editor se hizo una investigación de campo, la cual consistió en entrevistar a programadores en lenguaje COBOL en las diferentes empresas del área de San Salvador, que desarrollan sus aplicaciones en este lenguaje; y además se efectuó un estudio de los editores que existen actualmente en el mercado, con el propósito de conocer las ventajas y desventajas que presentan en la codificación en lenguaje COBOL.

Para la construcción del editor se consideraron como aspectos fundamentales: tipos de ayuda, el uso de menús,

facilidades para el movimiento de cursor, facilidad de uso, el tamaño (en KB) de texto que podrá ser editado, inserción de código, funciones de edición normal, control de periféricos, manejo de memoria dinámica y tipos de estructuras de datos; aspectos que por su naturaleza y grado de dificultad hicieron necesario el uso del lenguaje de programación C.

Para la creación del editor se utilizarán técnicas modernas de ingeniería de software (programación estructurada, técnicas de abstracción de datos, etc.) que permitieron un diseño descendente y modular del mismo.

1.3 OBJETIVOS.

1.3.1 GENERAL(ES).

- Construir un editor de texto que de facilidades para la codificación de programas en lenguaje COBOL.
- Proporcionar a la Universidad de El Salvador y a otras instituciones interesadas en el diseño de software, una metodología adecuada de la ingeniería de software que muestre el diseño de un editor.

1.3.2 ESPECIFICOS.

- Desarrollar la metodología para seleccionar el conjunto de funciones o comandos que tiene el editor, de tal manera que

facilite la edición de texto en COBOL.

- Proporcionar a la Universidad de El Salvador (y a otras instituciones educativas) una herramienta de software que de soporte práctico a los cursos de programación en COBOL.

- Identificar el conjunto de estructuras de datos más adecuado para el diseño de un editor.

- Facilitar el trabajo de codificación, compilación y ejecución, a los programadores en lenguaje COBOL.

- Desarrollar una herramienta de software flexible que permita la ampliación de sus funciones.

- Hacer un buen uso de los recursos de hardware disponibles para la implementación del editor.

- Demostrar capacidad y conocimiento para desarrollar un producto competitivo, que satisfaga necesidades reales dentro del ámbito de la informática en El Salvador.

1.4 IMPORTANCIA.

Los editores de texto son parte fundamental dentro del quehacer de la informática, por tanto, debido al uso del lenguaje COBOL en muchas aplicaciones, resulta importante construir un editor que sirva de soporte para el desarrollo de las mismas.

La importancia de un editor de texto como el que se desarrolla en este proyecto radica básicamente en que como producto final se tiene:

a) Una metodología adecuada para el diseño de editores

b) Un editor de texto que beneficiará a un amplio sector dentro del ambiente de la informática, como lo son los programadores en COBOL, estudiantes en cursos de programación en COBOL, empresas que desarrollan aplicaciones en COBOL a nivel de PC's, instituciones que imparten cursos de aprendizaje del lenguaje COBOL, etc., y a otros usuarios de la informática en general.

c) Una herramienta de software escrita en español, lo que la convierte en una herramienta fácil de aprender y no exige conocimiento previo de inglés técnico para ser utilizada.

1.5 JUSTIFICACION.

Debido al avance tecnológico y al crecimiento empresarial en nuestro país, se hace cada vez más necesario el uso de herramientas adecuadas que faciliten el trabajo y permitan obtener mejores resultados. Dentro de este conjunto de herramientas se encuentra en la parte de gestión de información el uso de la computadora, la cual se ha convertido en los últimos años en el soporte técnico-práctico de instituciones educativas, comerciales y de investigación; dentro de este marco surge la idea del proyecto "**Editor de texto orientado a COBOL**", con la visión de construir una herramienta de software que permita la ampliación del mismo en trabajos futuros.

Por otra parte se tiene en nuestro país, un gran número de empresas comerciales e instituciones educativas que desarrollan una variada cantidad de aplicaciones informáticas, en lenguaje COBOL, a pesar de que existen muchas otras herramientas de software. Además, de acuerdo con las investigaciones realizadas, no se encontró en el mercado nacional un editor para PC's que dé facilidades para procesar texto en COBOL.

1.6 DELIMITACIONES.

En este proyecto se construye una herramienta de software que presenta todas las funciones básicas de un editor normal, con énfasis en las funciones especificas para facilitar el trabajo de codificación de programas en lenguaje COBOL.

Para el desarrollo global del proyecto, se previeron las siguientes limitantes, las cuales en su mayoría fueron superadas a medida que se avanzaba en el desarrollo del mismo.

- No existe una bibliografía que desarrolle una metodología para el diseño de editores.
- Se determinó establecer una interfaz con al menos uno de los compiladores de lenguaje COBOL, (p.e., RM-COBOL, MS-COBOL).
- Que se diseñaría un editor para ser usado a nivel de PC's.
- Se determinó el uso del lenguaje de programación C, por la facilidad que ofrece de portabilidad y transportabilidad de programas hacia otros ambientes, (p.e., a multiusuarios).

- Que el conjunto de funciones del editor, se vería limitado por el tiempo (10 meses) con que se contaba para desarrollar el proyecto.
- El poco conocimiento del lenguaje C y otras herramientas de software, dejaría de ser limitante en el menor tiempo posible.
- Que el manejo de periféricos estaría sujeto a la bibliografía que se adquiriera.

1.7 PLANTEAMIENTO DEL PROBLEMA

En la mayoría de los casos las instituciones diseñadoras de productos de informática en el país, se limitan al diseño de aplicaciones haciendo uso de lenguajes de otros productos de programación importados, dejando de lado el impulso que puede dársele a la ingeniería de software con el diseño y construcción de herramientas para el desarrollo de tales aplicaciones. Es con estos propósitos que se desarrolla en este proyecto una herramienta que presente un ambiente que facilite la programación en COBOL.

1.8 HIPOTESIS PLANTEADA

El diseño de herramientas de software, como los editores de texto con una orientación particular, es posible diseñarlos en nuestro país, realizando estudios y evaluaciones de productos de software importados, y haciendo investigaciones

bibliográficas sobre trabajos desarrollados en el área de diseño de software.

1.9 METODOLOGIA

La metodología que se siguió en el desarrollo de este proyecto es:

a) Investigación bibliográfica en la Universidad de El Salvador (UES) y en la Universidad Centroamericana "José Simeón Cañas" (UCA), sobre trabajos desarrollados en la línea de diseño de software.

b) Estudio y evaluación de editores más conocidos y utilizados en el mercado nacional, principalmente los utilizados para editar texto en COBOL.

c) Encuestas a empresas e instituciones que programan en lenguaje COBOL.

d) Consultas a expertos en el tema.

e) Reuniones periódicas con el asesor y coordinador del proyecto para discutir los avances del mismo.

f) Utilizar el programa SHOW PARTNER para simular las pantallas (menús) que tiene el editor, las cuales fueron sometidas a observaciones en la primera evaluación del proyecto.

1.10 PLAN DE SOLUCION.

En el desarrollo de este proyecto se operativiza un plan de solución con base en la metodología sugerida para el

desarrollo de un producto de informática.

El plan de solución es el siguiente:

a) Recopilación de información sobre editores utilizados para editar texto en COBOL, en las empresas que desarrollan aplicaciones en este lenguaje.

b) Revisión de las ventajas y desventajas que dan los editores utilizados para editar texto en COBOL.

c) Selección del conjunto de funciones que forman el editor. Para esto se tomó en cuenta los resultados de la información recopilada, el estudio de editores más conocidos en el mercado, sugerencias de expertos, etcétera.

d) Definición y clasificación de las funciones que forman el editor, de acuerdo al tipo de operación que realizan y a las áreas globales que se consideran en el diseño del editor.

e) Asignación de secuencia de teclas a cada función. Para esto se tomaron criterios que conducen a procesos fáciles para activar cada función.

f) Estudio y evaluación de estructuras utilizadas en el proceso de diseño del editor.

g) Fase de diseño del editor. Se desarrolla en dos etapas: diseño lógico (o arquitectónico) y diseño físico (o procedimental).

En el diseño lógico se desarrollan diagramas de tipo HIPO (Hierarchy-Input-Process-Output) para los procedimientos de cada módulo del editor, descripción conceptual de estructuras

de datos, nombre y descripción de funciones especiales que garantizan el funcionamiento del editor.

En el diseño físico se escribe el código fuente de todas las especificaciones hechas para cada uno de los módulos en el diseño lógico, y se integran los módulos en un sólo código fuente.

h) Revisión. Se hace una revisión del código fuente, chequeando principalmente la estructura lógica y la definición de objetos, tanto globales como locales.

i) Prueba. Esta se realiza en dos etapas:

1) Pruebas de unidad y evaluación de resultados.

2) Prueba del editor como producto final.

j) Elaboración del manual del usuario. En este se describen los requerimientos de hardware, el proceso de instalación del software, y se instruye al usuario para el uso del editor.

k) Conclusiones y recomendaciones. Como producto del proceso desarrollado en el proyecto, se dan aportes en la línea de diseño de software.

CAPITULO II

DETERMINACION DE REQUERIMIENTOS PARA EL DISEÑO DEL EDITOR

INTRODUCCION.

Tomando en cuenta que la ingeniería de software es una disciplina que se dedica al desarrollo, instrumentación y mantenimiento de productos de programación, y que ésta ofrece formas metodológicas y técnicas para el diseño de este tipo de herramientas, se planteó, para el desarrollo de este proyecto, una metodología con base en el modelo del ciclo de vida del software, la cual fue guiando el desarrollo del mismo.

En este capítulo se desarrolla la fase de "Determinación de requerimientos", la cual conduce a la selección, definición y clasificación del conjunto de funciones que tiene el editor. Se presenta, además, un resumen de funciones y se asigna una secuencia de teclas a cada función, con la que será activada. Finalmente, se presenta una descripción del funcionamiento preliminar del editor.

2.1 DEFINICION DEL PROYECTO.

En este proyecto se diseña un editor de texto al que se le llama de aquí en adelante FCEDIT.

El FCEDIT da facilidades para la edición de texto en COBOL, permitiendo la compilación y ejecución de programas

dentro de su propio ambiente. Presenta además facilidades para editar código fuente escrito en otros lenguajes de programación diferentes de COBOL, lo que lo convierte en una herramienta de software de propósito general.

2.2 PROCESO PARA DETERMINAR EL CONJUNTO DE FUNCIONES QUE TIENE EL FCEDIT.

Para lograr mejores resultados y presentar la metodología que se siguió, se aborda este tema describiendo el proceso de investigación y presentación de resultados, los criterios de selección para obtener, finalmente, el conjunto de funciones que tiene el FCEDIT.

2.2.1 INVESTIGACION Y PRESENTACION DE RESULTADOS.

Por el tipo de información que se requiere para desarrollar un proyecto como éste, se decidió utilizar como instrumento el CUESTIONARIO (ver anexo #1), el cual fue pasado en empresas e instituciones del área de San Salvador, que desarrollan aplicaciones en lenguaje COBOL (ver en anexo #1 listado de empresas).

Producto de esta investigación se obtuvieron los siguientes resultados, los cuales pueden consultarse en el anexo #1, para mayor detalle.

a) Un listado de editores que utilizan estas empresas para editar texto en COBOL.

b) Un cuadro resumen por cada editor, el cual presenta la siguiente información: el nombre de la empresa donde se realizó la encuesta, el nombre de la persona entrevistada, años de experiencia, grado académico, dominio de conocimiento (en %) que tiene sobre el editor, las ventajas y desventajas que presenta el editor según el criterio de la persona entrevistada.

c) Exigencias de edición de texto en COBOL, satisfechas y no satisfechas por los editores que utilizan estas empresas.

d) Necesidades de codificación en lenguaje COBOL, que los editores utilizados no las satisfacen.

e) Tipo de aplicaciones que desarrollan las empresas donde se pasó la encuesta.

f) Tipo de software utilizado en las empresas donde se pasó la encuesta.

g) Sugerencias dadas por las personas encuestadas, en torno al diseño de un editor que dé facilidades para la codificación de programas en COBOL.

h) Editores de mayor uso en las empresas encuestadas.

i) Listado de empresas consultadas, las cuales no aportaron información útil al proyecto.

2.2.1.1 CONCLUSIONES

Conclusiones obtenidas de la investigación realizada, en empresas que desarrollan aplicaciones en lenguaje COBOL.

- *No hay una tendencia específica hacia un editor que satisfaga las exigencias de los programadores en COBOL, debido a que la frecuencia de uso de los editores es bien dispersa.*
- *Se obtuvo un listado de 15 editores los cuales son utilizados para programar en COBOL, entre estos sobresalen con mayor frecuencia de uso el Q, SideKick y el editor de WANG; los dos primeros son utilizados en ambiente PC's y el último en multiusuarios.*
- *El editor de WANG es el único que presenta funciones especificas para trabajar en COBOL.*
- *Fortalecer las funciones de propósito general del FCEDIT, por la utilidad que éstas ofrecen en el momento de edición de texto.*
- *Entre las exigencias demandadas por los programadores en COBOL, que los editores que utilizan no las satisfacen, tenemos:*

  a) Edición, compilación y ejecución dentro del ambiente del editor.

  b) Indentación automática.

  c) Manejo de más de un archivo.

  d) Manejo de ventanas.

  e) Limitantes en el tamaño de texto.

- *Los tipos de aplicaciones que desarrollan las diferentes empresas son variadas, y dependen de los propósitos de las mismas.*

- No se mantiene uniformidad en el uso de compiladores COBOL, en las diferentes empresas que desarrollan aplicaciones en este lenguaje, a excepción del compilador COBOL VS WANG en ambiente multiusuario.

- La información obtenida es bien limitada, lo que atribuimos a la poca disposición que prestaban las personas a ser entrevistadas.

- Otro de los aspectos que se observó en la investigación, es el bajo nivel académico que tienen los programadores en COBOL y sus pocos años de experiencia, lo que los hace ser menos exigentes.

- De acuerdo a los resultados obtenidos, se puede decir, que los programadores desconocen una herramienta eficiente que les facilite el desarrollo de sus aplicaciones en lenguaje COBOL.

- En las sugerencias dadas por los programadores, no se demanda ningún tipo de facilidades para las funciones de impresión, y lo útil que resultaría el uso del MOUSE.

2.2.2 CRITERIOS PARA LA SELECCION DE FUNCIONES.

Los criterios tomados para seleccionar el conjunto de funciones están basados en:

a) Las exigencias de edición de texto en COBOL.

b) Las necesidades de codificación en lenguaje COBOL.

c) El estudio y evaluación de los editores de mayor uso en las empresas donde se pasó la encuesta.

d) Sugerencias dadas por las personas encuestadas.

e) Las sugerencias dadas por el asesor del proyecto, dado su conocimiento en el tema, y el dominio que posee sobre el editor EDWIN propio para el desarrollo de aplicaciones en lenguaje LISP.

d) Las áreas globales que es posible manejar con el FCEDIT, como son: archivos, ventanas, bloques, funciones de uso especial, funciones propias para la edición de texto en COBOL y funciones de salida al sistema operativo, definidas a partir de los resultados de la investigación y del estudio de la estructura organizativa de otros editores.

2.2.3 CONJUNTO DE FUNCIONES (O COMANDOS).

El FCEDIT está formado por el conjunto de funciones que se presentan a continuación.

La implementación de los comandos que aparecen marcados con un asterisco (*), está sujeta a las limitaciones de tiempo que se tienen para el desarrollo del proyecto.

- *Leer archivo en la memoria*
- *Escribir archivo*
- *Insertar archivo en el texto editado*
- *Imprimir archivo*
- *Ignorar cambios en el archivo*
- *Dividir pantalla*
- *Cambiar ventana*

. *Borrar ventana*

. *Marcar bloque*

. *Copiar bloque*

. *Mover bloque*

. *Borrar bloque*

. *Restaurar bloque*

. *Tabulador*

. *Moverse una palabra a la izquierda*

. *Moverse una palabra a la derecha*

. *Moverse al inicio de línea*

. *Moverse al final de linea*

. *Moverse al inicio del archivo*

. *Moverse al final del archivo*

. *Moverse una página hacia arriba*

. *Moverse una página hacia abajo*

. *Centrar página*

. *Insertar línea*

. *Borrar línea*

. *Borrar hasta el final de línea*

. *Borrar hasta el inicio de línea*

. *Borrar una palabra a la izquierda*

. *Borrar una palabra a la derecha*

. *Activar modo de backups ON/OFF*

. *Repetir último comando*

. *Abortar ejecución de comando*

. *Reemplazar cadena de caracteres*

. *Búsqueda hacia adelante*

. *Búsqueda hacia atrás*

. *Convertir mayúsculas a minúsculas y viceversa(*)*

. *Activar tabla ASCII(*)*

. *Activar RATON (*)*

. *Activar/Desactivar uso de funciones COBOL*

. *Compilación*

. *Ejecución*

. *Palabras reservadas*

. *Plantilla numerada para COBOL*

. *Variables (*)*

. *Párrafos (*)*

. *Salir temporalmente al DOS (Os Shell)*

. *Salir al DOS*

2.3 DEFINICION Y CLASIFICACION DE FUNCIONES.

La definición y clasificación que hace del conjunto de funciones que tiene el FCEDIT, se da en base al tipo de operación que cada una de estas tiene como propósito, así por ejemplo, la función encargada de copiar una región en el texto actualmente en memoria, aparecerá dentro de las funciones para el manejo de bloque.

2.3.1 FUNCIONES PARA EL MANEJO DE ARCHIVO.

. **Leer archivo en la memoria:** Transporta un archivo desde cualquier unidad de disco hacia la memoria del ordenador.

. **Escribir archivo:** Almacena el texto editado, como un archivo permanente, en cualquier unidad de disco.

. **Insertar archivo:** Transporta un archivo desde cualquier unidad de disco hacia la memoria del ordenador, insertándolo en el texto editado, a partir de la posición del cursor.

. **Imprimir archivo:** Envía a impresión el texto editado. El texto debe residir en la memoria del ordenador

. **Ignorar cambios en el archivo:** Cancela las modificaciones hechas al texto editado, actualmente en memoria, presentándose en pantalla la versión original del archivo.

2.3.2 FUNCIONES PARA EL MANEJO DE VENTANA

. **Dividir pantalla:** Divide la pantalla en dos ventanas, para permitir al programador trabajar dos archivos a la vez.

. **Cambiar ventana:** Cambia la ventana de trabajo.

. **Borrar ventana:** Borra la ventana, consultando al programador

si desea salvar el archivo que se encuentra en esa parte del BUFFER de video.

2.3.3 FUNCIONES PARA EL MANEJO DE BLOQUE.

. **Marcar bloque:** Marca un rango de líneas dentro del texto editado, con el propósito de ejecutar cualquiera de las siguientes operaciones: copiar, mover, borrar, etcétera.

. **Copiar bloque:** Copia parte del texto editado en una localización especificada por la posición del cursor.
Se considera la implementación de este comando, de tal manera que se puedan hacer varias copias, simplemente indicando las localizaciones con el cursor.

. **Mover bloque:** Mueve parte del texto editado desde una localización a otra. La posición destino se indica por la posición cursor.

. **Borrar bloque:** Borra parte del texto editado.

.**Restaurar bloque:** Recupera parte del texto editado, que se haya borrado previamente con el comando borrar bloque.
Se considera la implementación de este comando, de tal manera que se puedan recuperar varios bloques de texto borrado.

2.3.4 FUNCIONES PARA EL MOVIMIENTO DE CURSOR.

Las operaciones

. **Moverse un caracter a la izquierda**

. **Moverse un caracter a la derecha**

. **Moverse una línea hacia arriba**

. **Moverse una línea hacia abajo**

se efectuarán con las teclas para el movimiento del cursor.

. **Tabulador:** Si el archivo que se procesa es un programa en COBOL y la función Activar/desactivar está activa, desplaza el cursor a las columnas 8, 12, y 72, en otro caso tiene un funcionamiento normal.

. **Moverse una palabra a la izquierda:** Desplaza el cursor una palabra a la izquierda de su posición actual.

. **Moverse una palabra a la derecha:** Desplaza el cursor una palabra a la derecha de su posición actual.

. **Moverse al inicio de línea:** Desplaza el cursor al inicio de la línea, en la que se encuentre ubicado en el momento de la ejecución del comando.

. **Moverse al final de línea:** Desplaza el cursor al final de la línea, en la que se encuentre ubicado en el momento de la

ejecución del comando.

. ***Moverse al inicio del archivo:*** *Presenta en pantalla la primera página del texto editado.*

. ***Moverse al final del archivo:*** *Presenta en pantalla la última página del texto editado.*

. ***Moverse una página hacia arriba:*** *Presenta en pantalla la página previa a la posición del cursor.*

. ***Moverse una página hacia abajo:*** *Presenta en pantalla la página siguiente a la posición del cursor.*

.***Centrar página:*** *Centra una página del texto editado, en torno a la posición del cursor.*

2.3.5 FUNCIONES PARA INSERTAR/BORRAR.

Las operaciones

. Borrar un caracter a la izquierda

. Borrar un caracter a la derecha

se efectúan con las teclas de RETROCESO y DELETE, respectivamente.

. ***Insertar línea:*** *Inserta una nueva línea en el archivo, y la habilita para continuar el trabajo de edición.*

. ***Borrar línea:*** *Borra una línea del archivo. El efecto de este comando se produce en la línea donde se encuentre ubicado el cursor, previo a su ejecución.*

. ***Borrar hasta el final de línea:*** *Borra el texto editado a la derecha de la posición del cursor, en la línea que se encuentre en el momento que se ejecute el comando.*

. ***Borrar hasta el inicio de línea:*** *Borra el texto editado a la izquierda de la posición del cursor, en la línea que se encuentre en el momento que se ejecute el comando.*

. ***Borrar una palabra a la izquierda:*** *Borra la palabra que se encuentre a la izquierda de la posición del cursor.*

. ***Borrar una palabra a la derecha:*** *Borra la palabra que se encuentre a la derecha de la posición del cursor.*

2.3.6 FUNCIONES ESPECIALES.

. ***Activar modo de backups ON/OFF:*** *Activa/Desactiva la operación de backups, para archivos que estan siendo procesados en memoria, y que serán salvados en un dispositivo*

de almacenamiento secundario.

. ***Repetir último comando:*** *Repite la ejecución del último comando que se haya activado.*

Abortar ejecución de comando: *Cancela la operación del comando en ejecución.*

. ***Reemplazar cadena de caracteres:*** *Sustituye una cadena de caracteres en el texto actualmente en memoria.*

. ***Búsqueda hacia adelante:*** *Busca cadenas de caracteres en el texto actualmente en memoria, hacia adelante de la posición del cursor.*

. ***Búsqueda hacia atrás:*** *Busca cadenas de caracteres en el texto actualmente en memoria, hacia atrás de la posición del cursor.*

. ***Convertir mayúsculas a minúsculas, o viceversa (*):*** *Convierte parte del texto editado escrito en minúscula, a mayúscula, o viceversa.*

. ***Activar tabla ASCII(*):*** *Activa la tabla de los códigos ASCII.*

. ***Activar RATON(*):*** *Activa el uso del RATON.*

2.3.7 FUNCIONES PARA LA EDICION DE TEXTO EN COBOL.

. ***Activar/Desactivar:*** *Activa/desactiva el uso de las funciones para la edición de texto en COBOL.*

. ***Compilación:*** *Establece la interfaz con el compilador COBOL, para compilar el código fuente que se almacena en disco.*

. ***Ejecución:*** *Ejecuta archivo libre de errores, compilado en COBOL.*

. ***Palabras reservadas:*** *Proporciona un listado de palabras reservadas en COBOL, las cuales pueden ser trasladadas al texto que se está editando.*

.***Plantilla numerada para COBOL:*** *Presenta una linea numerada en la parte superior de la pantalla para facilitar la indentación de código fuente.*

. ***Variables(*):*** *Proporciona un listado de nombres de variable (en orden alfabético), definidas para el manejo de datos en el programa fuente. Presenta además el tipo y la longitud de cada variable.*

. ***Párrafos(*):*** *Proporciona un listado de nombres de párrafo, definidos en el programa fuente.*

2.3.8 FUNCIONES PARA SALIR AL DOS.

. ***Salir temporalmente al DOS:*** *Interrumpe la ejecución del editor y devuelve el control al sistema operativo, temporalmente.*

. ***Salir al DOS:*** *Cancela la ejecución del editor y devuelve el control al sistema operativo.*

2.4 RESUMEN DE FUNCIONES Y ASIGNACION DE SECUENCIA DE TECLAS A CADA FUNCION.

La forma de operar el FCEDIT se describe detalladamente en el siguiente tema, basta decir que el asociar una secuencia de teclas propia a cada función para tener una forma de accesarlas (se accesarán también a través de menús), se hizo en base a un estilo propio, tratando de involucrar una letra del nombre mnemónico de la función en la secuencia respectiva, con la cual también es posible llegar hasta la función una vez se haya ubicado con el cursor en el menú que la contiene.

COMANDOS DE ARCHIVO	*TECLAS*
LEER	*<Alt> + <L>*
ESCRIBIR	*<Alt> + <E>*
INSERTAR	*<Alt> + <I>*

	TECLAS
IMPRIMIR	<Alt> + <P>
IGNORAR_C	<Alt> + <G>
COMANDO DE VENTANA	
DIVIDIR	<Alt> + <D>
CAMBIAR	<Alt> + <C>
BORRAR	<Alt> + <B>
COMANDOS DE BLOQUE	
MARCAR	<Alt> + <M>
COPIAR	<Alt> + <O>
MOVER	<Alt> + <V>
BORRAR	<Alt> + <R>
RESTAURAR	<Alt> + <S>
COMANDOS PARA EL MOVIMIENTO DE CURSOR	
TABULADOR	<TAB>
PALABRA A LA IZQUIERDA	<Alt> + <←>
PALABRA A LA DERECHA	<Alt> + <→>
INICIO DE LINEA	<HOME>
FINAL DE LINEA	<END>
INICIO DE ARCHIVO	<Ctrl>+<PgUp>
FINAL DE ARCHIVO	<Ctrl>+<PgDn>
PAGINA HACIA ARRIBA	<PgUp>
PAGINA HACIA ABAJO	<PgDn>
CENTRAR PAGINA	<Alt> + <T>

COMANDOS PARA INSERTAR/BORRAR	TECLAS
INSERTAR LINEA	<Alt> + <N>
BORRAR LINEA	<Alt> + <Y>
BORRAR HASTA EL FINAL DE LINEA	<Ctrl> + <END>
BORRAR HASTA EL INICIO DE LINEA	<Ctrl> + <HOME>
BORRAR UNA PALABRA A LA IZQUIERDA	<Alt> +<BACKSPACE>
BORRAR UNA PALABRA A LA DERECHA	<Alt> + <DELETE>

COMANDOS ESPECIALES	
BACKUPS ON/OFF	<Alt> + <F1>
REPETIR COMANDO	<Alt> + <F2>
ABORTAR COMANDO	<Alt> + <F3>
REEMPLAZAR CADENA	<Alt> + <F4>
BUSQUEDA ADELANTE	<Alt> + <F5>
BUSQUEDA ATRAS	<Alt> + <F6>
MAYUSCULA/MIN	<Alt> + <F7>
ASCII	<Alt> +<F8>
RATON	<Alt> + <F9>

COMANDOS PARA LA EDICION DE TEXTO EN COBOL	
ACTIVAR/DESACTIVAR	<F2>
COMPILACION	<F3>
EJECUCION	<F4>
RESERVADAS_P	<F5>
PLANTILLA	<F6>

	TECLAS
VARIABLES	<F7>
PARRAFOS	<F11>

COMANDOS DE SALIDA

SALIR AL DOS	<F8>
SALIR	<F9>

OTRAS FUNCIONES

<F1>: Muestra una plantilla del editor para ayuda del usuario.

<F10>: Accesa el MENU principal.

<ESC>: Tecla de escape.

2.5 FUNCIONAMIENTO PRELIMINAR DEL EDITOR.

Con el objeto de proporcionar, una visión general de lo que es el funcionamiento del FCEDIT, se describen a continuación los detalles más importantes.

Primeramente se hace ver que el uso de menús es una de las ventajas que presenta el FCEDIT.

Las áreas globales que se han considerado son: funciones para el manejo de archivo, funciones para el manejo de ventana, funciones para gestión de bloque, funciones para insertar/borrar, funciones para movimiento de cursor, funciones especiales, funciones para la edición de texto en COBOL y funciones para salir al DOS, las cuales ya han sido

definidas.

Producto del estudio realizado de los editores que más se utilizan para editar texto en COBOL y la revisión de la estructura organizativa de otros editores, se consideró a bien que el menú principal del FCEDIT lo formaran las áreas de manejo de archivo, ventana, bloque, funciones especiales, funciones para la edición de texto en COBOL y funciones para salir al DOS, las cuales aparecen con su respectivo nombre mnemónico, esto es:

funciones para el manejo de archivo = ARCHIVO
funciones para el manejo de ventana = VENTANA
funciones para el manejo de bloque = BLOQUE
funciones especiales = ESPECIAL
funciones para la edición de texto en COBOL = COBOL
funciones para salir al DOS = SALIR

las que llamaremos de ahora en adelante "opciones del menú principal".

Para seleccionar una de las opciones del menú principal puede hacerse de dos formas: desplazando el cursor hasta la opción deseada y presionar <ENTER> o presionar la primera letra del nombre mnemónico de la opción deseada, esto conduce a un submenú que contiene las funciones que le corresponden a esa opción del menú principal. Las opciones dentro de un submenú pueden seleccionarse desplazando el cursor o presionando la letra en mayúscula de su nombre mnemónico,

luego de que una de estas opciones es seleccionada aparece en la última línea de la pantalla una breve explicación del propósito de la función y la secuencia de teclas que se deben presionar para poder ejecutarla desde la ventana del editor.

La ejecución de cualquier función puede hacerse a través del menú principal o presionando la secuencia de teclas respectiva desde la ventana del editor.

Las funciones que corresponden a las áreas ***insertar/borrar*** *y movimiento de* ***cursor,*** *sólo es posible ejecutarlas desde la ventana del editor, presionando la secuencia de teclas respectiva.*

Para ver la plantilla de ayuda del FCEDIT, se debe presionar <F1>, la cual muestra las funciones que corresponden a cada área y la secuencia de teclas que se debe presionar para ejecutar una determinada función.

Para accesar o salir del menú principal del FCEDIT, se debe presionar <F10>.

CAPITULO III

DISEÑO DEL EDITOR

INTRODUCCION.

El diseño del FCEDIT (producto de software en desarrollo) se divide en dos etapas: el diseño lógico y el diseño físico.

En el diseño lógico se utilizan los diagramas de tipo HIPO (Hierarchy-Input-Process-Output) como esquemas de representación que describen las entradas, los pasos de los procesos y las salidas que generan cada procedimiento o función del FCEDIT.

En el diseño físico se utiliza el lenguaje de programación C, para escribir el código fuente (o programas) para los procesos que se definieron en el diseño lógico.

Esta fase se desarrolla utilizando los conceptos fundamentales de diseño, los criterios de modularización, las notaciones para el diseño de la programación, propuestos por la ingeniería de software para el desarrollo de productos de computación.

3.1 Diseño Lógico.

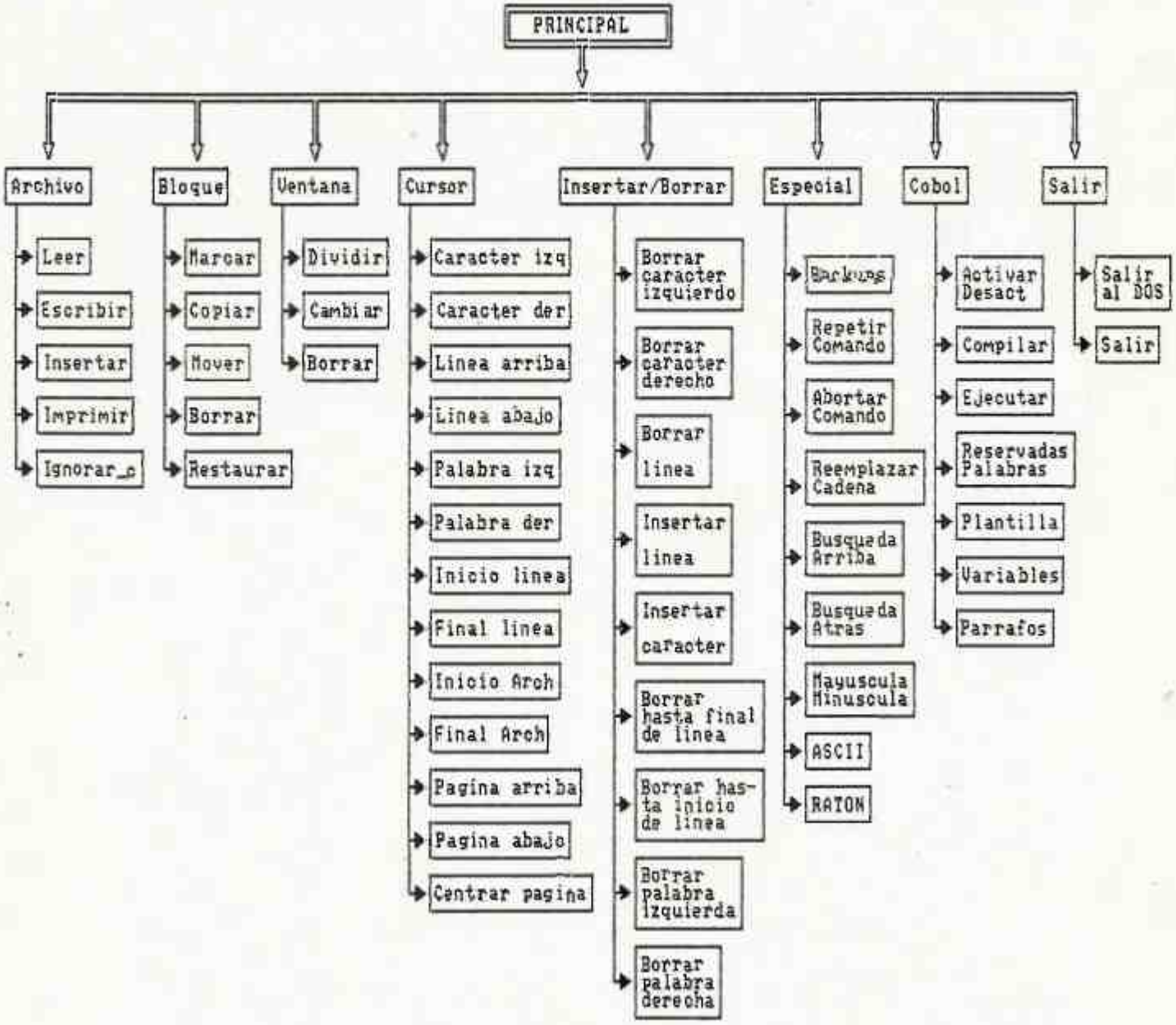
El FCEDIT, de acuerdo a los requerimientos definidos, es una herramienta de software integrada por un conjunto de comandos, clasificados de acuerdo a su función en ocho áreas, como son: archivo, bloque, ventana, movimiento de cursor, insertar/borrar, especiales, para edición de texto en COBOL y

de salida, los cuales aparecen en el menú principal del FCEDIT, a excepción de comandos para movimientos de cursor e insertar/borrar.

Se identifica cada una de estas áreas con un nombre mnemónico, anteponiendo el calificativo de módulo. esto es: módulo archivo, módulo bloque, módulo ventana, módulo insertar/borrar, módulo cursor, módulo especial, módulo cobol, y módulo salir.

Cada uno de estos módulos está formado por un conjunto de funciones, tal como se clasificaron y definieron en el capítulo dos, las que se denotan con sus respectivos nombres mnemónicos. El diagrama estructurado del FCEDIT se presenta en la ***figura-3.1****, el cual muestra en el nivel cero el módulo ejecutivo, en el nivel uno los diferentes módulos, en el nivel dos las funciones o procedimientos de cada módulo.*

Figura-3.1 Diagrama de módulos y funciones del FCEDIT.



En esta fase del diseño se utiliza una forma de documentación para cada uno de los módulos según los aspectos que se muestran en la ***figura-3.2.***

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO _____ FECHA _____ DISEÑO _____ MODULO No. _____ NOMBRE DEL MODULO _____	
PROCEDIMIENTOS INCLUIDOS	
1. 2. 3. 4. 5. . . .	

FIGURA-3.2 DIAGRAMA DE DOCUMENTACION DE MODULOS.

El diseño detallado de las diferentes funciones o procedimientos, se hace utilizando los diagramas jerárquicos del ciclo entrada-procesamiento-salida, también llamados diagramas HIPO (Hierarchy-Input-Process-Output), tal como se muestra en la ***figura-3.3.*** *En un diagrama HIPO se combina la descripción detallada de la lógica de un procedimiento mediante pseudocódigo con una indicación de qué entradas o*

variables necesita el procedimiento y de qué salidas o variables produce o modifica. La descripción detallada mediante la escritura de pseudocódigo estructurado, se hace utilizando una sintaxis similar a la del lenguaje C, dado que para el diseño físico de esta herramienta de software se plantea el uso de este lenguaje.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.		ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO _____		FECHA _____
DISEÑO _____		
PROCEDIMIENTO No. _____	NOMBRE DEL PROCEDIMIENTO _____	
ENTRADA	PROCESO	SALIDA

OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
PARAMETRO			
VARIABLE GLOBAL			
VARIABLE LOCAL			
TIPOS DE DATO GLOBAL			
TIPOS DE DATO LOCAL			
LIMITACIONES			

FIGURA-3.3 FORMATO DE DOCUMENTACION DE PROCEDIMIENTOS.

3.1.1 Identificación de módulos.

La identificación de cada módulo está formada por un nombre de código de cinco caracteres, a excepción de los módulos INSERTAR/BORRAR y COBOL, cuyos nombres de código es de seis caracteres.

Los dos primeros caracteres son las primeras letras del nombre del editor (FCEDIT), el tercer caracteres la primera letra del nombre mnemónico del módulo, y los últimos caracteres es el número correlativo que le corresponde a cada módulo de acuerdo a la estructura del FCEDIT.

Módulo	*Código*
ARCHIVO	*FCA00*
BLOQUE	*FCB00*
VENTANA	*FCV00*
CURSOR	*FCC00*
INSERTAR/BORRAR	*FCIB00*
ESPECIAL	*FCE00*
COBOL	*FCC000*
SALIR	*FCS00*

3.1.2 Identificación de procedimientos.

La identificación de cada procedimientos está formada por un nombre de código de cinco caracteres, a excepción de los módulos INSERTAR/BORRAR y COBOL, cuyos nombres de código es de seis caracteres.

Los dos primeros caracteres son las primeras letras del nombre del editor (FCEDIT), el tercer caracteres la primera letra del nombre mnemónico del módulo, y los últimos caracteres es el número correlativo que le corresponde a cada procedimiento en su módulo respectivo.

Procedimiento	*Código*
MODULO ARCHIVO	
Leer Archivo	*FCA01*
Escribir Archivo	*FCA02*
Insertar Archivo	*FCA03*
Imprimir Archivo	*FCA04*
Ignorar cambios en el Archivo	*FCA05*
MODULO BLOQUE	
Marcar bloque de texto	*FCB01*
Copiar bloque de texto	*FCB02*
Mover bloque de texto	*FCB03*
Borrar bloque de texto	*FCB04*
Restaurar bloque de texto	*FCB05*
MODULO VENTANA	
Dividir pantalla en dos ventanas	*FCV01*
Cambiar ventana de trabajo	*FCV02*
Borrar ventana inactiva	*FCV03*
MODULO CURSOR	
Tabulador	*FCC01*
Carácter a la izquierda	*FCC02*

Carácter a la derecha	*FCC03*
Línea arriba	*FCC04*
Línea abajo	*FCC05*
Palabra izquierda	*FCC06*
Palabra derecha	*FCC07*
Inicio de línea	*FCC08*
Final de línea	*FCC09*
Inicio de archivo	*FCC10*
Final de archivo	*FCC11*
Página arriba	*FCC12*
Página abajo	*FCC13*
Centrar página	*FCC14*
MODULO INSERTAR/BORRAR	
Borrar carácter izquierdo	*FCIB01*
Borrar carácter derecho	*FCIB02*
Borrar línea	*FCIB03*
Borrar hasta final de línea	*FCIB04*
Borrar hasta inicio de línea	*FCIB05*
Borrar palabra izquierda	*FCIB06*
Borrar palabra derecha	*FCIB07*
Insertar línea	*FCIB08*
Insertar carácter	*FCIB09*
MODULO ESPECIAL	
Backups	*FCE01*
Repetir comando	*FCE02*

Abortar comando	*FCE03*
Reemplazar cadena	*FCE04*
Búsqueda adelante	*FCE05*
Búsqueda atrás	*FCE06*
Convertir de mayúscula/minúscula y viceversa	*FCE07*
Activar tabla de códigos ASCII	*FCE08*
Activar/desactivar ratón	*FCE09*
MODULO COBOL	
Activar/desactivar funciones	*FCC001*
Compilar programa	*FCC002*
Ejecutar programa	*FCC003*
Mostrar listado de palabras reservadas	*FCC004*
Mostrar plantilla	*FCC005*
Mostrar Variables	*FCC006*
Mostrar Párrafos	*FCC007*
MODULO SALIR	
Salir temporalmente al DOS	*FCS01*
Salir al DOS (fin de sesión)	*FCS02*

3.1.3 Requerimientos informáticos.

3.1.3.1 Identificación de pantallas.

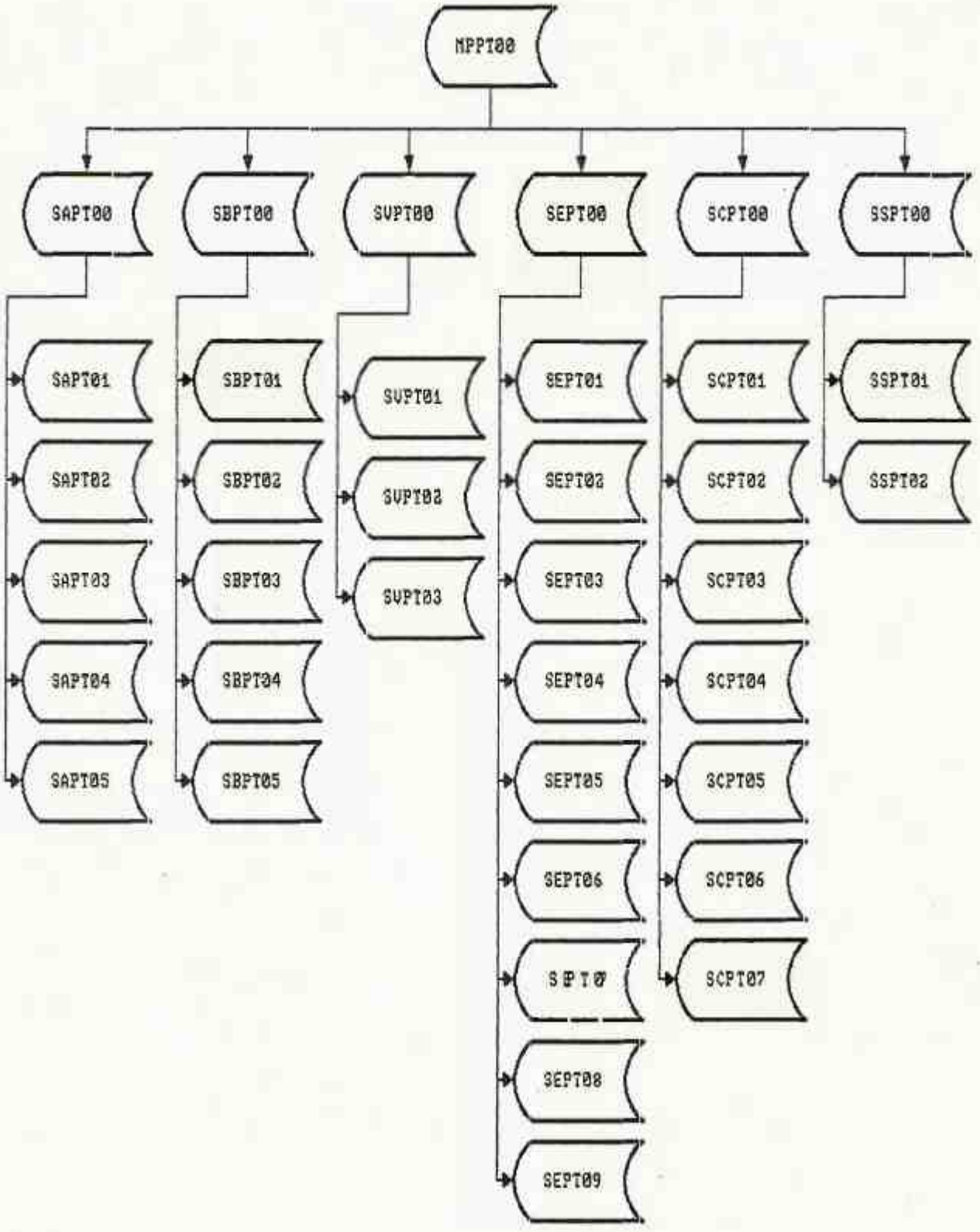
La identificación de cada pantalla está formada por un nombre de código que etiqueta exclusivamente cada pantalla por módulo, siguiendo la estructura funcional y procedimental del FCEDIT.

Pantalla	Código
GENERAL(ES)	
Menú principal de FCEDIT	MPPT00
SUBMENU ARCHIVO	
Submenú de funciones de archivo	SAPT00
Leer Archivo	SAPT01
Escribir Archivo	SAPT02
Insertar Archivo	SAPT03
Imprimir Archivo	SAPT04
Ignorar cambios en el Archivo	SAPT05
SUBMENU BLOQUE	
Submenú de funciones de bloque	SBPT00
Marcar bloque de texto	SBPT01
Copiar bloque de texto	SBPT02
Mover bloque de texto	SBPT03
Borrar bloque de texto	SBPT04
Restaurar bloque de texto	SBPT05
SUBMENU VENTANA	
Submenú de funciones de ventana	SVPT00
Dividir pantalla en dos ventanas	SVPT01
Cambiar ventana de trabajo	SVPT02
Borrar ventana inactiva	SVPT03
SUBMENU ESPECIAL	
Submenú de funciones especiales	SEPT00
Backups	SEPT01

Repetir comando	*SEPT02*
Abortar comando	*SEPT03*
Reemplazar cadena	*SEPT04*
Búsqueda adelante	*SEPT05*
Búsqueda atrás	*SEPT06*
Convertir de mayúscula/minúscula y viceversa	*SEPT07*
Activar tabla de códigos ASCII	*SEPT08*
Activar/desactivar ratón	*SEPT09*
SUBMENU COBOL	
Submenú de funciones cobol	*SCPT00*
Activar/desactivar funciones	*SCPT01*
Compilar programa	*SCPT02*
Ejecutar programa	*SCPT03*
Mostrar listado de palabras reservadas	*SCPT04*
Mostrar plantilla	*SCPT05*
Mostrar Variables	*SCPT06*
Mostrar Párrafos	*SCPT07*
SUBMENU SALIR	
Submenú de funciones de salida al DOS	*SSPT00*
Salir temporalmente al DOS	*SSPT01*
Salir al DOS (fin de sesión)	*SSPT02*
OTRAS PANTALLAS	
- Mensaje al ejecutar la función leer "Documento a recuperar"	*MEPT00*

*La **figura-3.4** muestra el diagrama jerárquico de pantallas de los módulos y funciones del menú principal del FCEDIT. Por las características propias de los módulos FCC00 (módulo cursor) y FCIB00 (módulo insertar/borrar) y la naturaleza de las funciones que los forman no se hace necesario el diseño de pantallas para tales casos.*

Figura-3.4 Diagrama jerárquico de pantallas.

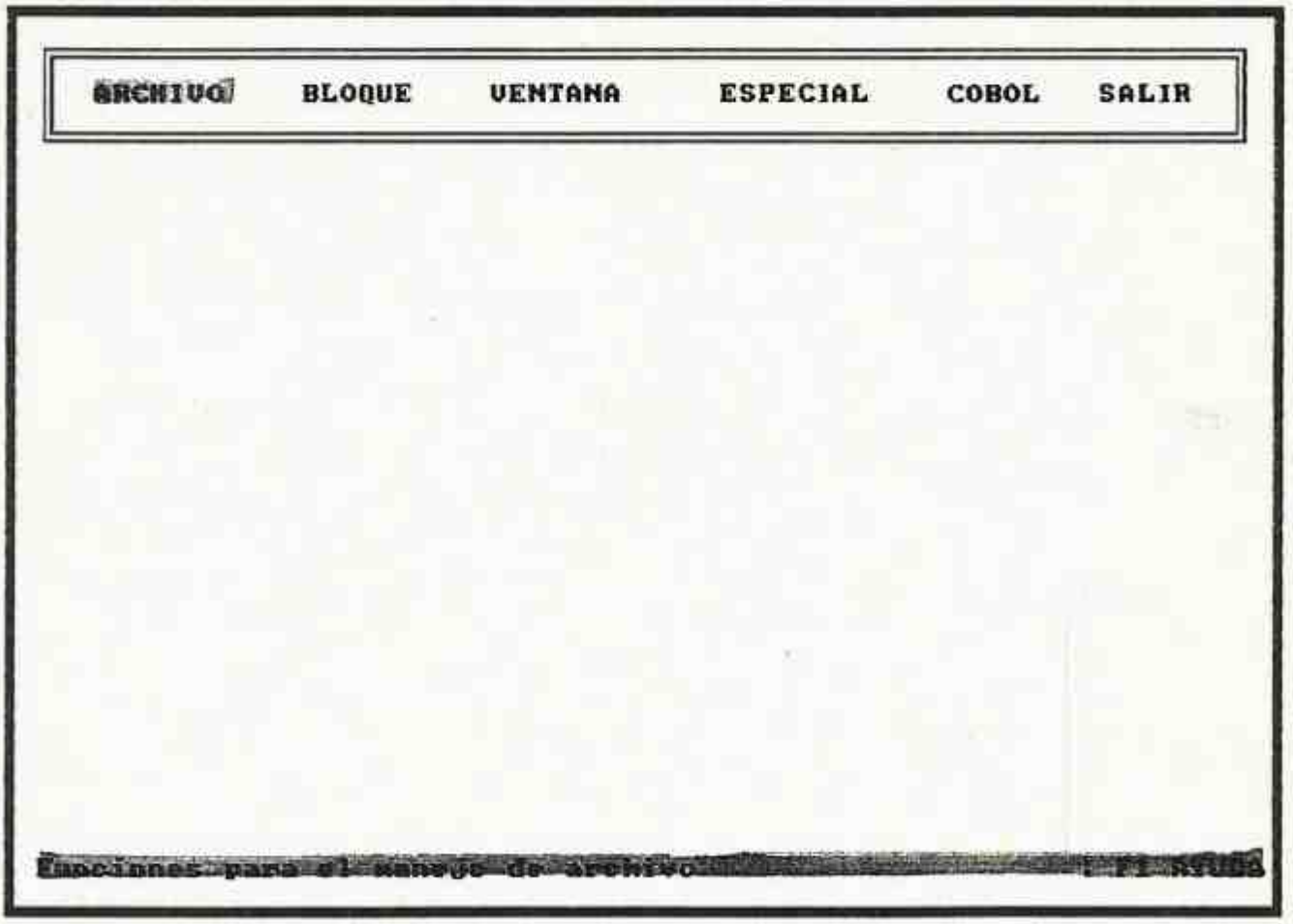


3.1.3.2 DISEÑO DE PANTALLAS.

El diseño de pantallas sigue una lógica, funcionamiento y presentación similar, razones por las cuales en los diseños que se presentan a continuación no aparece la totalidad de estas pantallas; sin embargo, se hace énfasis en las de COBOL por ser ésta el área en la que el FCEDIT da su mayor aporte.

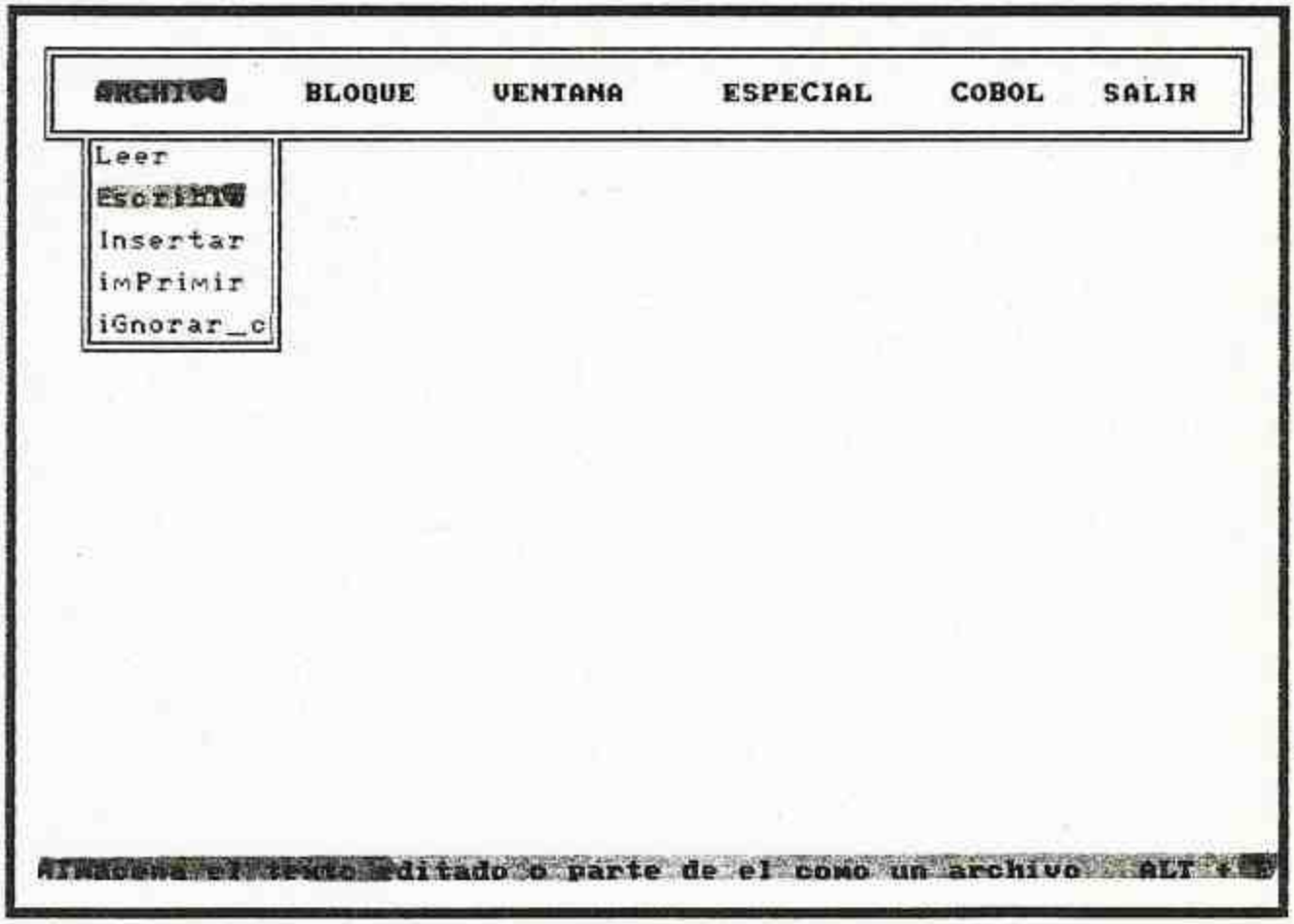
3.1.3.2.1 Menú principal del FCEDIT.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: MPPT00	NOMBRE: MENU PRINCIPAL	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



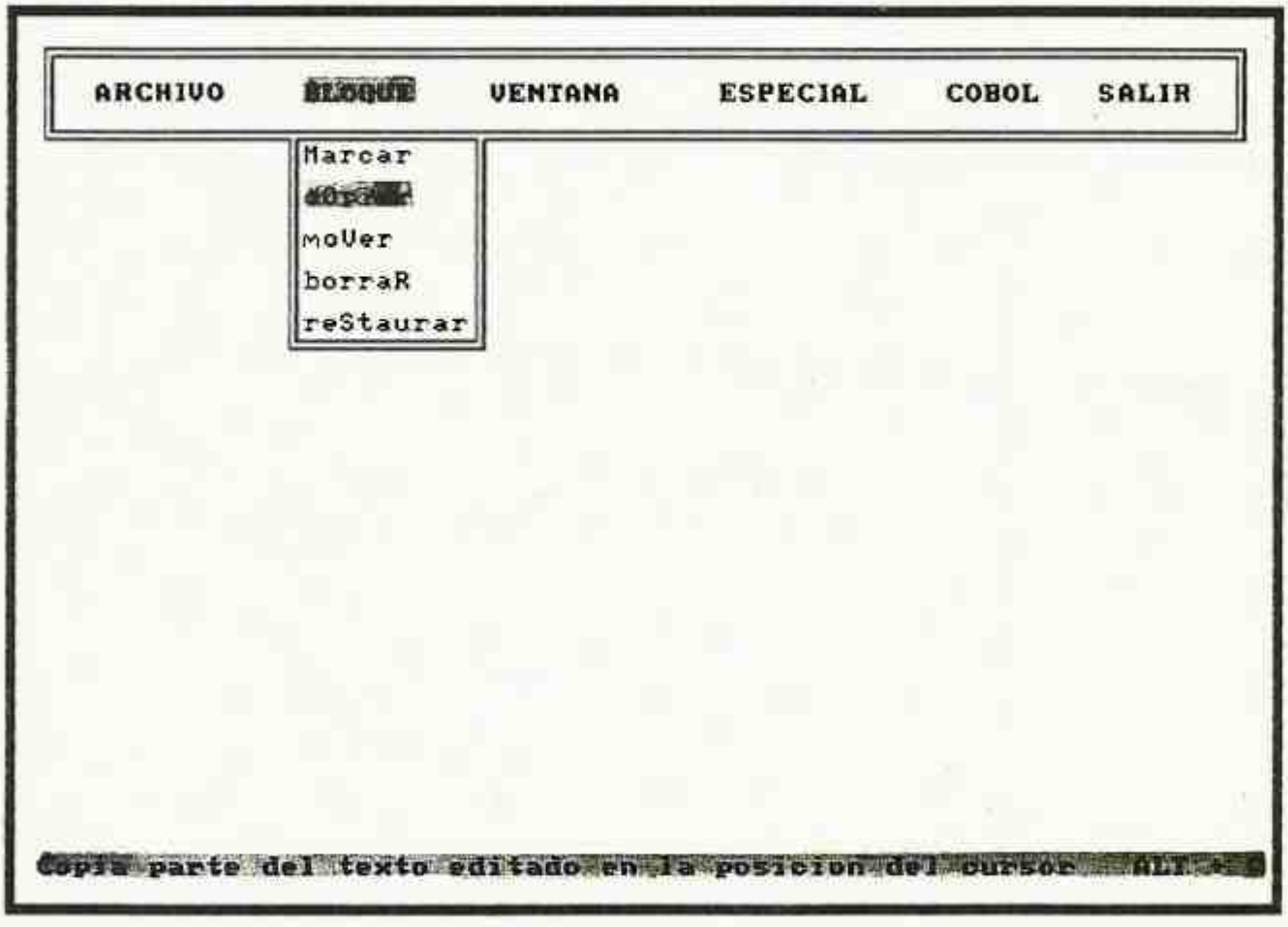
3.1.3.2.2 Escribir archivo.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SAPI82	NOMBRE: ESCRIBIR ARCHIVO	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



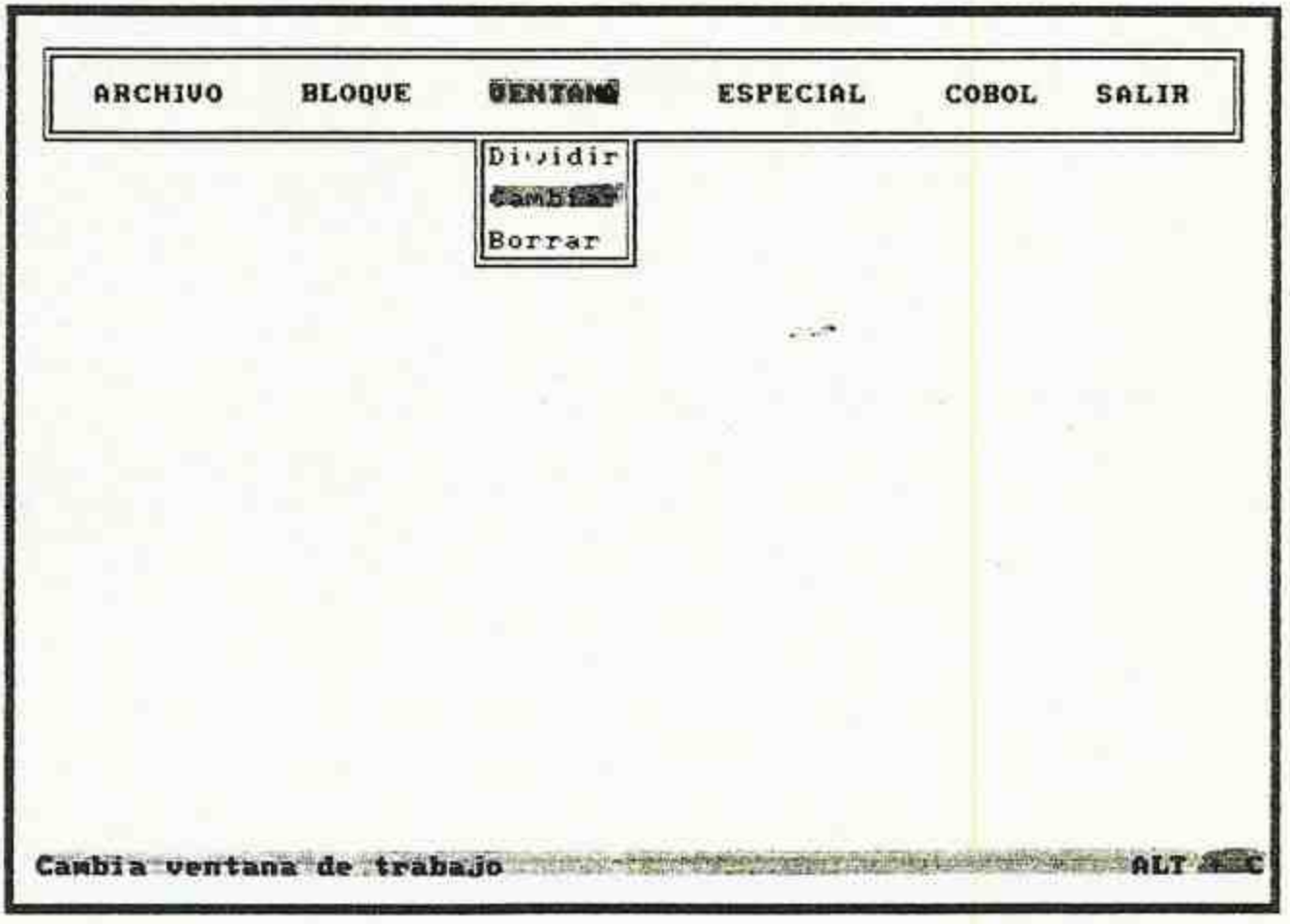
3.1.3.2.3 Copiar bloque de texto.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SBPT02	NOMBRE: COPIAR BLOQUE DE TEXTO	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



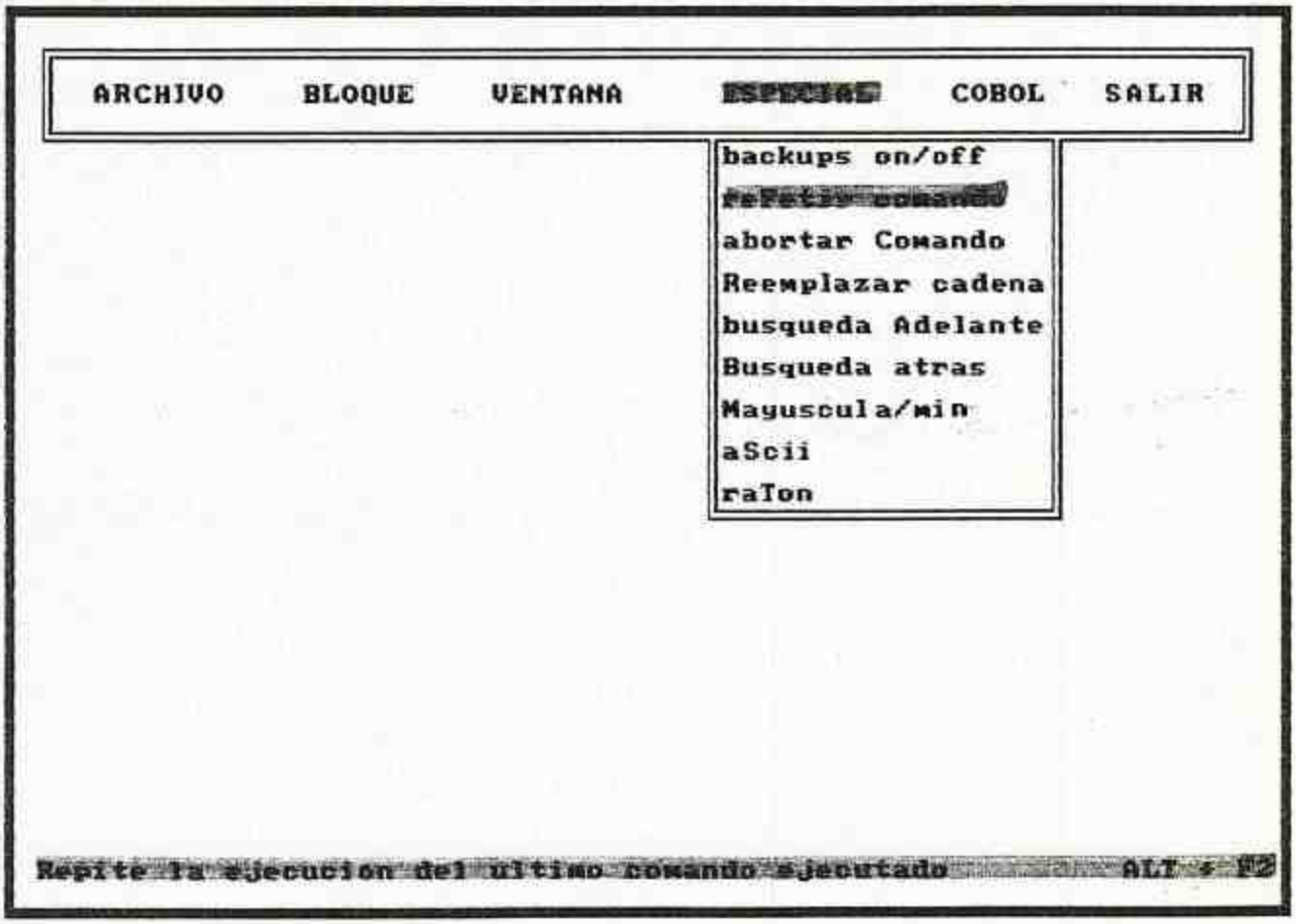
3.1.3.2.4 Cambiar ventana de trabajo.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **SVPT02**	NOMBRE: **CAMBIAR VENTANA DE TRABAJO**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



3.1.3.2.5 Repetir comando.

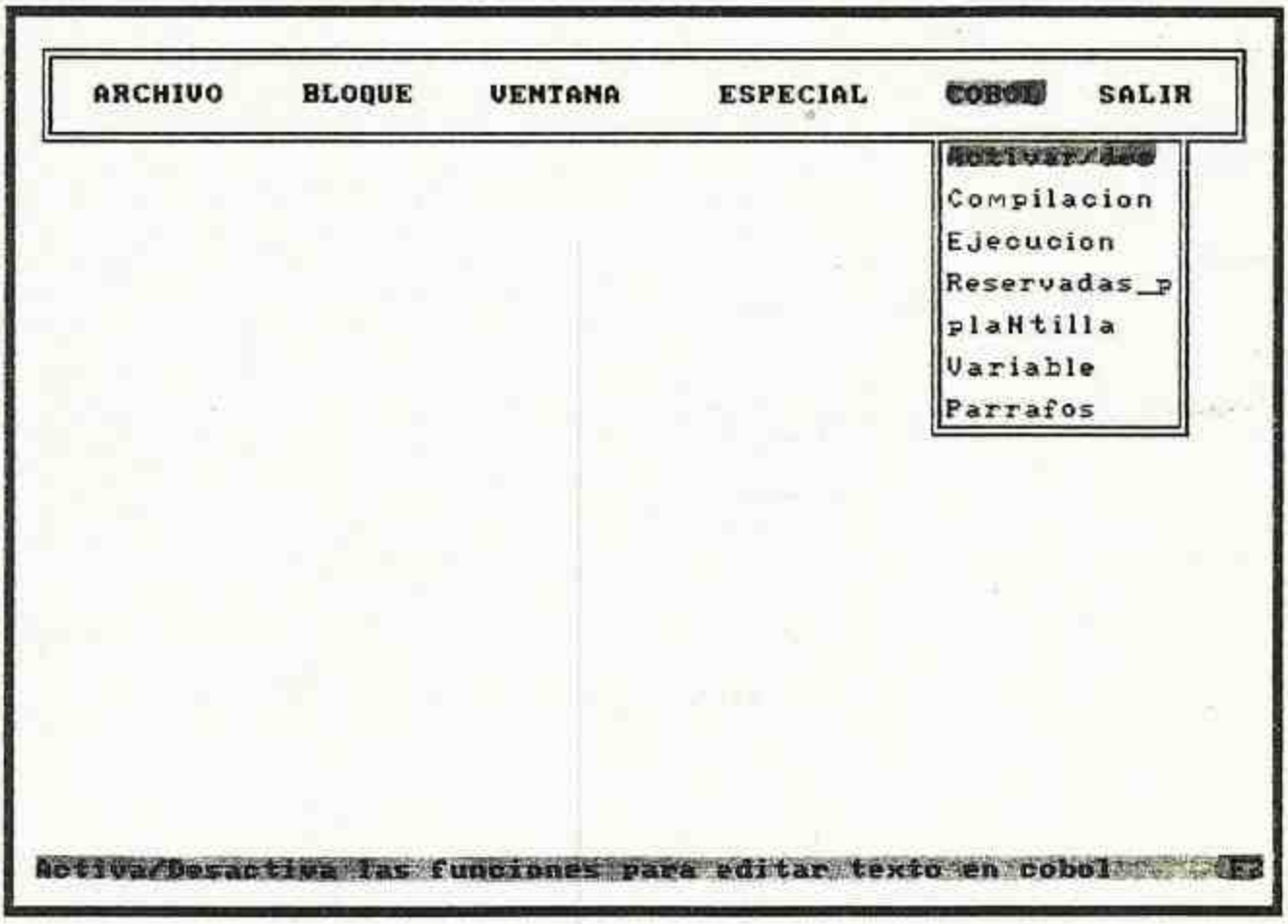
DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SEPT02	NOMBRE: REPETIR COMANDO	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



3.1.3.2.6 SUBMENU COBOL.

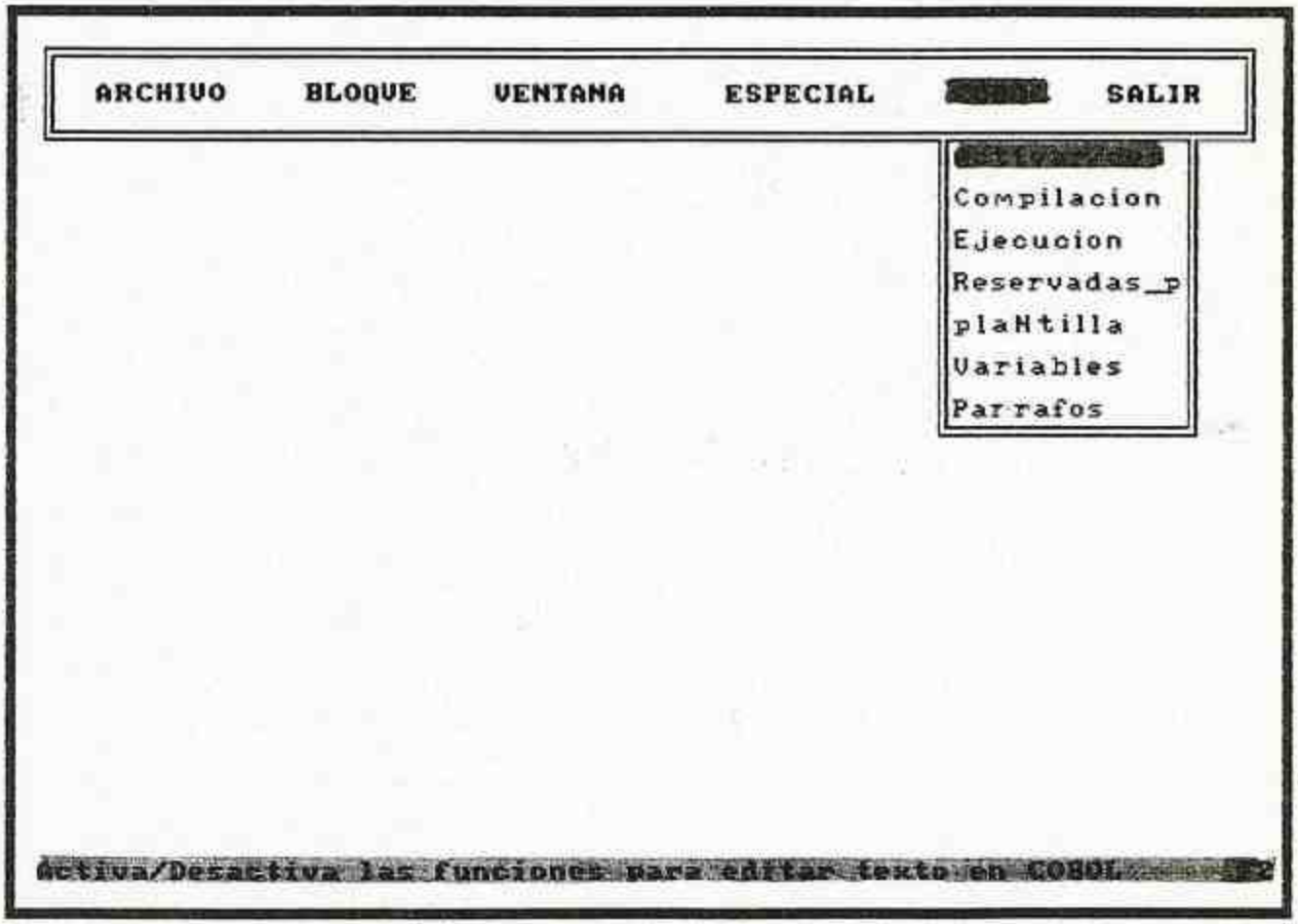
Submenú de funciones COBOL.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SCPT00	NOMBRE: SUBMENU COBOL	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



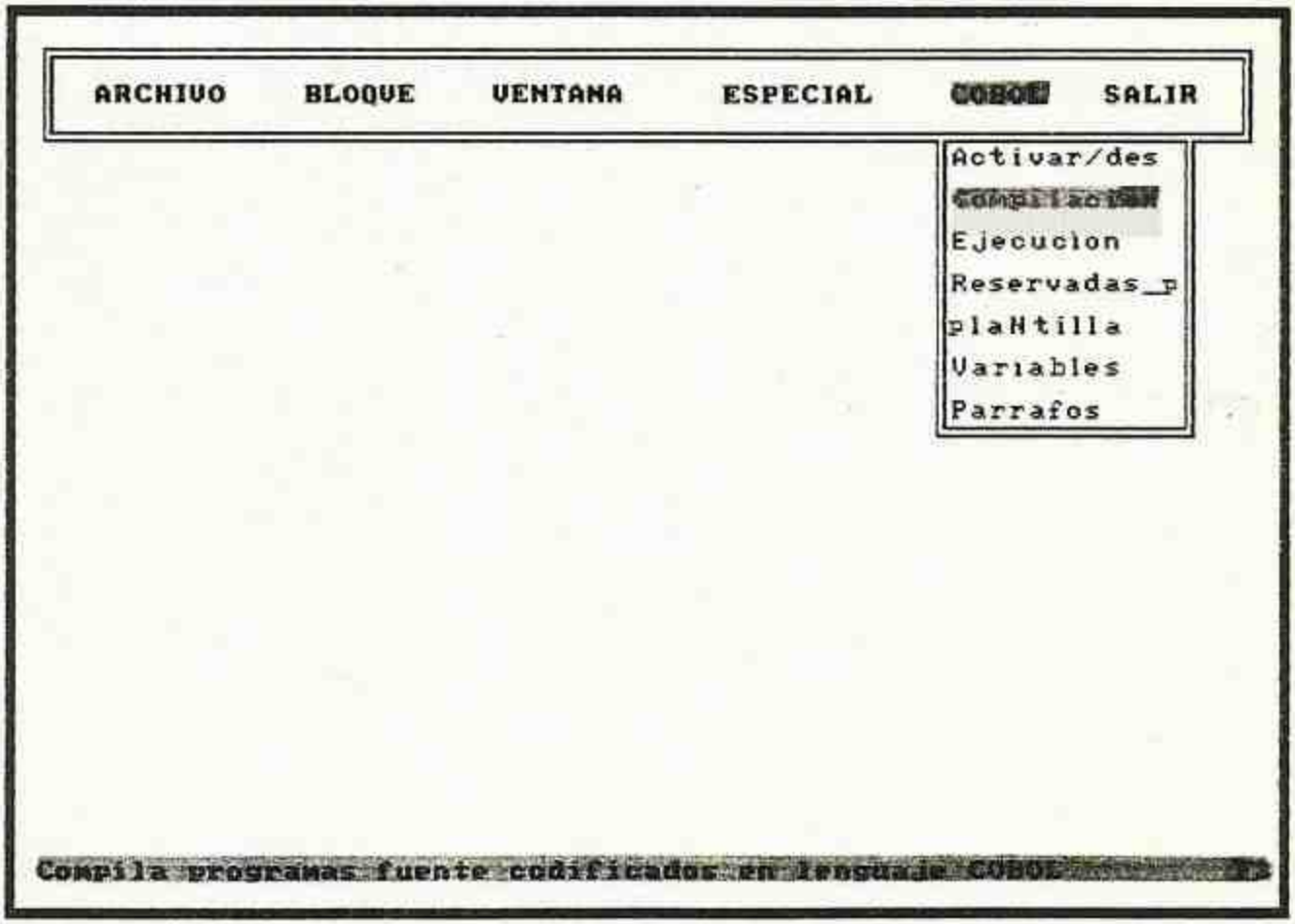
Activar/desactivar funciones.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SCPT01	NOMBRE: ACTIVAR/DESACTIVAR FUNCIONES	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



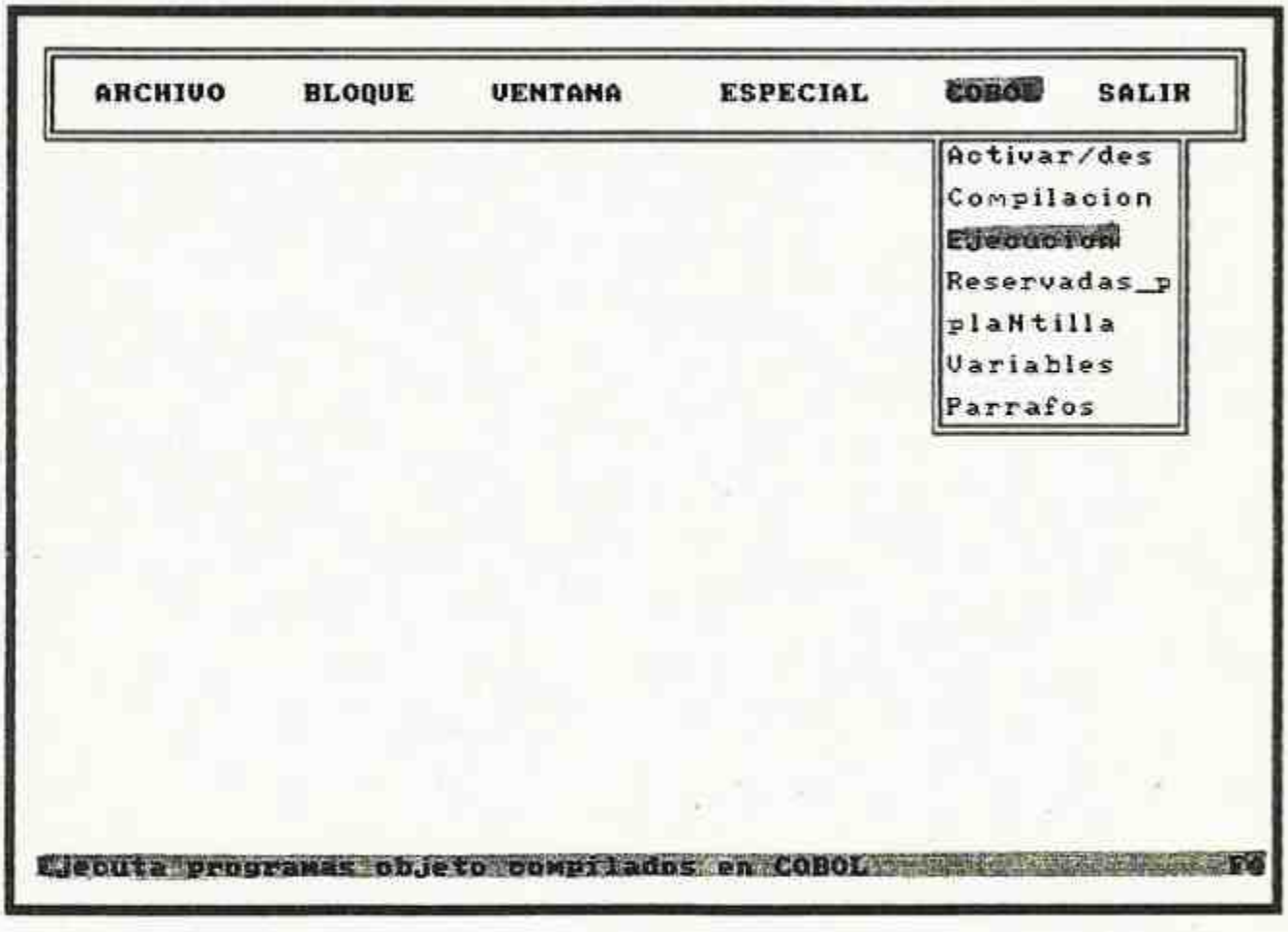
Compilar programa.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **SCPT02**	NOMBRE: **COMPILAR PROGRAMA**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



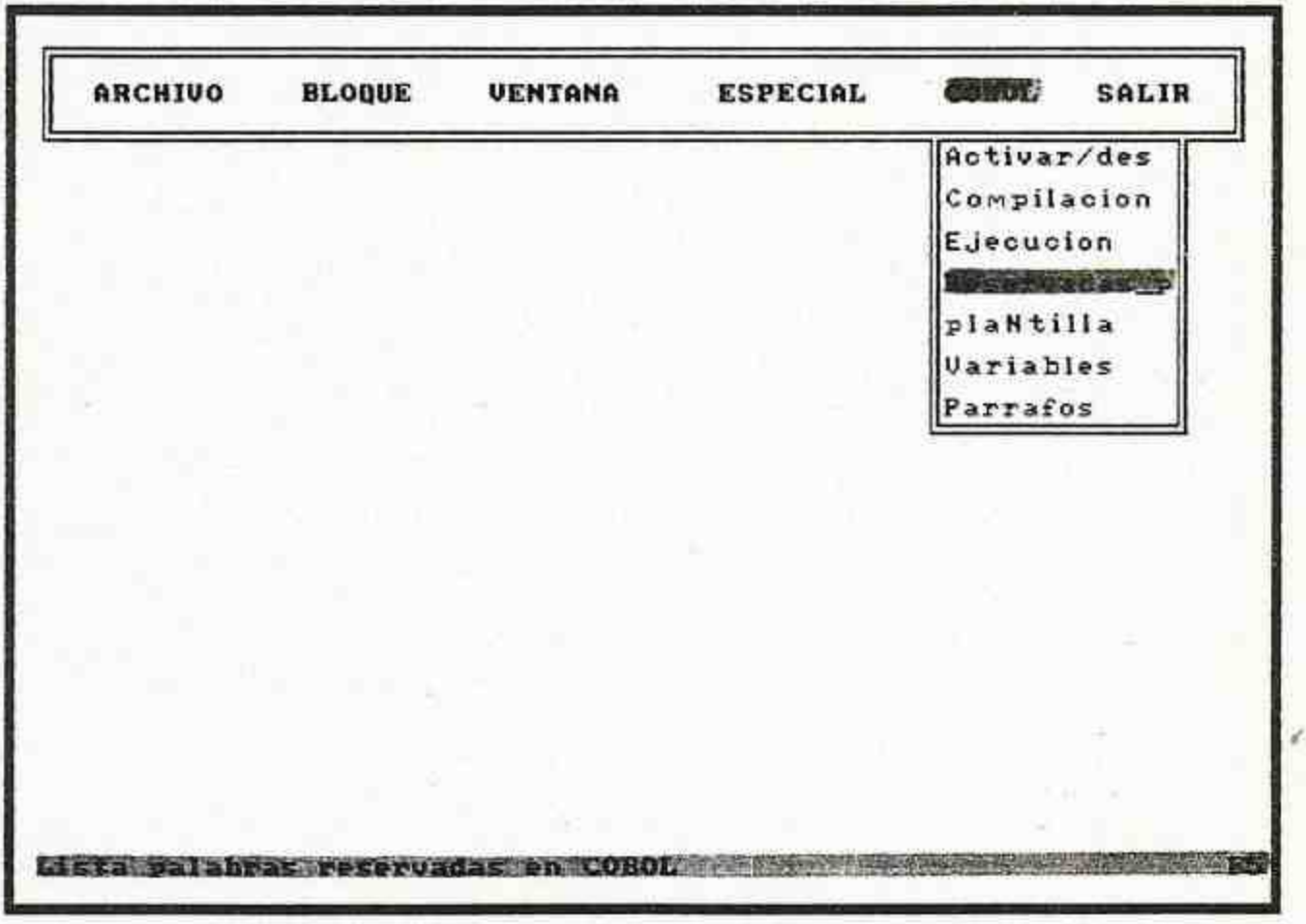
Ejecutar programa.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **SCPT03**	NOMBRE: **EJECUTAR PROGRAMA**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



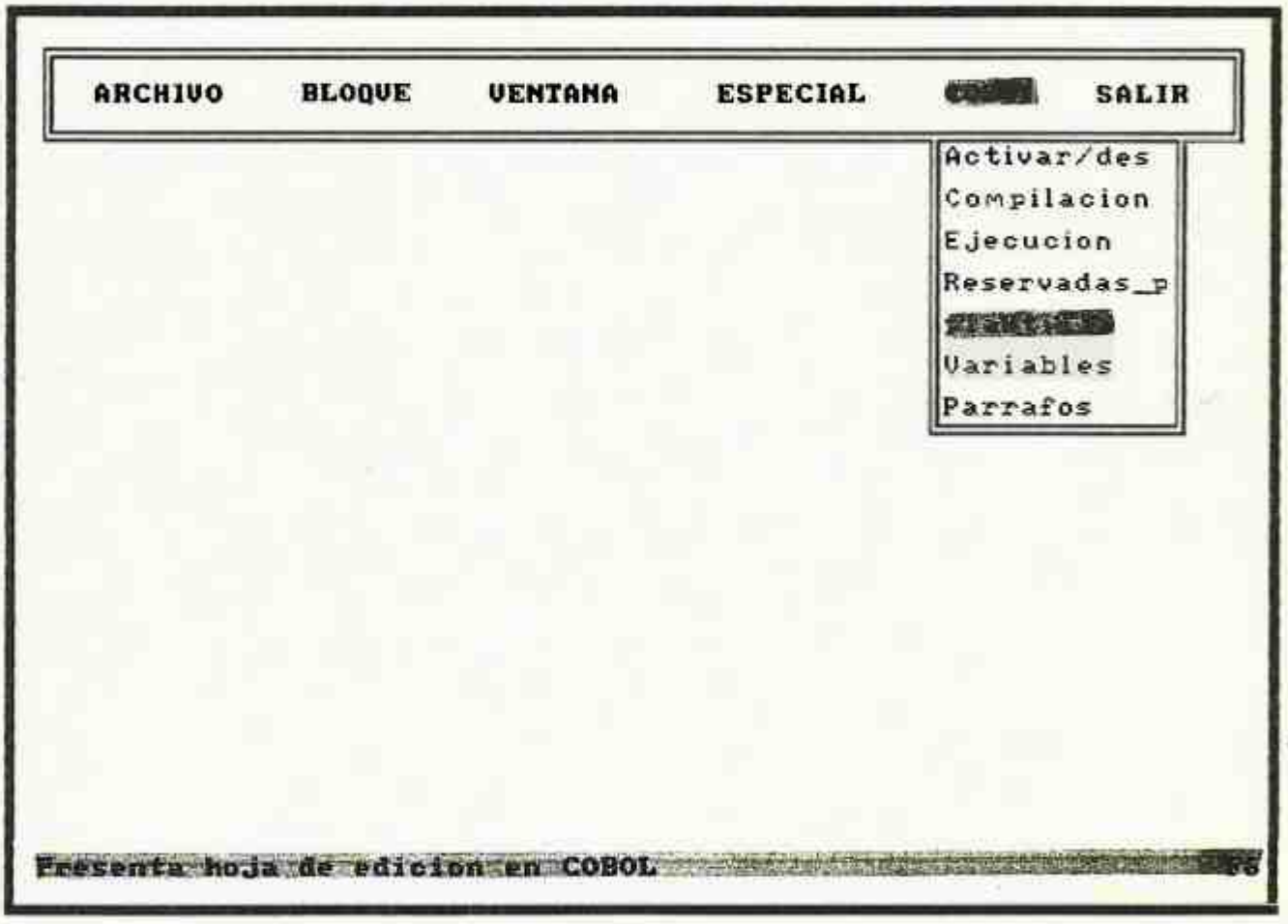
Mostrar listado de palabras reservadas.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: **SCPT04**	NOMBRE: **MOSTRAR LISTADO DE PALABRAS RESERVADAS**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



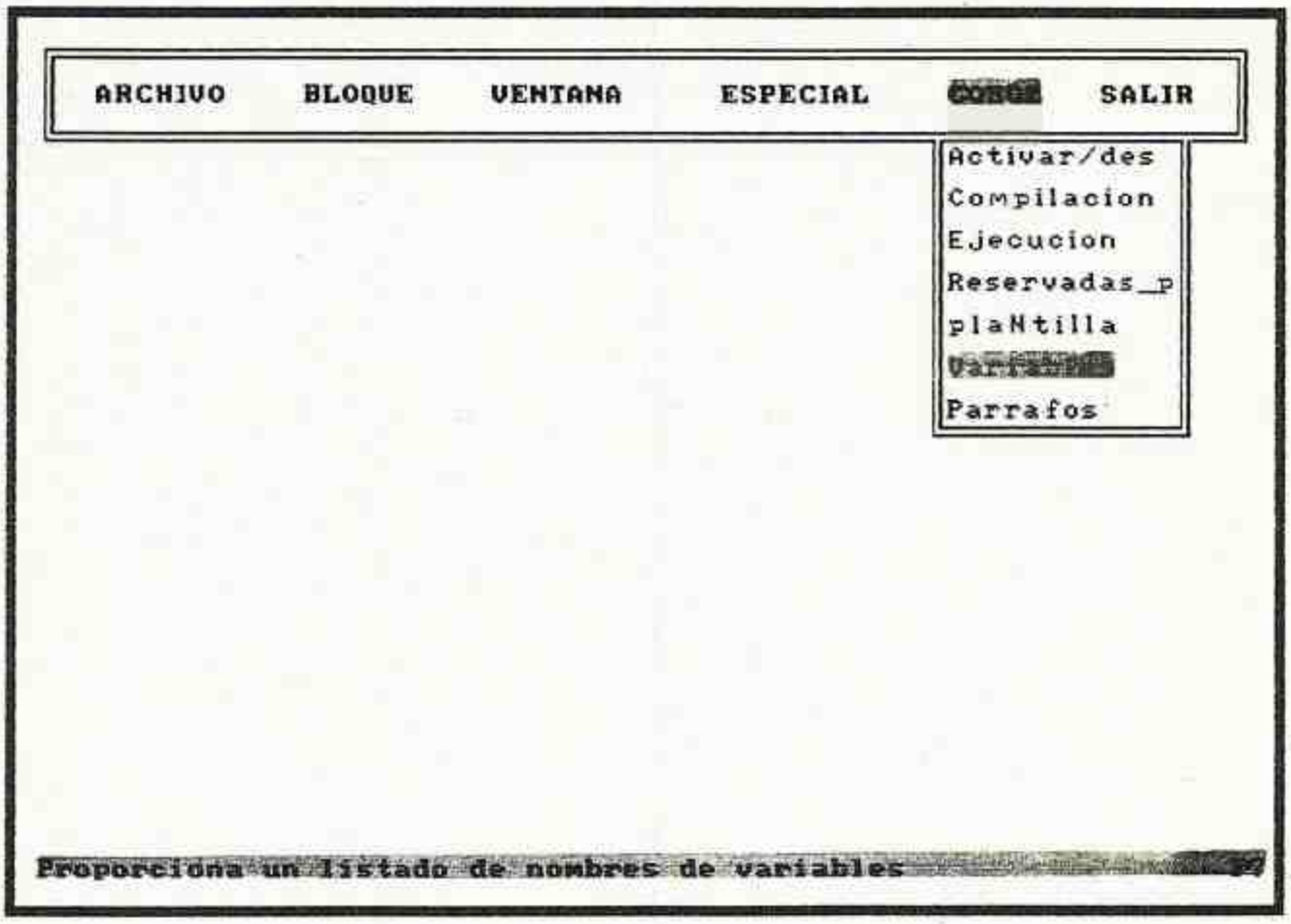
Mostrar plantilla.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SCPT05	NOMBRE: MOSTRAR PLANTILLA	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



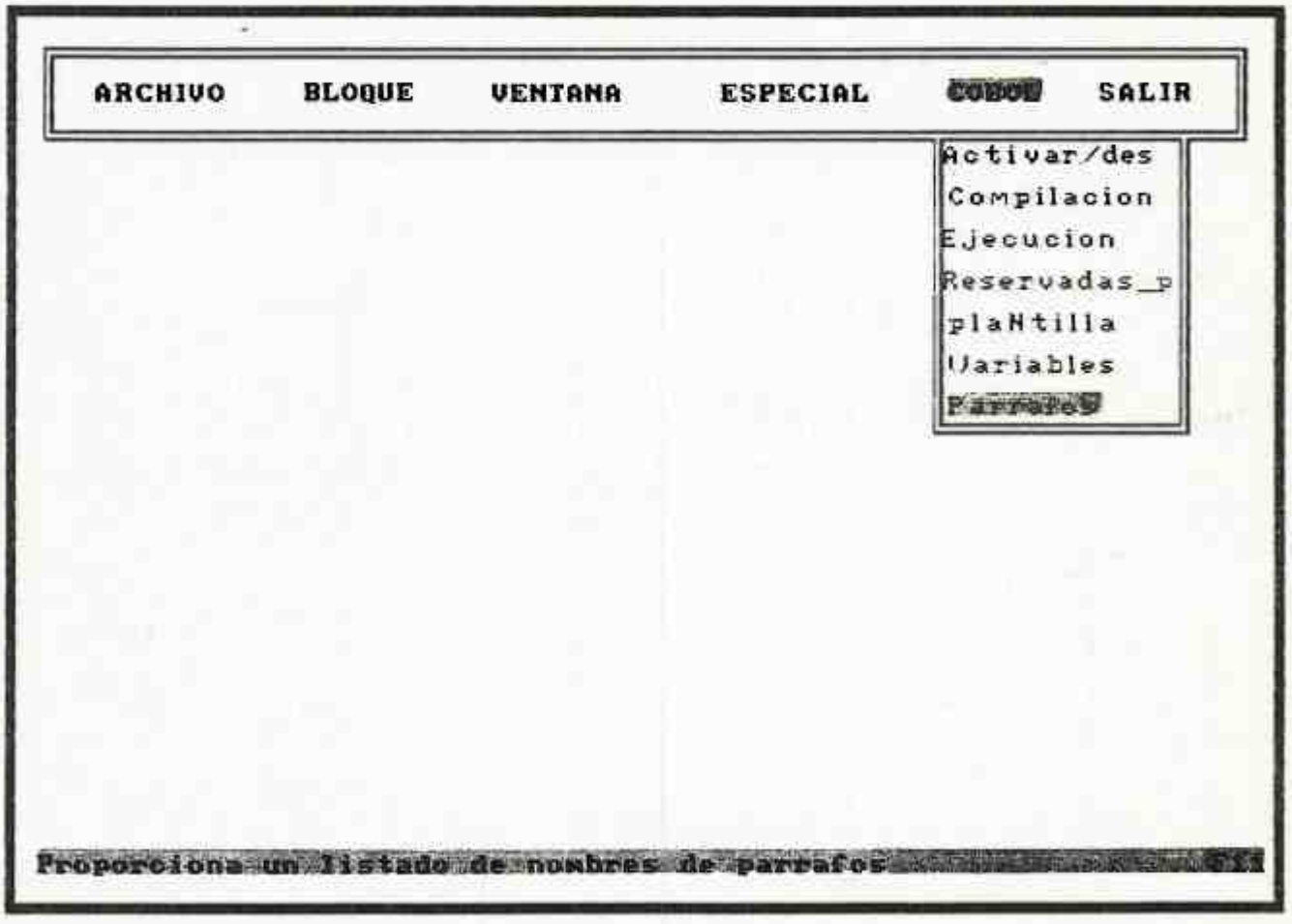
Mostrar Variables.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SCPT06	NOMBRE: MOSTRAR VARIABLES	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



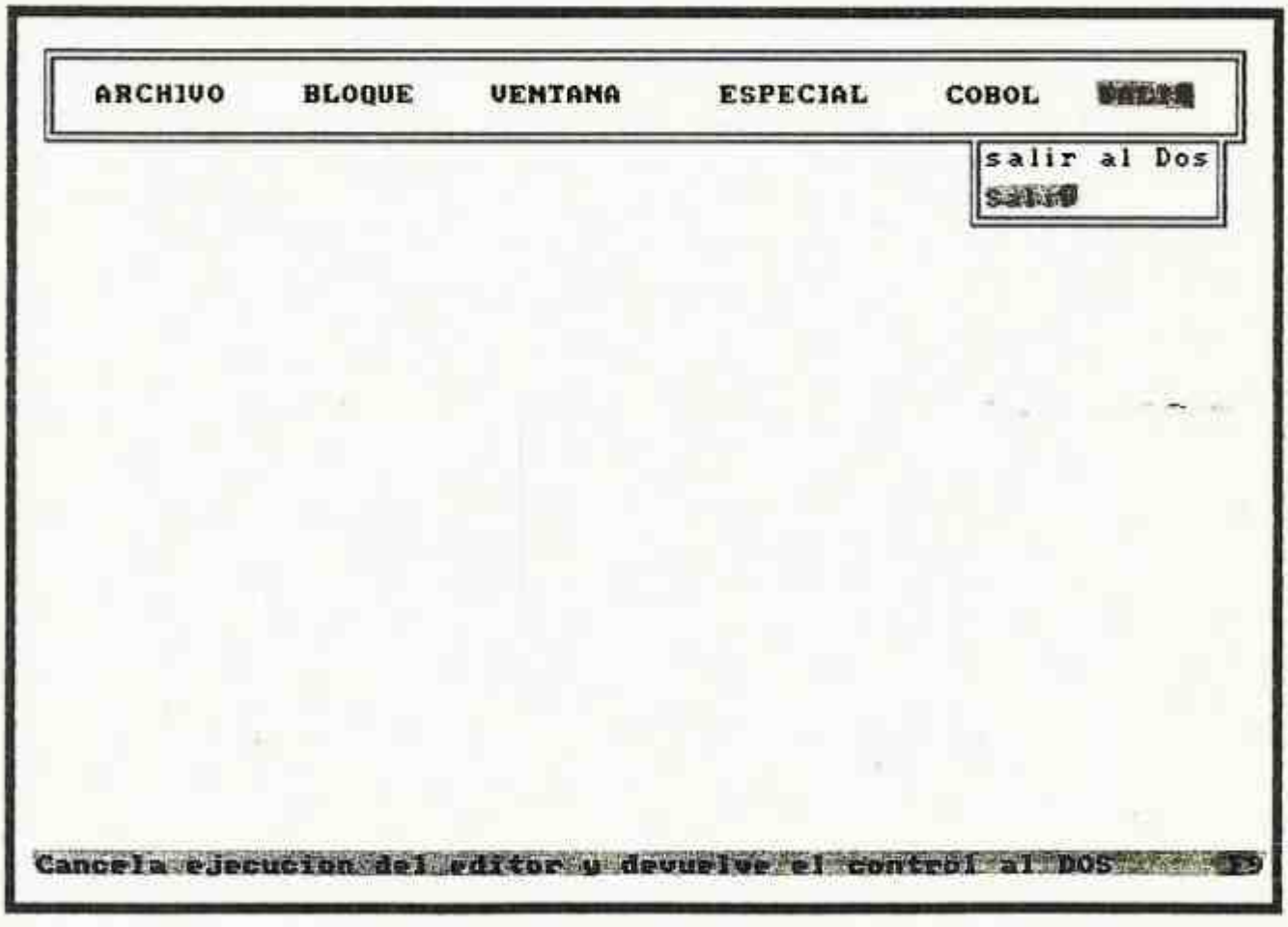
Mostrar Párrafos.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: SCPT07	NOMBRE: MOSTRAR PARRAFOS	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



3.1.3.2.7 Salir al DOS (fin de sesión).

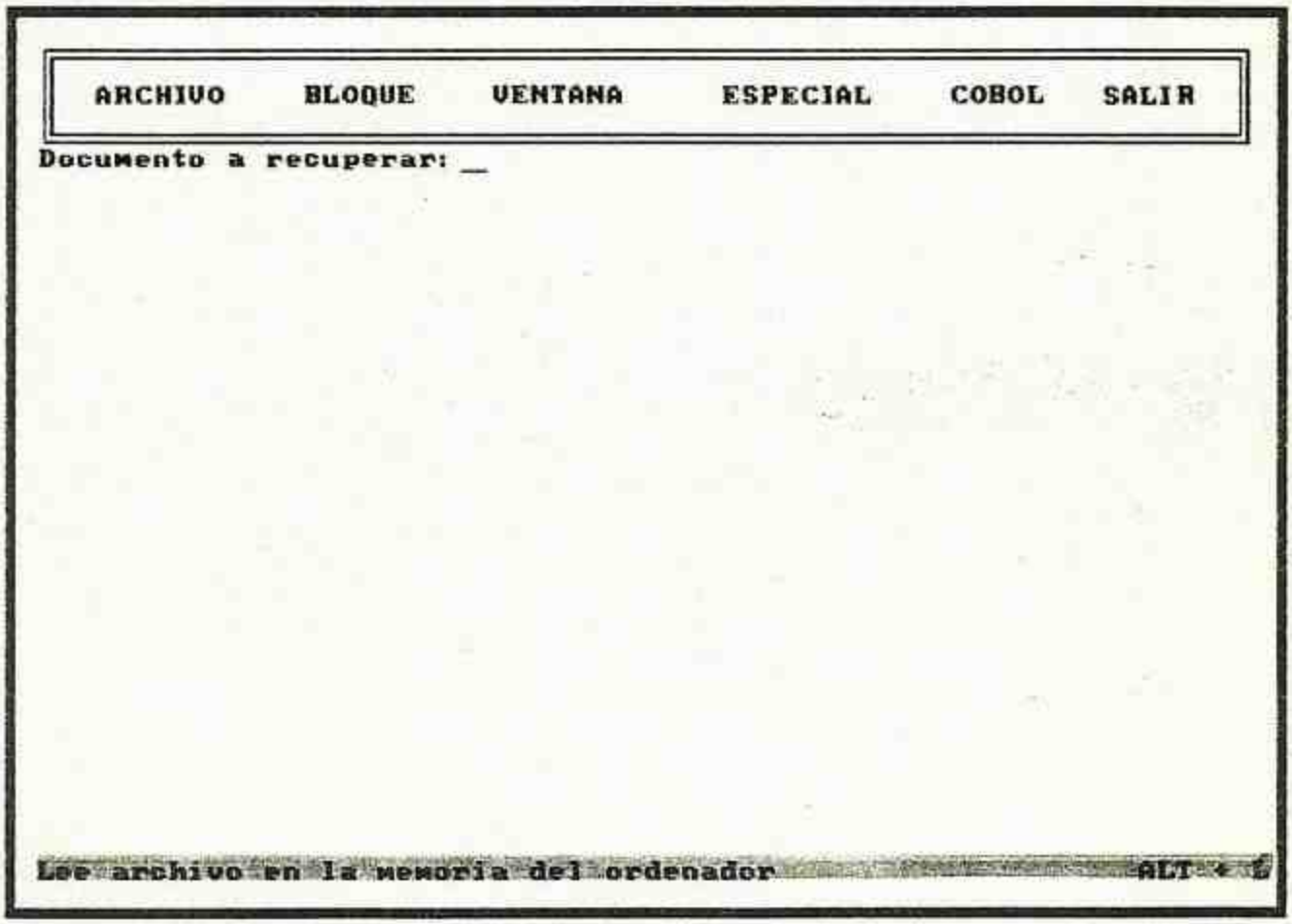
DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: SSPT02	NOMBRE: SALIR AL DOS(fin de sesion)	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



3.1.3.2.8 OTRAS PANTALLAS.

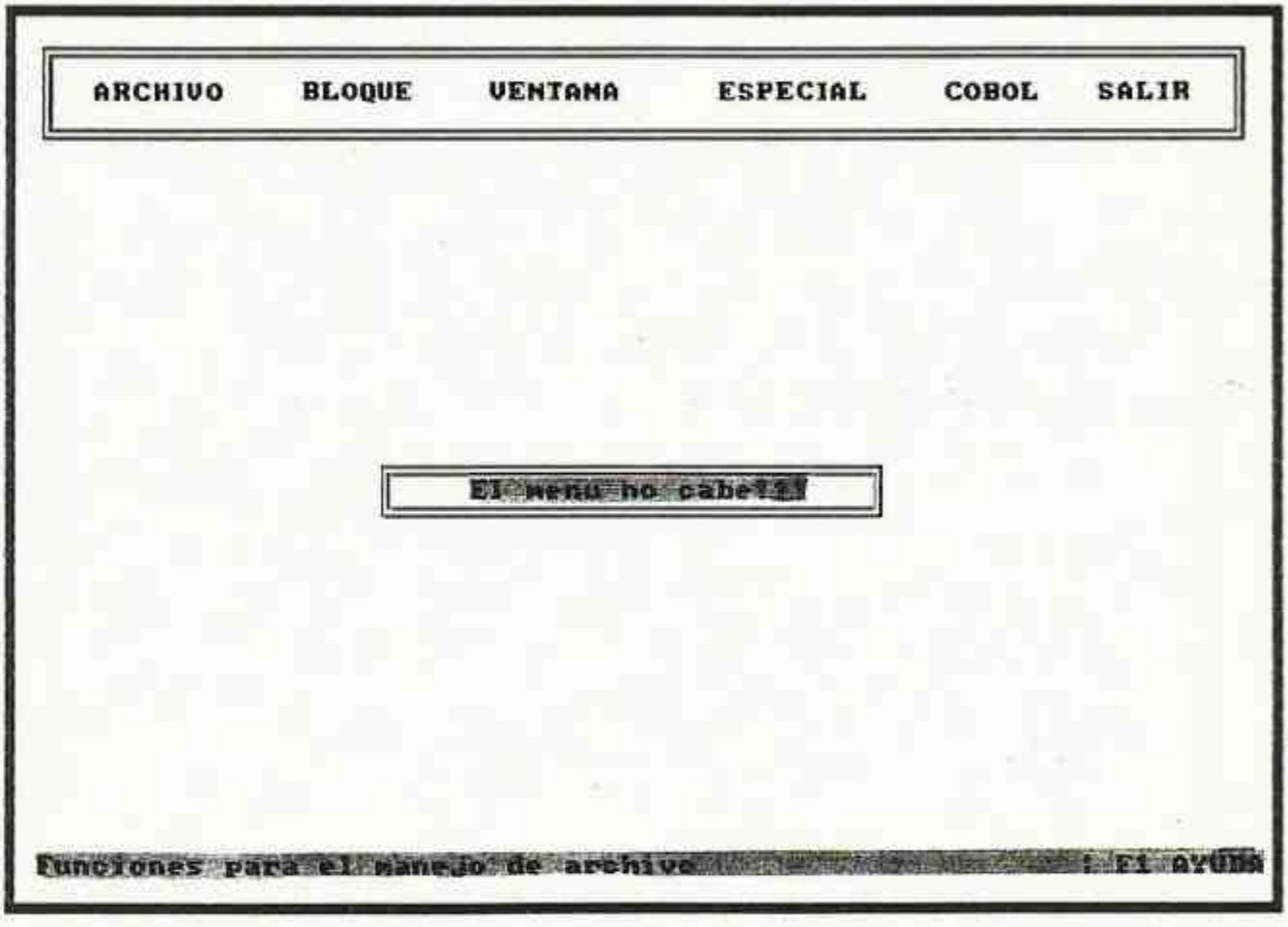
Mensaje al ejecutar la función leer "Documento a recuperar".

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: MEPT00	NOMBRE: EJECUTAR LA FUNCION LEER	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



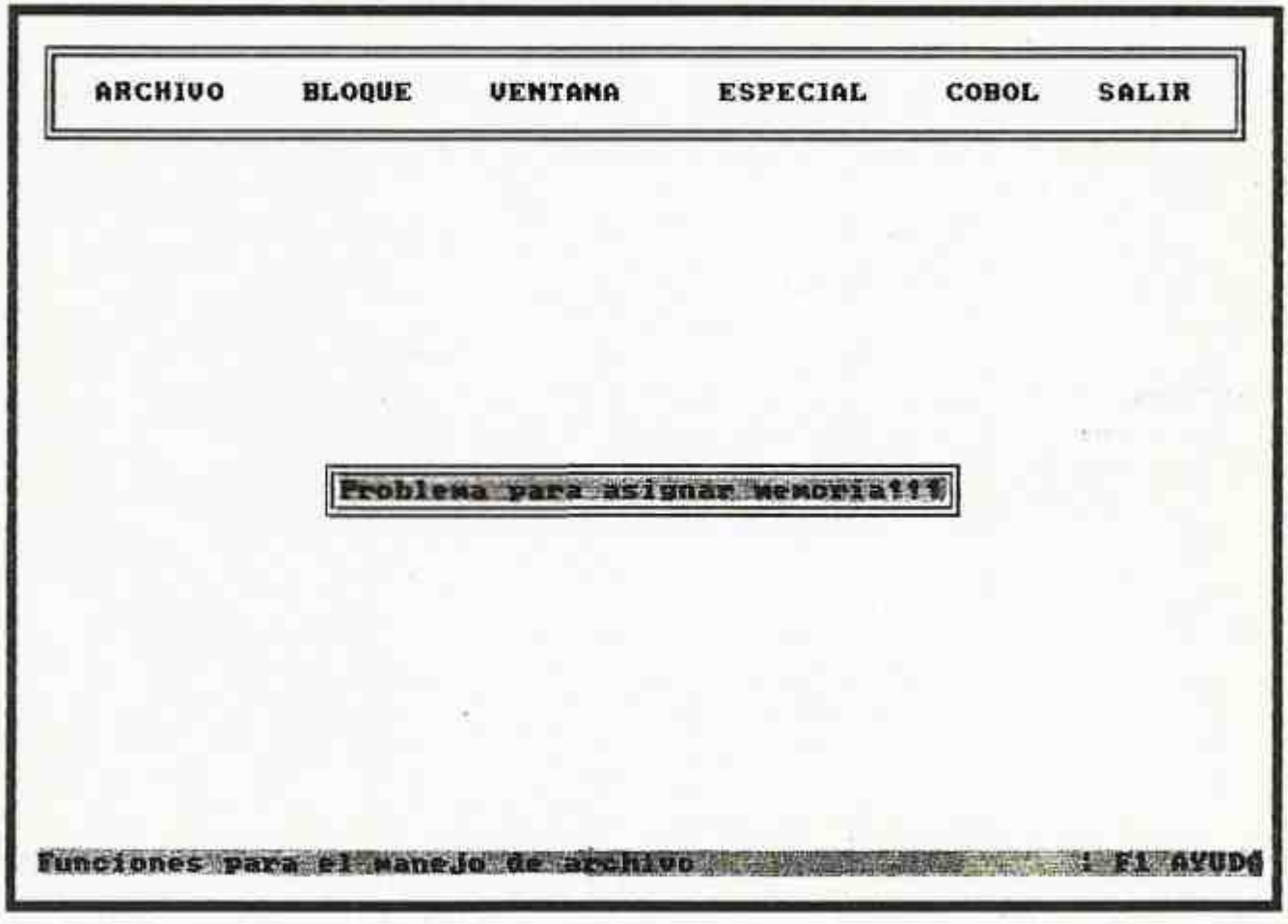
Mensaje de error si el menú no cabe en pantalla.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: MEPT03	NOMBRE: MENU DEMASIADO GRANDE	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



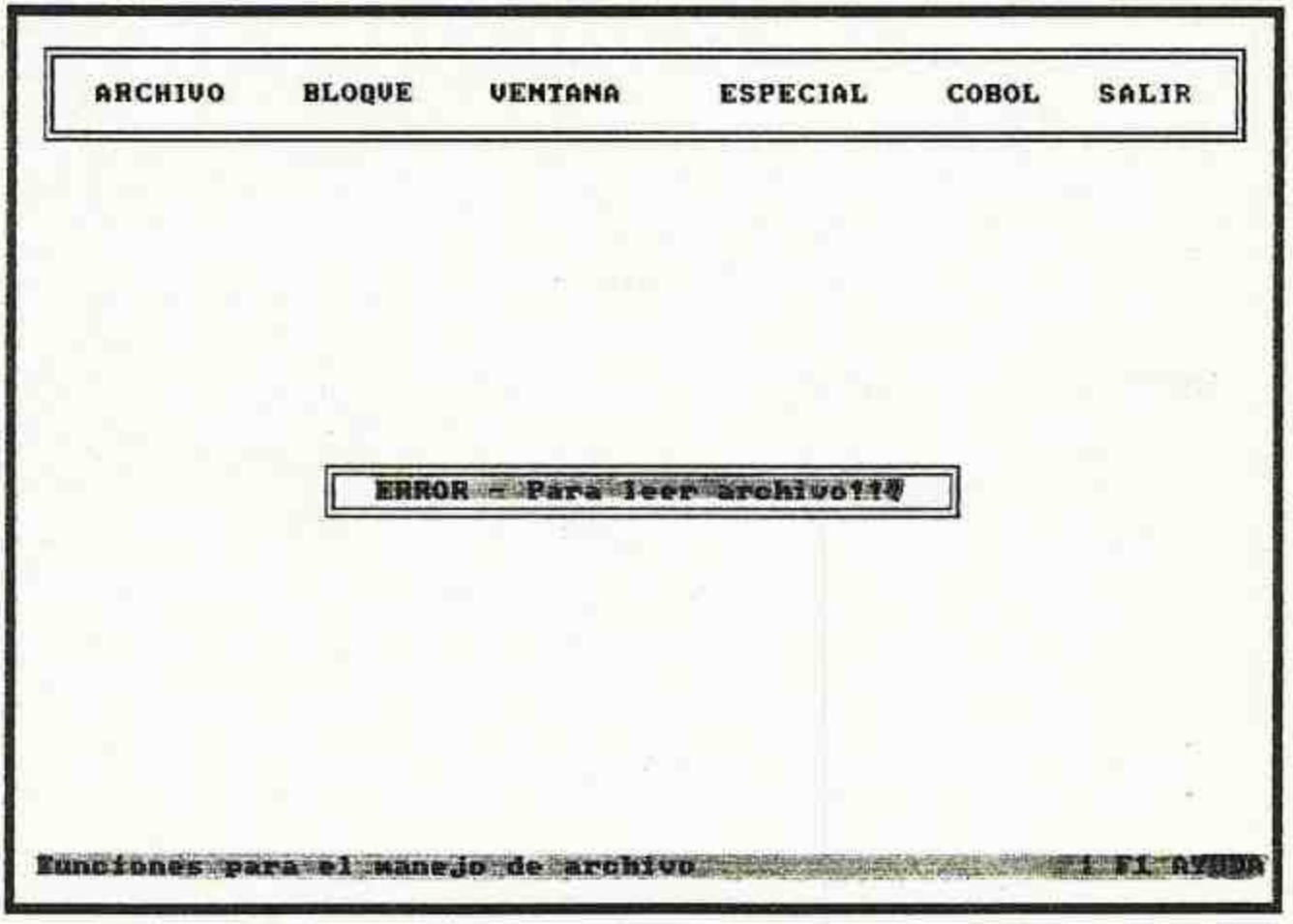
Mensaje de error al asignar memoria.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **MEPT04**	NOMBRE: **ERROR AL ASIGNAR MEMORIA**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



Mensaje de error al leer archivo.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: MEPT05	NOMBRE: ERROR AL LEER ARCHIVO	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93

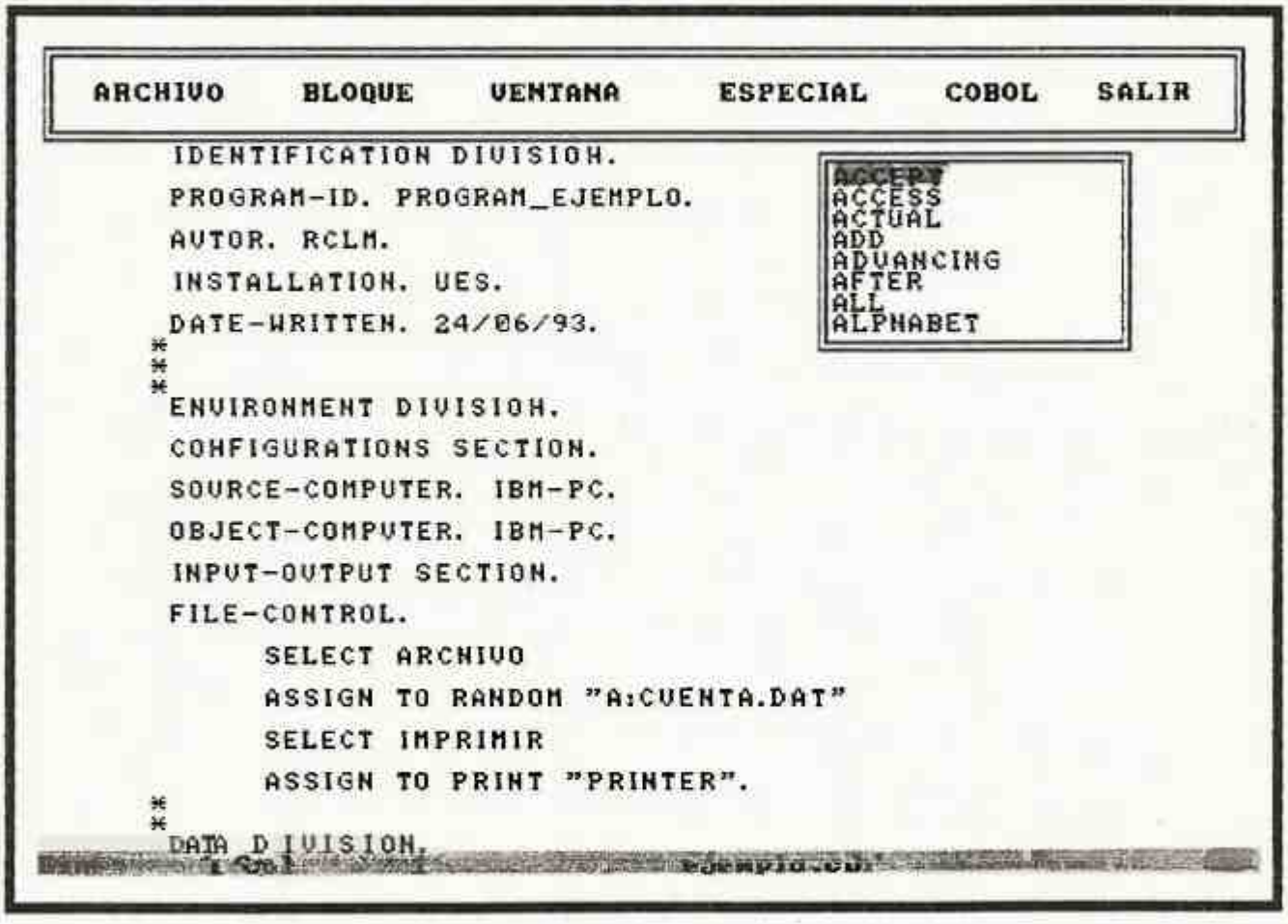


Mensaje si desea guardar el archivo antes de leer uno nuevo o al finalizar la sesión.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: MEPT08	NOMBRE: DESEA GUARDAR EL ARCHIVO	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93

Ventana de palabras reservadas.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: MEPT09	NOMBRE: VENTANA DE PALABRAS RESERVADAS	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



Plantilla numerada de COBOL.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **MEPT10**	NOMBRE: **PLANTILLA NUMERADA DE COBOL**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93

```
12345678901234567890123456789012345678901234567890123456789012345678901234
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PROGRAM_EJEMPLO.
       AUTOR. RCLM.
       INSTALLATION. UES.
       DATE-WRITTEN. 24/06/93.
      *
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATIONS SECTION.
       SOURCE-COMPUTER. IBM-PC.
       OBJECT-COMPUTER. IBM-PC.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT ARCHIVO
            ASSIGN TO RANDOM "A:CUENTA.DAT"
            SELECT IMPRIMIR
            ASSIGN TO PRINT "PRINTER".
      *
      *
       DATA DIVISION.
Ln  1  Col  1                    Ejemplo.cbl
```

Estructura.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: **MEPT11**	NOMBRE: **ESTRUCTURA**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93

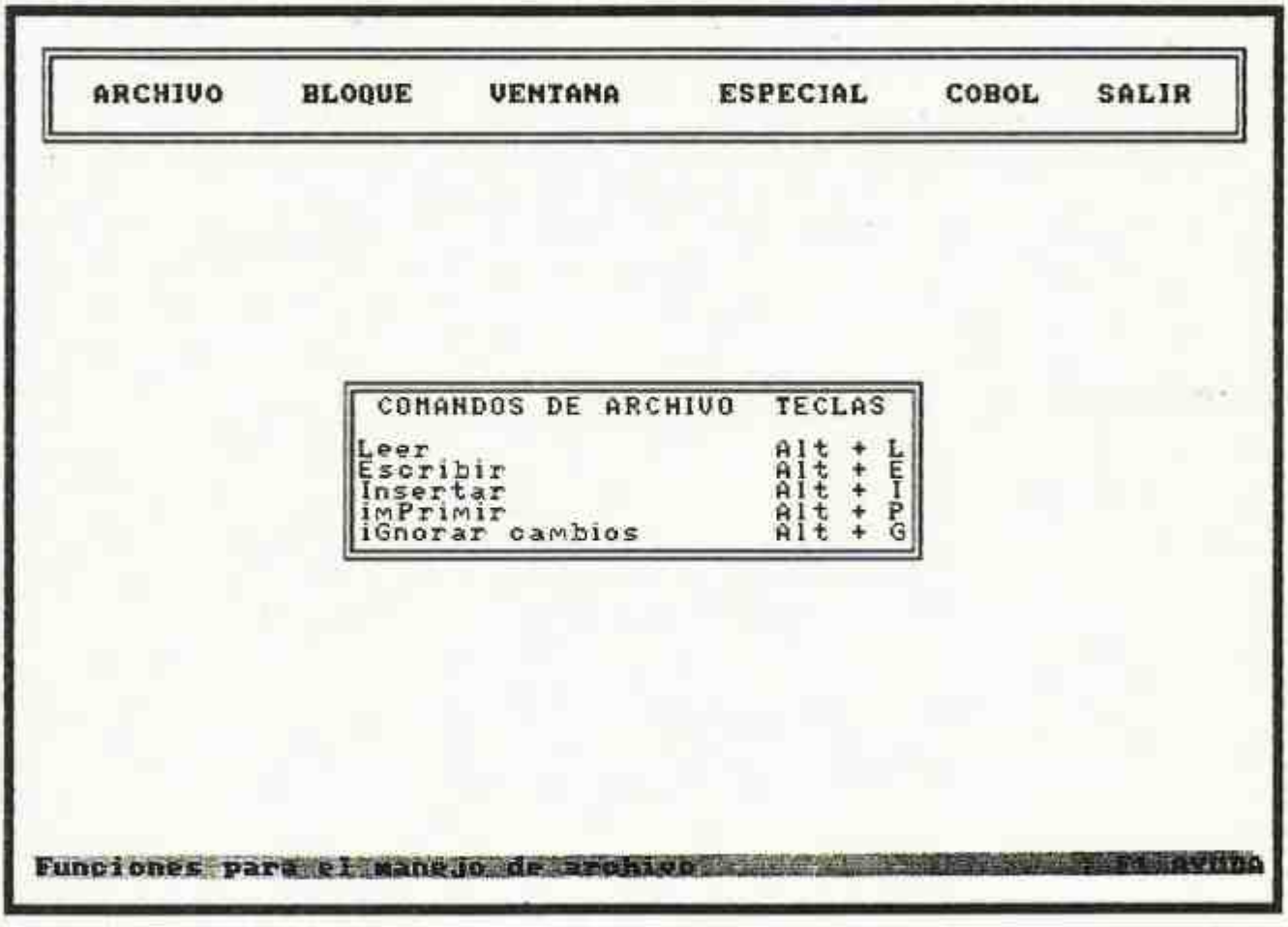
```
 ARCHIVO    BLOQUE    VENTANA    ESPECIAL    COBOL    SALIR

       IDENTIFICATION DIVISION.
       PROGRAM-ID. _
       AUTOR.
       INSTALLATION.
       DATE-WRITTEN.
       DATE-COMPILED.
       SECURITY.
      *
      *
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.
       OBJECT-COMPUTER.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT
            ASSIGN TO
      *
      *
      *
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
      *
      *
      *
       PROCEDURE DIVISION.
       STOP RUN.
Lin      2 Col     26      estruct.cbl                COBOL ACTIVO
```

3.1.3.2.9 AYUDA DEL FCEDIT.

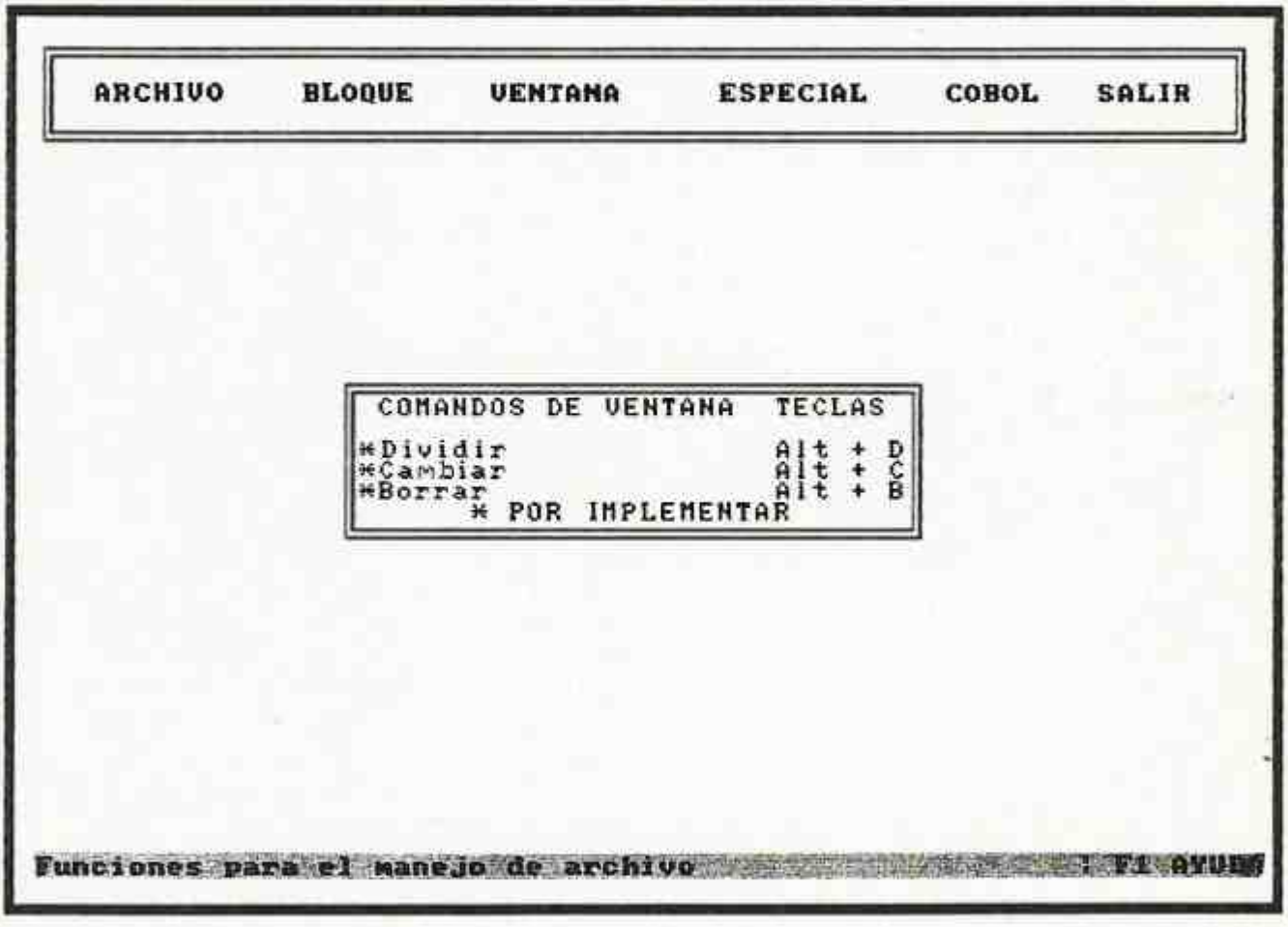
Funciones de archivo.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA01**	NOMBRE: **FUNCIONES DE ARCHIVO**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



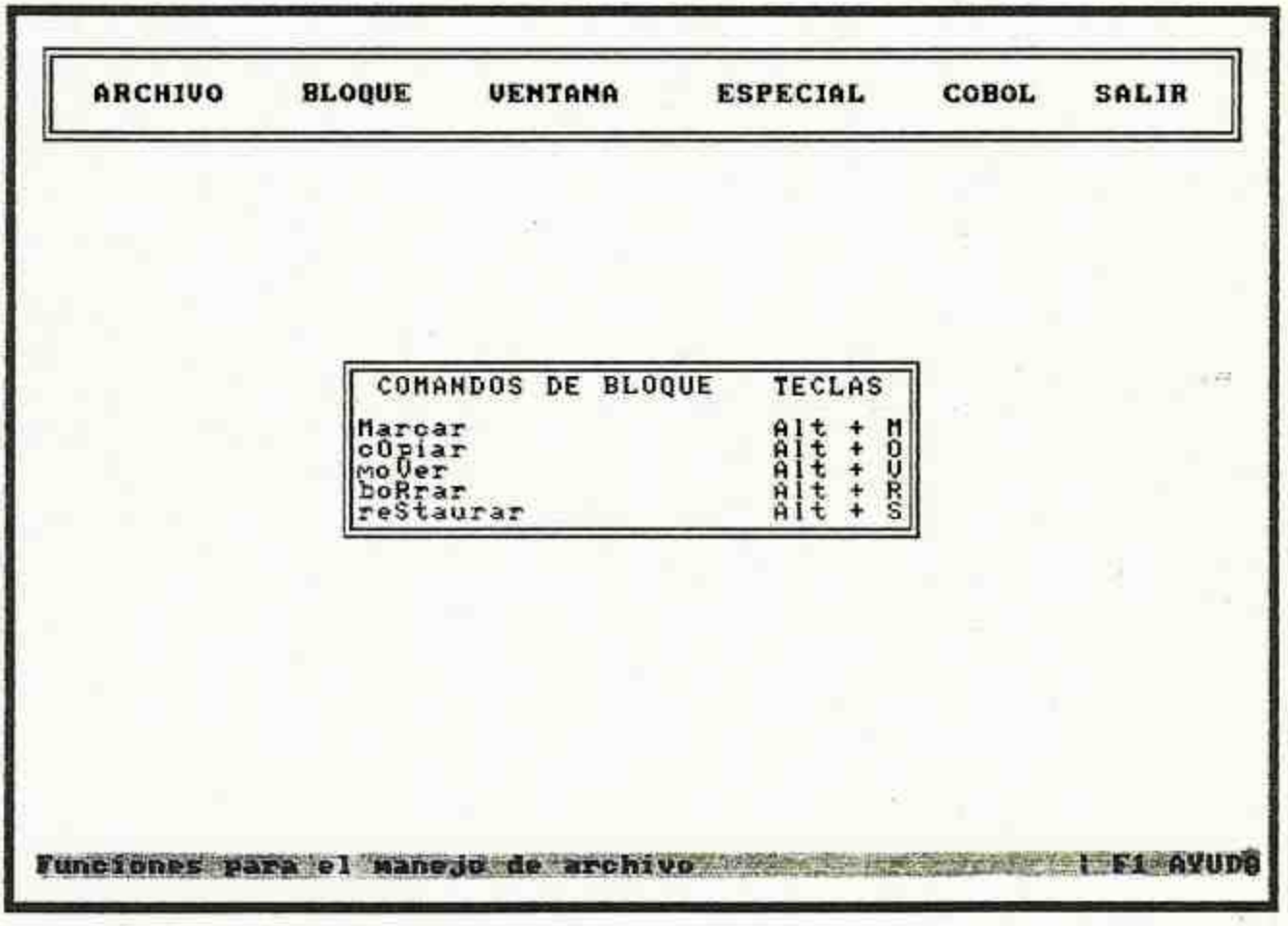
Funciones de ventana.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA02**	NOMBRE: **FUNCIONES DE VENTANA**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



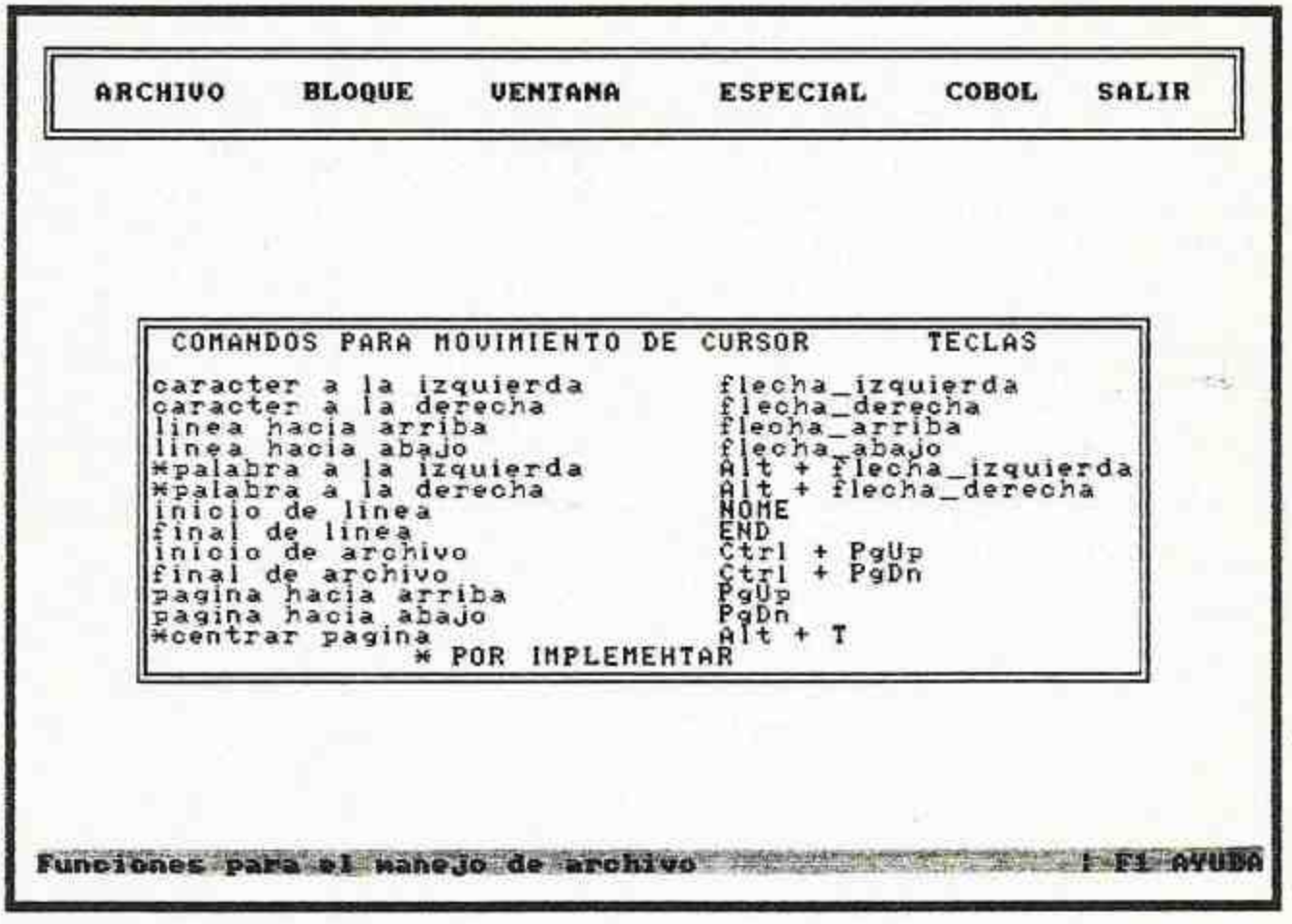
Funciones de bloque.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA03**	NOMBRE: **FUNCIONES DE BLOQUE**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



Funciones de cursor.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: AYUDA04	NOMBRE: FUNCIONES DE CURSOR	
PROGRAMA:	PROGRAMADOR: R.C.L. y F.A.G.	FECHA: SEPTIEMBRE/93



Funciones de insertar/borrar.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA05**	NOMBRE:**FUNCIONES DE INSERTAR/BORRAR**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93

ARCHIVO BLOQUE VENTANA ESPECIAL COBOL SALIR

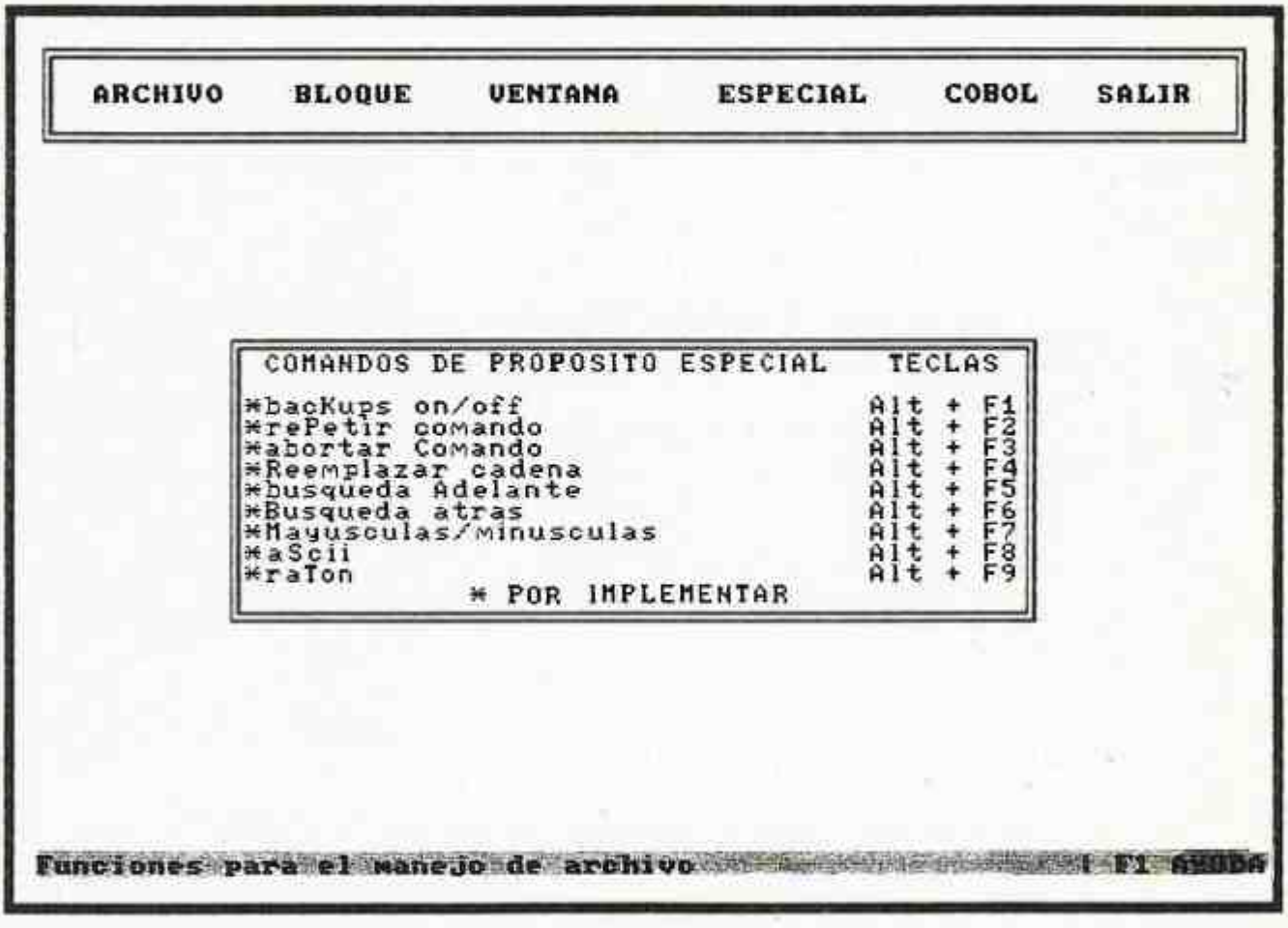
COMANDOS PARA INSERTAR/BORRAR	TECLAS
borrar caracter a la izquierda	BACKSPACE
borrar caracter a la derecha	DELETE
borrar linea	Alt + Y
borrar hasta el final de linea	Ctrl + END
borrar hasta el inicio de linea	Ctrl + HOME
*borrar una palabra a la izquierda	Alt + BACKSPACE
*borrar una palabra a la derecha	Alt + DELETE
*insertar linea	Alt + N
insertar caracter	caracter

* POR IMPLEMENTAR

Funciones para el manejo de archivo | F1 AYUDA

Funciones de propósito especial.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA06**	NOMBRE: **FUNCIONES DE PROPOSITO ESPECIAL**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



Funciones de cobol.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA07**	NOMBRE: **FUNCIONES DE COBOL**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93

ARCHIVO BLOQUE VENTANA ESPECIAL COBOL SALIR

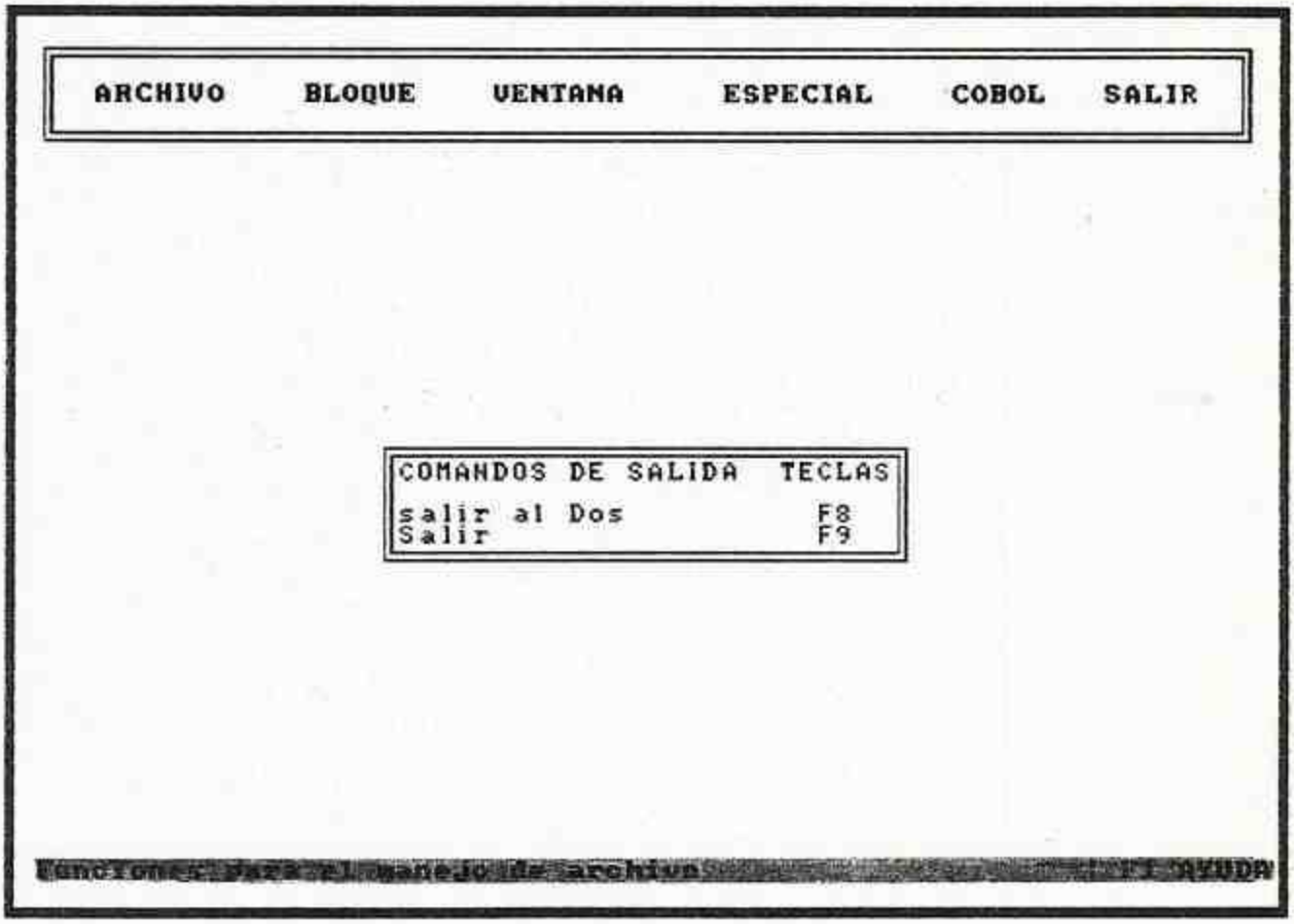
COMANDOS DE COBOL	TECLAS
Activar/desactivar	F2
Compilacion	F3
Ejecucion	F4
palabras Reservadas	F5
Mostrar plaNtilla	F6
*Variables	F7
*Parrafos	F11

* POR IMPLEMENTAR

Funciones para el manejo de archivo : F1 AYUDA

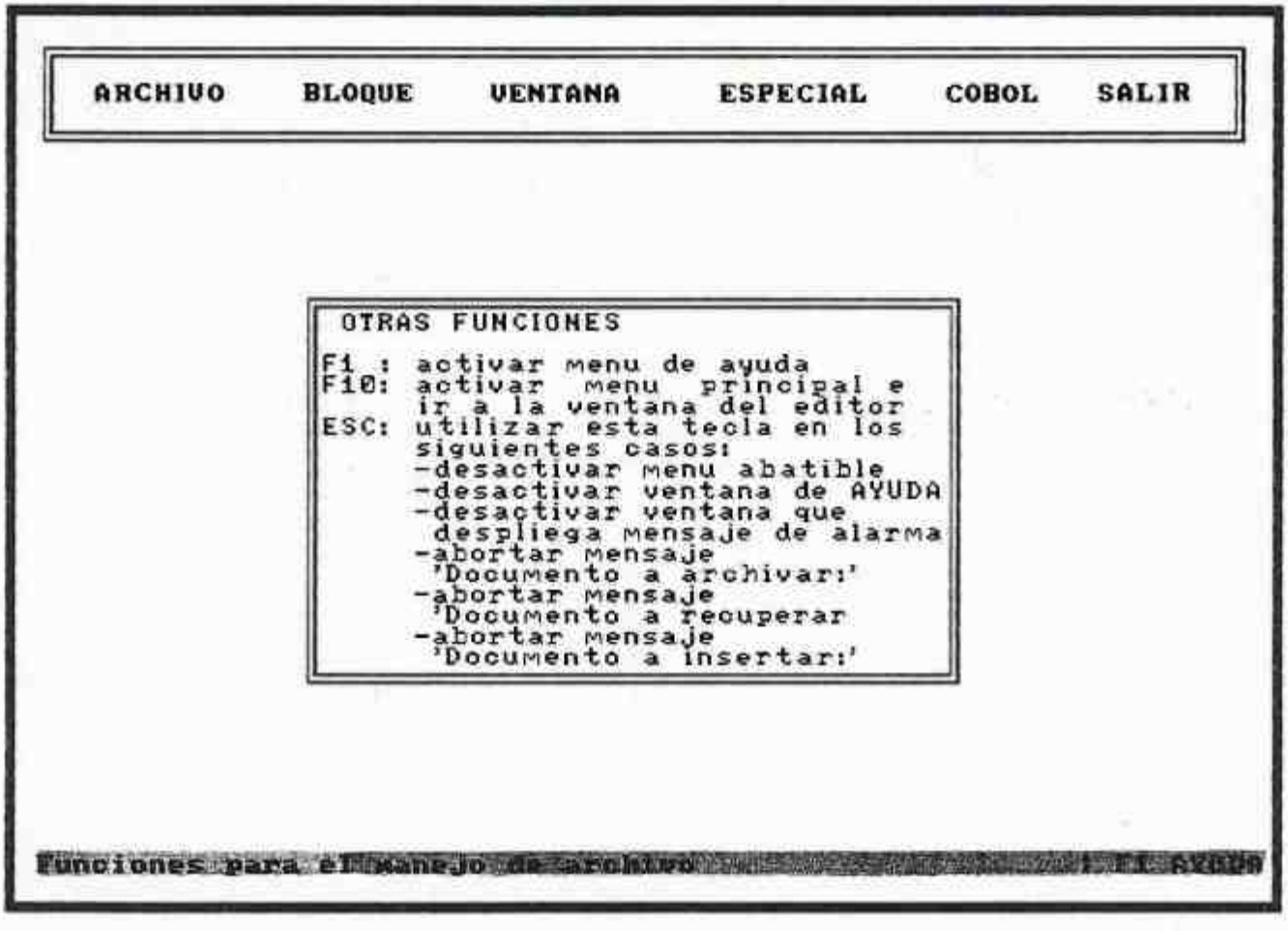
Funciones de salida.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA08**	NOMBRE: **FUNCIONES DE SALIDA AL DOS**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



Otras funciones.

DISENO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA09**	NOMBRE: **OTRAS FUNCIONES**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/93



Ayuda de contexto.

DISEÑO DE DESPLIEGUE DE PANTALLA		
CODIGO: **AYUDA10**	NOMBRE: **AYUDA IMPORTANTE**	
PROGRAMA:	PROGRAMADOR: **R.C.L. y F.A.G.**	FECHA: SEPTIEMBRE/'93

ARCHIVO **BLOQUE** **VENTANA** **ESPECIAL** **COBOL** **SALIR**

IMPORTANTE

Las funciones del módulo COBOL se habilitan únicamente cuando se está trabajando código fuente en COBOL. Responder a la pregunta: Nombre de compilador? con el nombre del compilador que utilizas, de la forma que lo harías si estuvieras en el DOS. Por ejemplo, si utilizas RMCOBOL la respuesta correcta es RM. Cuando ejecutes la función Activar/des y el BUFFER este limpio, el FCEDIT carga un archivo llamado estruct.cbl que contiene la estructura general de un programa en COBOL, este archivo debe estar en el directorio del FCEDIT, si la operación no tiene éxito se despliega un mensaje de error. Si ejecutas esta función teniendo un programa COBOL en memoria se activa la tecla TAB para mover el cursor a las columnas 8, 12, y 72, y además se despliega en la línea de estado el rótulo COBOL ACTIVO. Ejecutar por segunda vez esta función para desactivar estas facilidades. Situación similar se presenta cuando ejecutas la función Reservada_p, esta despliega una lista de palabras reservadas las cuales puedes accesar presionando ENTER cuando señale la palabra que te interesa.

Funciones para el manejo de archivo **(F1 AYUDA**

3.1.4 Estructuras de datos a utilizar.

El diseño del FCEDIT se basa en el uso de una estructura dinámica de datos que además de dar la facultad de poder hacer variar su tamaño, permite insertar, borrar, modificar, etc., el archivo de texto o código fuente en COBOL, cargado en el entorno del editor.

Esta estructura dinámica de datos a la que se hace referencia es una lista doblemente enlazada con cabecera, la que se identifica, en lo que sigue, con el nombre de LISTA. La técnica utilizada para el manejo de LISTA es la de hacer asignaciones dinámicas de memoria para los componentes individuales al tiempo que estos son creados durante la ejecución del programa.

La ***figura-3.5****, muestra en forma lógica el diseño de LISTA, la cual es manipulada por cuatro punteros:*

cab : puntero al nodo cabecera.

inicio: puntero al primer nodo.

abajo: puntero que lee la lista en cualquier dirección.

ultimo: puntero al último nodo.

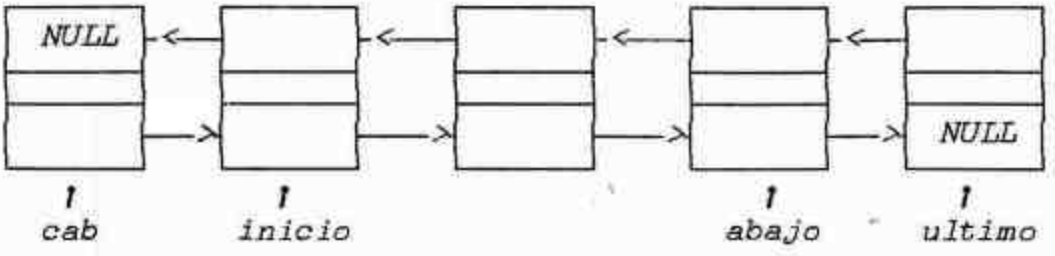


figura-3.5

En LISTA se llevan a cabo todos las operaciones que conllevan al buen funcionamiento de los comandos o funciones que integran el FCEDIT.

Con el propósito de documentar los procedimientos y facilitar su posterior codificación, se hace uso de la siguiente declaración, la cual obedece a la sintaxis del lenguaje C y muestra en primer lugar la estructura de cada elemento de LISTA y posteriormente la declaración de los punteros que se utilizan para realizar cualquier operación con los elementos de la misma.

```
struct editor{
              editor far *ant;
              char texto[81];
              editor far *sig;
             }
editor far *cab;
editor far *inicio;
editor far *abajo;
editor far *ultimo;
```

Como puede verse, tanto en la ***figura-3.5*** *como en la declaración anterior, cada elemento de LISTA tiene dos enlaces, uno al elemento anterior (editor far *ant) y otro al elemento siguiente (editor far *sig), esto es realmente lo que permite leer la LISTA en cualquier dirección; el otro campo que aún falta que describir es char texto[81], en este se almacena la información, la declaración obedece a que el FCEDIT maneja líneas de texto con una longitud máxima de 81*

caracteres (incluyendo el caracter de fin de línea), lo que implica que cada nodo o elemento de LISTA debe almacenar en su campo de datos una línea de texto, que corresponde a una línea visible en la pantalla.

Finalmente se declaran cuatro punteros far: cab, inicio, abajo y ultimo a estructuras de tipo editor, que son las que permiten manipular la LISTA, la decisión de utilizar una declaración de punteros far, es porque de esta forma se puede acceder a cualquier dirección o segmento de la memoria.

3.1.5 DOCUMENTACION DE MODULOS.

3.1.5.1 Módulo ARCHIVO.

Carga, graba, inserta, imprime archivos e ignora cambios en el archivo cargado en el entorno del FCEDIT.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO	
MODULO No. FCA00	NOMBRE DEL MODULO ARCHIVO
PROCEDIMIENTOS INCLUIDOS	
1. leer() 2. escribir() 3. insertar() 4. imprimir() 5. ignorar_c()	

3.1.5.2 Módulo BLOQUE.

Lleva a cabo las funciones básicas para manipular bloques de texto.

UNIVERSIDAD DE EL SALVADOR *FACULTAD DE CC. NN. Y MAT.*	*ESCUELA DE MATEMATICA* *SEC. DE EST. Y COMPUTACION*
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL *FECHA SEP/93* *DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO* *MODULO No. FCB00* *NOMBRE DEL MODULO BLOQUE*	
PROCEDIMIENTOS INCLUIDOS	
1. marcar() *2. copiar()* *3. mover()* *4. borrar()* *5. restaurar()*	

3.1.5.3 Módulo VENTANA

Divide la pantalla en dos ventanas, permite cambiar de una ventana a otra y borrar la ventana inactiva.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93 DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO MODULO No. FCV00 NOMBRE DEL MODULO VENTANA	
PROCEDIMIENTOS INCLUIDOS	
1. dividir() 2. cambiar() 3. borrar()	

3.1.5.4 Módulo CURSOR.

Este módulo dá la facilidad de ejecutar los siguientes movimientos de cursor: moverse un caracter a la derecha o a la izquierda, una línea hacia arriba o hacia abajo, una palabra a la derecha o a la izquierda, ir al inicio de línea o final de línea, ir al inicio de archivo o al final de archivo, ir página hacia arriba o hacia abajo y centrar página.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR_DE_TEXTO_ORIENTADO_A_COBOL FECHA SEP/93 DISEÑO ROSA_CRISTINA_LEON_Y_FREDYS_ARTURO_GARCIA_PORTILLO MODULO No. FCCOO NOMBRE DEL MODULO CURSOR	
PROCEDIMIENTOS INCLUIDOS	
1. tabulador() 2. caracter_a_la_izq() 3. caracter_a_la_der() 4. linea_hacia_arriba() 5. linea_hacia_abajo() 6. palabra_a_la_izq() 7. palabra_a_la_der() 8. inicio_de_linea() 9. inicio_de_arch() 10. final_de_arch() 11. final_de_linea() 12. pagina_hacia_arriba() 13. pagina_hacia_abajo() 14. centrar_pagina()	

3.1.5.5 Módulo INSERTAR/BORRAR

Este módulo permite efectuar las siguientes operaciones: borrar un caracter a la izquierda o a la derecha del cursor, borrar una línea, borrar hasta el final de línea o hasta el inicio de línea, borrar una palabra a la derecha o a la izquierda e insertar una línea o un caracter en la posición del cursor.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL. FECHA SEP/93 DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO MODULO No. FCIB00 NOMBRE DEL MODULO INSERTAR/BORRAR	
PROCEDIMIENTOS INCLUIDOS	
1. borrar_caracter_izquierdo() 2. borrar_caracter_derecho() 3. borrar_linea() 4. insertar_linea() 5. insertar_caracter() 6. borrar_hasta_final_de_linea() 7. borrar_hasta_inicio_de_linea() 8. borrar_palabra_izquierda() 9. borrar_palabra_derecha()	

3.1.5.6 Módulo ESPECIAL.

Lleva a cabo funciones de propósito especial como son: backups, repetir o abortar un comando, reemplazar una cadena, búsquedas de cadena de caracteres, convertir mayúsculas a minúsculas o viceversa, tabla de códigos ASCII y el uso del RATON.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL. FECHA SEP/93 DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO MODULO No. FCE00 NOMBRE DEL MODULO ESPECIAL	
PROCEDIMIENTOS INCLUIDOS	
1. backups() 2. repetir_comando() 3. abortar_comando() 4. reemplazar_cadena() 5. busqueda_arriba() 6. busqueda_atras() 7. mayuscula_minuscula() 8. ASCII() 9. RATON()	

3.1.5.7 Módulo COBOL.

Permite ejecutar funciones para:

1.- Compilar y ejecutar el programa COBOL que esté cargado en el entorno.

2.- Activar/desactivar el tabulador y el despliegue o no de la estructura general de un programa en COBOL.

3.- Ver y accesar una lista de palabras reservadas en COBOL y activar/desactivar una plantilla para auxiliar el trabajo de edición de programas en COBOL.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93 DISEÑO ROSA CRISTINA LEON Y FREDYS ARTURO GARCIA PORTILLO MODULO No. FCC000 NOMBRE DEL MODULO COBOL	
PROCEDIMIENTOS INCLUIDOS	
1. activar_des() 2. compilar() 3. ejecutar() 4. reservadas_p() 5. plantilla() 6. variables() 7. parrafos()	

3.1.5.8 Módulo SALIR.

Este tiene la función de salir temporalmente al DOS o salir del editor en forma definitiva.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR_DE_TEXTO_ORIENTADO_A_COBOL FECHA SEP/93 DISEÑO ROSA_CRISTINA_LEON_Y_FREDYS_ARTURO_GARCIA_PORTILLO MODULO No. FCS00 NOMBRE DEL MODULO SALIR	
PROCEDIMIENTOS INCLUIDOS	
1. salir_al_DOS( ) 2. salir( )	

3.1.6 DOCUMENTACION DE PROCEDIMIENTOS.

3.1.6.1 Análisis Preliminar.

El FCEDIT cuenta con un sistema de menús que permiten al usuario poder ejecutar fácilmente cualesquiera de sus funciones, además de poseer la opción de presionar la secuencia de teclas respectiva desde su ventana de edición.

En vista de lo anterior se hace necesario crear dos procedimientos (o funciones) que lleven a cabo la ejecución de cualquier función, ya sea desde la ventana de edición o desde el menú. Se identifican estas funciones con los nombres de ***cursor()*** *y* ***menu_instantaneo()*** *respectivamente.*

La función ***cursor()*** *tiene como propósito capturar, procesar y ejecutar cualquier orden del usuario desde la ventana del editor; por su parte la función* ***menu_instantaneo()*** *tiene similar propósito con la única diferencia que su gestión es a nivel de menús.*

En seguida se describe con más detalles el propósito de al menos una de estas dos funciones.

Antes de iniciar con la documentación, es necesario, para mayor claridad funcional de los procedimientos, hacer la siguiente clasificación de funciones:

- *Funciones de propósito especial.*
- *Funciones propias del FCEDIT.*
- *Funciones predefinidas en C.*

*Las **funciones de propósito especial** constituyen un conjunto de funciones que son creadas para cumplir con una tarea especifica en el funcionamiento del FCEDIT. La **tabla-3.1** muestra un listado tentativo de estas funciones, con sus respectivos nombres y una breve descripción de lo que hace cada función.*

Tabla-3.1 Listado de funciones de propósito especial.

FUNCION	*PROPOSITO*
menu_principal()	*.Chequea las dimensiones del menú a desplegar, desvuelve un mensaje de error de rango si el menú no cabe en pantalla.* *.Invoca la función* ***modo_video()*** *para chequear el modo de video del PC e inicializa la dirección de memoria RAM de video.* *.Asigna suficiente memoria para el BUFFER de video.* *.Salva los datos de la actual pantalla.* *.Invoca la función* ***dibuja_borde()*** *para dibujar el borde del menú principal.* *.Invoca función* ***visualiza_principal()*** *para desplegar el menú principal.* *.Invoca la función* ***obtiene_seleccion()*** *para capturar la orden del usuario en el menú principal.* *.Restaura la pantalla original.* *.Libera memoria utilizada por el BUFFER de video.*
modo_video()	*.Devuelve el modo de video actual.*
visualiza_ principal()	*.Despliega el menú principal en su posición.*
obtiene_seleccion()	*.Elegir cualquier opción del menú principal, ya sea escribiendo la letra clave o haciendo uso de las flechas cursoras y la tecla ENTER.* *.Invoca la función* ***vete_xy()*** *para posicionar el cursor.* *.Invoca la función* ***escribe_cadena()*** *para desplegar cada item con el atributo correspondiente, en la posición correcta.* *.Invoca la función* ***menu_instantaneo()*** *para desplegar el menú abatible (pull-down menú) correspondiente a la elección del usuario en el menú principal.* *.Invoca la función* ***dibuja_borde()*** *para redibujar el borde del menú principal.* *.Si se presionan teclas que no corresponden a letras claves, a la tecla ENTER, o las flechas de cursor, la orden es denegada.*

...CONTINUACION

FUNCION	PROPOSITO
menu_instantaneo()	.Chequea las dimensiones del menú abatible a desplegar, devuelve un mensaje de error de rango si el menú no cabe en pantalla. .Asigna suficiente memoria para el BUFFER de video. .Salva los datos de la porción de pantalla en la que se despliega el menú abatible. .Invoca la función **dibuja_borde()** para dibujar el borde del menú abatible. .Invoca la función **obtiene_respuesta()** para capturar la orden del usuario en el menú abatible. .Restaura porción de pantalla. .Libera memoria utilizada por el BUFFER de video. .Si en el menú abatible actual, se presiona flecha de cursor izquierda, invoca la función **que_ventana75()** para desplegar el menú abatible correspondiente a la opción que se encuentra a la izquierda de la opción actual en el menú principal. .Si en el menú abatible actual, se presiona la flecha de cursor derecha, invoca la función **que_ventana77()** para desplegar el menú abatible correspondiente a la opción que se encuentra en la derecha de la opción actual en el menú principal.
visualiza_menus()	.Despliega menú abatible en su posición.
obtiene_respuesta()	.Elegir cualquier función del menú abatible, ya sea escribiendo la letra clave o haciendo uso de las flechas cursoras y la tecla ENTER. .Invoca la función **vete_xy()** para posicionar el cursor. .Invoca función **escribe_cadena()** para desplegar cada item con el atributo correspondiente en la posición correcta. .Después de capturar la orden del usuario, restaura porción de pantalla y libera memoria utilizada por el BUFFER de video.

... CONTINUACION

FUNCION	PROPOSITO
obtiene_respuesta( )	.Invoca la función correspondiente a la orden del usuario. Por ejemplo, supongamos que la elección en el menú principal fue la opción ARCHIVO, se despliega el menú abatible, y que dentro de este se elige la función imprimir, entonces la función que se invoca es **imprimir()** para mandar a impresión el archivo cargado en el entorno actual o denegar la orden en caso de que no exista archivo alguno. .Si el usuario presiona la flecha de cursor izquierda, esta función devuelve el valor 77 y el valor 75 en caso de que la tecla presionada sea la flecha de cursor derecha. .Si se presionan teclas que no correspondan a letras claves, a la tecla ENTER, o a la tecla ESC que desactive el menú abatible, la orden es denegada.
cursor()	.Captura, procesa y ejecuta cualquier orden del usuario; permitiendo con ello la edición de texto, la ejecución de cualquier función del FCEDIT haciendo uso de la secuencia de teclas respectivas, ejerce control total sobre los movimientos del cursor en la pantalla. .Invoca la función correspondiente a la orden del usuario.
posicion_cursor()	.Despliega la posición del cursor en la línea de estado del FCEDIT, esta se indica en líneas y columnas. .Si se ha especificado nombre para el archivo cargado en el entorno actual, este aparecerá en la línea de estado. .Cuando la función Activar/des del módulo COBOL esté en modo ON, **posición_cursor()** desplegará además de la posición del cursor, el nombre de archivo si existe, un rótulo que indica que las funciones para editar, compilar y ejecutar programas en COBOL están habilitados.

... CONTINUACION

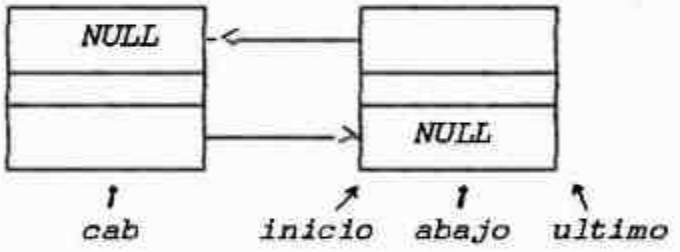
FUNCION	PROPOSITO
display_lista()	.Es invocada desde la función **cursor()** en el instante que se presiona la tecla PgUp o PgDn, por tanto su propósito es desplegar páginas de texto.
reserva_memoria_ primer_nodo()	.Asigna memoria para la creación del nodo cabecera y el primer nodo de LISTA, estas asignaciones se hacen con la función predefinida del C de la siguiente manera: cab=(editor *) farmalloc(sizeof(editor)); para el nodo cabecera. inicio=(editor *) farmalloc(sizeof(editor)); para el primer nodo. .Si el puntero devuelto por farmalloc apunta a una dirección de memoria nula, se despliega un mensaje que indica al usuario que no hay memoria libre para asignar.
reserva_memoria()	.Su papel fundamental es asignar memoria en forma dinámica en la medida que varía LISTA, esto se hace con la siguiente asignación: nuevo=(editor *) farmalloc(sizeof(editor)); .Por default el FCEDIT tiene modo de edición INSERT, esto significa que en el instante de editar o cargar un archivo debe determinarse la posición en que se va a efectuar la inserción, esta tarea la realiza la función **reserva_memoria()**.

ILUSTRACION LOGICA DE LAS FUNCIONES

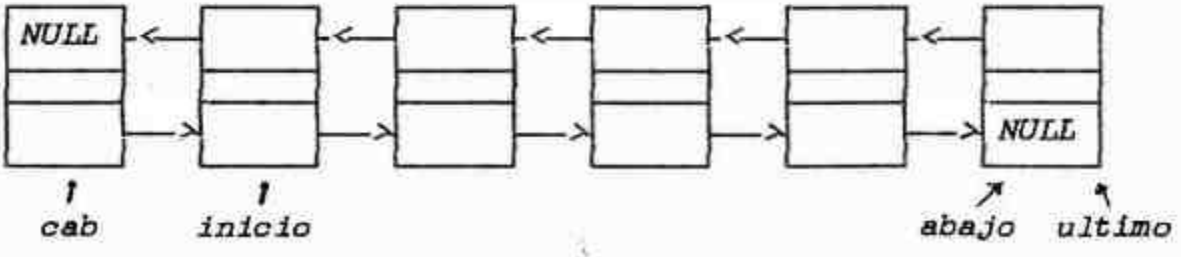
reserva_memoria_primer_nodo( ) y reserva_memoria().

Supongamos que se carga un archivo. Se ha dicho que el FCEDIT maneja una lista doblemente enlazada con cabecera en la cual se controlan todas las operaciones del editor. En el momento de cargar un archivo se hacen asignaciones de memoria en forma dinámica para ir leyendo el archivo y almacenando cada linea de texto en un nodo de lista.

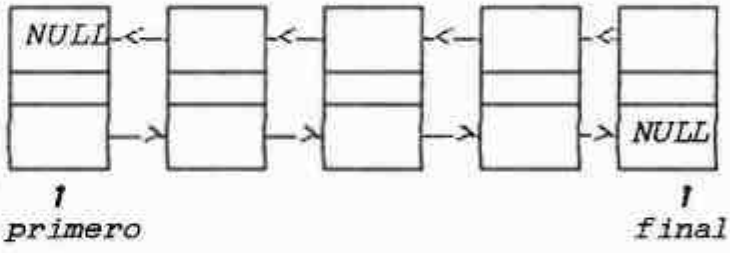
El estado de LISTA después de una llamada a la función ***reserva_memoria_primer_nodo()*** *es:*



Si el archivo que se carga tienen lineas de texto, se hacen n-1 llamadas a la función ***reserva_memoria()****, asi, si n=5 el estado final de LISTA luego de haber leido el archivo es:*



... CONTINUACION

FUNCION	PROPOSITO
reserva_memoria_ primer_nodo_insert() reserva_memoria_ nodo_insert()	.Estas dos funciones asignan memoria en forma dinámica para crear una lista doblemente enlazada que contiene información de un archivo que se ha de insertar en el texto actualmente en memoria, o ya sea que se trate de una operación de bloque; **reserva_memoria_primer_nodo()** crea el primer nodo de esta lista y **reserva_memoria_nodo_insert()** se ejecuta n-1 veces (n número de líneas de texto del archivo). Esta lista la manejan los punteros primero y final. Suponiendo que el archivo a insertar o un copiar, mover o borrar bloque tiene cinco líneas, esta lista se ilustraría de la siguiente manera:  luego la función **insert()** del módulo ARCHIVO se encarga de determinar en que parte de LISTA debe de hacerse el insertado.
ayuda_del_fcedit()	.Invoca la función **despliega_ayuda()**, esta se encarga de desplegar los menús de ayuda del FCEDIT.
despliega_ayuda()	.Declara un array de tipo int, en donde cada componente representa el tamaño en byte de la memoria necesaria para desplegar cada menú de ayuda del FCEDIT. .Chequea las dimensiones del menú a desplegar, devuelve un mensaje de error de rango si el menú no cabe en pantalla.

... CONTINUACION

FUNCION	PROPOSITO
despliega_ayuda()	.Asigna suficiente memoria para el BUFFER de video. .Salva los datos de la porción de pantalla en donde se hará el despliegue. .Invoca la función **dibuja_borde()** para dibujar el borde del menú de ayuda. .Invoca la función **visualiza_ayuda()** para desplegar la ayuda.
visualiza_ayuda()	.Invoca función **escribe_cadena()** para desplegar cada item del menu de ayuda con el atributo correspondiente, en la posición correcta.
escribe_cadena()	.Escribe una cadena de caracteres con atributo especificado.
escribe_char()	.Escribe un caracter con atributo especificado.
cls()	.Limpia la pantalla o parte de ella, dependiendo de las especificaciones del usuario.
vete_xy()	.Posiciona el cursor en la posición de pantalla especificada por el usuario.
escroll_d()	.Permite el escroll hacia abajo.
escroll_a()	.Permite el escroll hacia arriba.
dibuja_borde()	.Dibuja un borde (recuadro) a doble línea que encierra los items de un menú, la posición donde se dibuja tal borde es especificada por el usuario proporcionando las coordenadas de la esquina superior izquierda y las coordenadas de la esquina inferior derecha en valores enteros.
que_ventana77()	.Chequea en que menú abatible se ha presionado la tecla de cursor izquierda (flecha izquierda) para presentar el correspondiente menú abatible. Por ejemplo, si está activa la ventana de opción ESPECIAL del menú principal y se presiona la tecla antes mencionada, se presenta el menu abatible correspondiente a la opción COBOL.

... CONTINUACION

FUNCION	PROPOSITO
que_ventana77()	.Invoca función **escribe_cadena()** para desplegar cada items con el atributo correspondiente, en la posición correcta. .Invoca la función **dibuja_borde()** para dibujar el borde del menú a desplegar. .Invoca la función **menu_instantaneo()** para desplegar el menú abatible correspondiente a la elección del usuario.
que_ventana75()	.El papel de esta función es similar al de la función **que_ventana77()**, con la diferencia de que su ejecución depende de si en el menú abatible activo se presiona la tecla de cursor izquierda (flecha izquierda); para presentar la ventana menú correspondiente. Por ejemplo si esta activa la ventana de la opción SALIR y se presiona la tecla mencionada, se presenta el menú abatible correspondiente a la opción COBOL.
palabras_ reservadas()	.Invoca función **despliega()**, para desplegar una ventana en la esquina superior derecha de la pantalla que contiene las palabras reservadas COBOL. Estas palabras reservadas se encuentran en el archivo **palabras. cbl**, que el FCEDIT carga cuando se presiona la tecla de función F5 o se ejecuta la función **reservada_p()** de la opción COBOL en el menú principal. Con el objetivo de que el usuario pueda manipular este conjunto de palabras reservadas, el archivo palabras.cbl se carga en memoria en una lista doblemente enlazada la cual es controlada por tres punteros: fist, last, data. -first: puntero al primer nodo. -last: puntero al último nodo. -data: puntero que lee la lista en cualquier dirección. .Invoca la función **escribe_cadena()** para escribir items con atributo especificado, en la posición dada por el usuario.

... CONTINUACION

FUNCION	PROPOSITO
palabras_ reservadas()	.Controla cualquier orden del usuario en la ventana de palabras reservadas, por ejemplo, las teclas de cursor: flecha arriba y flecha abajo, la tecla ENTER con la cual se puede trasladar la palabra marcada, al archivo de trabajo, en la posición del cursor .Para desactivar esta ventana presionar la tecla ESC.
despliega()	.Invoca las funciones **memoria_primer_ nodo()** y **memoria()**, que asignan memoria en forma dinámica para crear la lista que almacena el archivo palabras.cbl. .Muestra el listado de palabras reservadas en COBOL en la zona de pantalla antes descrita.
memoria_primer_ nodo()	.Asigna memoria para la creación del primer nodo de la lista que almacena el archivo palabras.cbl si la asignación no es un éxito, se despliega un mensaje que indica al usuario que no hay memoria libre.
memoria()	.Asigna memoria en forma dinámica para crear la lista que almacena el archivo palabras.cbl. Por ejemplo si el archivo tienen 15 líneas, memoria se ejecuta 14 veces puesto que **memoria_ primer_nodo()** se encarga de crear el primer nodo, es decir leer la primera linea del archivo.
muestra_lista()	.Se invoca desde la función **palabras_ reservadas()** cuando la orden del usuario es Ctrl+PgUp o Ctrl+PgDn para ir al inicio o al final del archivo respectivamente. Por tanto **muestra_lista()** se encarga de mostrar la primera o última página del archivo palabras.cbl, según sea el caso.
plant_linea()	.Traza una línea horizontal en la tercera línea de la pantalla, separando la ventana del editor y la plantilla de números que facilitan la indentación de código fuente en COBOL.

... CONTINUACION

FUNCION	PROPOSITO
rotulo_salida()	.Muestra una ventana en el centro de la pantalla que dice "Desea guardar el archivo (S/N)" en el caso que se ejecute la función leer o la función salir y existe archivo en memoria. Si la respuesta es ESC, se desactiva la ventana y no ocurren cambios; si respuesta es S, **rotulo_salida()** solicita el nombre con el que se ha de guardar el archivo y procede a ejecutar la función original; si la respuesta es N, se desactiva la ventana y se procede a ejecutar la función original.
errorcreandoarch()	.Muestra una ventana en el centro de la pantalla con un mensaje de error si no tuvo éxito la ejecución de la función **leer()**. Para desactivar esta ventana presionar ESC.
errorguardandoarch()	.Muestra una ventana en el centro de la pantalla con un mensaje de error si no tuvo éxito la ejecución de la función **escribir()**. Para desactivar esta ventana presionar ESC.
errorinsertando arch()	.Muestra una ventana en el centro de la pantalla con un mensaje de error si no tuvo éxito la ejecución de la función **insertar()**. Para desactivar esta ventana presionar ESC.
errorignorar cambios()	.Muestra una ventana en el centro de la pantalla con un mensaje de error si no tuvo éxito la ejecución de la función **ignorar_c()**. Para desactivar esta ventana presionar ESC.
errorcargando estruct	.Muestra una ventana en el centro de la pantalla con un mensaje de error si el archivo que contiene la estructura general de un programa COBOL no puede ser leido.
errorcargando palabras()	.Muestra una ventana en el centro de la pantalla con un mensaje de error, si el archivo que contiene las palabras reservadas COBOL no puede ser leido.

...CONTINUACION

FUNCION	PROPOSITO
leernombrearch()	.Permite escribir el nombre del archivo a procesar. .Chequea la validez del nombre.
elimina_tab()	.Su papel fundamental es eliminar el caracter de tabulación en el momento que se carga un archivo de disco en la memoria del ordenador.

*Las **funciones propias del FCKDIT** constituyen el conjunto de funciones que integran ésta herramienta de software, los cuales ya han sido definidas e identificadas y que posteriormente se documentan con mayor detalle.*

*Las **funciones predefinidas en C,** se constituyen en un conjunto de funciones auxiliares que facilitan en gran medida la tarea de programación en el manejo de memoria, de cadenas, de archivos y otros. Un listado de las funciones predefinidas que se ocuparon en la implementación física del FCEDIT se detallan en el anexo #2.*

3.1.6.2 Consideraciones tomadas para la documentación de procedimientos y para fases posteriores.

Teniendo en cuenta las limitantes de tiempo que se tuvieron para el diseño del FCEDIT, se hizo necesario limitar la implementación de algunas funciones a las limitantes antes previstas. Por lo anterior y para conocimiento del usuario, escribiremos cuando sea el caso: "Procedimiento por implementar", tanto en el documento escrito como también en la AYUDA del FCEDIT, que no fue posible su implementación.

*Se ha destacado la importancia de dos procedimientos del conjunto de funciones de propósito especial, estas son **cursor()** y **menu_instantaneo()**, pero por tener similar propósito dentro del funcionamiento del FCEDIT se documenta únicamente la función **cursor()**, con la cual se inicia la*

documentación de procedimientos.

3.1.6.3 Procedimiento CURSOR.

Este procedimiento es uno de los más importantes en el conjunto de funciones de propósito especial. Realiza la tarea de procesar cualquier entrada desde el teclado. Para lograr esto, **cursor()** chequea en cada entrada, si se trata de una tecla normal o especial, en base al código que se asocia con cada tecla del dispositivo de entrada estándar. Esto nos conduce al uso de una variable que capture la tecla presionada y facilite chequear cuando es un código de caracter ASCII (tecla normal) y cuando es un código extendido (tecla especial). Es con este objetivo que se utiliza la siguiente declaración, la cual obedece a la sintaxis del C.

```
union inkey{
            char ch[2];
            int i;
          } c;
```

En esta declaración **union** es un tipo estructurado de datos en C, que contiene dos objetos de diferente tipo (ch,i), denominados miembros, en una misma zona de memoria. **inkey** es un identificador que nombra el nuevo tipo definido. **ch** es un arreglo de caracteres de longitud 2. **i** es una variable de tipo entero.

La digitación teclas es controlada a través de la función predefinida en C **bioskey**, de la siguiente manera:

```
while(!bioskey(1));
```

esta instrucción no permite avanzar mientras no se da una orden. Después de presionar una tecla, su código es almacenado en **c.i** con la siguiente instrucción:

```
c.i=bioskey(0);
```

teniendo esto simplemente se escribe una instrucción condicional, tal como se muestra, para determinar si se trata de una tecla normal o especial, con lo cual **cursor()** inicia un proceso mediante la sentencia **switch**, para determinar la función o procedimiento a ejecutar en respuesta a la orden del usuario.

```
if(c.ch[0]) // tecla normal
    switch(c.ch[0]){
                    . . .
                   }
else          // tecla especial
    switch(c.ch[1]){
                    . . .
                   }
```

Se aclara que las especificaciones de los objetos utilizados en este procedimiento son detallados en los módulos correspondientes que se documentan posteriormente.

La salida (o resultados) que genera este procedimiento está en correspondencia con la orden del usuario, por lo tanto no se especifica.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR_DE_TEXTO_ORIENTADO_A_COBOL FECHA SEP/93	
DISEÑO ROSA_CRISTINA_LEON_y_EREDYS_ARTURO_GARCIA_PORTILLO	
PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()	

ENTRADA	PROCESO	SALIDA

Orden del usuario	1.for(;;){
	//espera comando
	while(!bioskey(1));
	c.i=bioskey(0);
	if(c.ch[0]) //tecla normal
	switch(c.ch[0]){
	case 32://barra espaciadora
	//inserta caracter blanco
	if(((strlen(abajo->texto)+1)>79)
	|| ay>79) break;
	if(strlen(abajo->texto)+1)>ycur)
	{
	desplazar una posición más la
	cadena de caracteres a la
	derecha de ycur;
	escribir caracter blanco en la
	posición ycur;
	}
	else{
	escribir caracter blanco en la
	posición ycur;
	desplazar caracter de fin de
	línea a la posición ycur+1;
	}
	printf("%s",abajo->texto);
	++ycur;++ay; posición cursor;
	break;
	case 8 ://tecla de retroceso
	/*borrar caracter a la izquierda*/
	bksp();
	break;
	case 13://tecla de ENTER
	/*chequea si cb es 1 copia
	bloque*/
	if(cb){
	insertar_bloque();
	cb=0; ycur=0; ay=1;
	posicionar cursor;
	break;
	}
	//chequea si mb es 1 mueve bloque
		La salida de este procedi_ miento depende de la orden del usuario.

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()		
ENTRADA	PROCESO	SALIDA

	if(mb){
	determinar el punto de enlace
	del bloque de texto en LISTA;
	insertar_bloque;
	mb=0; posicionar cursor;
	break;
	}
	/*inserta nueva línea desde la
	posición del cursor*/
	inserta_l();
	break;
	case 9://tecla de función TAB
	/*mueve el cursor 9 espacios o a
	las columnas 8, 12, 72 si COBOL
	está activo*/
	tabulador();
	break;
	defaut:
	//tecla de caracteres imprimibles
	//escribe caracter
	ins_car();
	break;
	}//fin del switch(tecla pulsada)
	else //tecla pulsada es especial
	switch(c.ch(1)){
	case 77:
	/*flecha de cursor derecha, mueve
	el cursor una posición a la
	derecha*/
	car_der();
	break;
	case 75:
	/*flecha de cursor izquierda,
	mueve el cursor una posición a
	la izquierda*/
	car_izq();
	break;
	case 72:
	/*flecha de cursor hacia arriba,
	mueve el cursor una linea hacia
	arriba*/
	flecha_a();
	break;
	case 80:
	/*flecha de cursor hacia abajo, mueve
	cursor una línea hacia abajo*/	

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()		
ENTRADA	PROCESO	SALIDA

	flecha_ab();
	break;
	case 83:
	/*tecla DELETE, borra caracter a la derecha*/
	delete();
	break;
	case 73:
	/*tecla PgUp, escroll página hacia arriba*/
	pgup();
	break;
	case 81:
	/*tecla PgDn, scroll página hacia abajo*/
	pgdn();
	break;
	case -124:
	/*Ctrl+PgUp, ir al inicio de archivo*/
	inicio_ar();
	break;
	case 118:
	/*Ctrl+PgDn, ir al final del archivo*/
	final_ar();
	break;
	case 71:
	/*tecla HOME, ir al inicio de línea*/
	inicio_lin();
	break;
	case 79:
	/*tecla de END, ir al final de línea*/
	final_lin();
	break;
	case 119:
	/*Ctrl+HOME, borra hasta el inicio de línea*/
	b_inic_l();
	break;
	case 117:
	/*Ctrl+END, borra hasta el final de línea*/
	b_final_l();	

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()		
ENTRADA	PROCESO	SALIDA

	break;
	case 21: //Alt+Y, borra línea
	borrar_l();
	break;
	//Funciones de ARCHIVO
	case 38:
	/*Alt+L, lee archivo de disco a la memoria*/
	chequear el BUFFER de video;
	cprintf("Documento a recuperar:");
	/*lee y valida el nombre del archivo*/
	leenombrearch(0);
	/*si no existe lo crea*/
	abrir el archivo para lectura;
	/*lee el archivo y lo despliega en pantalla*/
	leer();
	xcur=3; ycur=0; ax=ay=1;
	posicionar cursor;
	break;
	case 18:
	/*Alt+E, guarda el archivo en disco*/
	chequear el BUFFER de video;
	cprintf("Documento a archivar:");
	/*despliega nombre actual y permite modificaciones*/
	leenombrearch(1);
	abrir el archivo para escritura;
	//escribe archivo en disco
	escribir();
	posicionar cursor;
	break;
	case 23:
	/*Alt+I, lee archivo de disco y lo inserta en el archivo en la posición del cursor*/
	chequear el BUFFER de video;
	cprintf("Documento a insertar:");
	/*lee y valida el nombre del archivo*/
	leenombrearch(0);	

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSORO.		
ENTRADA	PROCESO	SALIDA

	abrir el archivo para lectura;
	/*lee el archivo y lo despliega
	en pantalla*/
	leer();
	ycur=0; ay=1;
	posicionar cursor;
	break;
	case 25:
	//Alt+P, imprime el archivo
	chequear el BUFFER de video;
	//escribe en papel
	imprimir();
	break;
	case 34:
	/*Alt+G, lee nuevamente el
	archivo e ignora las
	modificaciones hechas*/
	chequear el BUFFER de video;
	abrir el archivo para lectura;
	leer();
	xcur=3; ycur=0; ax=ay=1;
	posicionar cursor;
	break;
	//Funciones de BLOQUE
	case 50://Alt+M, marcar bloque
	chequear el BUFFER de video;
	/*comprende marcar, copiar,
	mover y borrar*/
	operaciones_bloque();
	ycur=0; ay=1;
	posicionar cursor;
	break;
	case 31:
	/*Alt+S, restaurar bloque
	borrado en posición actual*/
	/*chequear si existe bloque
	borrado*/
	if(!bb) break;
	insertar_bloque();
	bb=ycur=0; ay=1;
	posicionar cursor;
	break;
	/*Funciones de VENTANA POR
	IMPLEMENTAR*/
	/*Funciones ESPECIAL POR
	IMPLEMENTAR*/	

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()		
ENTRADA	PROCESO	SALIDA

	//Funciones de COBOL
	case 60:
	/*tecla de función F2, Activar/ desactivar*/
	activar/des();
	break;
	case 61:
	/*tecla de función F3, compilar archivo*/
	compilar();
	break;
	case 62:
	/*tecla de función F4, ejecución del archivo COBOL*/
	ejecución();
	break;
	case 63:
	/*tecla de función F5, palabras reservadas*/
	reserva_p();
	break;
	case 64:
	//tecla de función F6, plantilla
	plantilla();
	break;
	case 65://F7,despliegue variables
	//POR IMPLEMENTAR
	break;
	case 133:
	/*F11, despliega nombres de párrafos*/
	//POR IMPLEMENTAR
	break;
	//Funciones para salir al DOS
	case 66:
	/*tecla de función F8, salir temporalmente del FCEDIT*/
	saldos();
	break;
	case 67:
	/*tecla de función f9, salir del FCEDIT*/
	salir();
	break;	

... CONTINUACION

PROCEDIMIENTO No. 00000 NOMBRE DEL PROCEDIMIENTO CURSOR()		
ENTRADA	PROCEDIMIENTO	SALIDA

	//Función de AYUDA
	case 59:
	/*tecla F1, despliega ayuda del FCEDIT*/
	ayuda_del_fcedit();
	break;
	case 68:
	/*tecla F10, accesar menú principal*/
	ir al menú principal;
	break;
	}//fin switch(tecla especial)
	posicionar cursor;
	}//fin for(;;)
	2. fin de proceso;	

3.1.6.4 PROCEDIMIENTOS DEL MODULO ARCHIVO.

Procedimiento LEER.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION	
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. ECA01 NOMBRE DEL PROCEDIMIENTO LEER( )		
ENTRADA	PROCESO	SALIDA

	1. a=linea=1;
	2. /*utilizada por ediciones previas.*/
	Liberar memoria;
	3. //lista vacía
	cab=inicio=NULL;
	abajo=ultimo=NULL;
	4. while(!feof(puntero)){
	if(!a){
	reserva memoria para más nodos;
	fgets(abajo->texto,80, puntero);
	}
	else{
	/*crea cabecera y primer nodo*/
	reserva memoria;
	fgets(abajo->texto,80, puntero);
	a=0;
	}
	}/*archivo leído de disco y vaciado en lista*/
	5. /*cerrar archivo apuntado por puntero*/
	fclose(puntero);
	/*mostrar archivo*/
	6. abajo=inicio;
	7. while(abajo && linea<22){
	printf("%s\n", abajo->texto);
	++linea;
	abajo=abajo->sig;
	}
	8. abajo=inicio;
	9. fin del proceso;
		Despliega en pantalla del archivo leído

... CONTINUACION

PROCEDIMIENTO No. FCA01 NOMBRE DEL PROCEDIMIENTO LEER( )			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	cab	editor	Almacenar la dirección del nodo cabecera de la lista.
	inicio	editor	Almacenar la dirección del primer nodo de la lista.
	ultimo	editor	Almacenar la dirección del último nodo de la lista.
	abajo	editor	Recorrer la lista en dos direcciones.
	puntero	FILE	Referenciar el archivo abierto para lectura.
VARIABLE LOCAL	linea	register	Contabilizar el número de líneas de la lista a ser desplegados en pantalla.
	a	register	Bandera que controla la reserva de memoria para cargar el archivo.
TIPOS DE DATO GLOBAL	editor	struct	Almacena LISTA.
	FILE	struct	Almacenar información acerca del archivo a leer.

Procedimiento ESCRIBIR.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	*ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION*
PROYECTO EDITOR DE TEXTO ORIENTADO A COBCL	*FECHA SEP/93*
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCA02	*NOMBRE DEL PROCEDIMIENTO ESCRIBIR()*

ENTRADA	*PROCESO*	*SALIDA*

	1. auxabajo=inicio;
	/*escribir lista en archivo referenciada por puntero*/
	2. while(auxabajo){
	fputs(auxabajo->texto, puntero):
	auxabajo=auxabajo->sig;
	}
	3. /*cerrar archivo apuntado por puntero*/
	fclose(puntero);
	4. auxabajo=NULL;
	5. fin del proceso;
		Archivo almacenado en la ruta especificada.

OBJETOS UTILIZADOS

OBJETO	*NOMBRE*	*TIPO*	*PROPOSITO*
VARIABLE GLOBAL	*inicio*	*editor*	*Almacenar la dirección del primer nodo de la lista.*
	puntero	*FILE*	*Referenciar el archivo abierto para escritura.*
VARIABLE LOCAL	*auxabajo*	*editor*	*Recorrer la lista de inicio a último escribiendo cada nodo en el archivo apuntado por puntero.*
TIPO DE DATO GLOBAL	*FILE*	*struct*	*Almacenar información acerca del archivo a escribir.*

Procedimiento INSERTAR.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP\93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCA03 NOMBRE DEL PROCEDIMIENTO INSERTAR( )	

ENTRADA	PROCESO	SALIDA

	1. auxxcur=xcur;
	2. auxycur=ycur;
	3. if(puntero){
	//crea primer nodo insertar
	reserva memoria;
	fgets(final->texto,80,
	puntero);
	}
	else interrumpir proceso;
	4. while(!feof(puntero)){
	reserva memoria para
	más nodos;
	fgets(final->texto,80,
	puntero);
	}
	/*insertando nueva lista en lista de trabajo*/
	5. if(abajo!=ultimo &&
	abajo!=inicio)
	if(ycur==0)
	insertar entre el nodo apuntado por abajo y el nodo apuntado por abajo->ant;
	else
	insertar entre el nodo apuntado por abajo y el nodo apuntado por abajo->sig;
	else
	if(abajo==ultimo)
	if(ycur==0)
	insertar entre el nodo apuntado por abajo y el nodo apuntado por abajo->ant;
	else
	insertar al final de la lista;
		Despliega la lista a partir de la posición donde se insertó el archivo.

... CONTINUACION

PROCEDIMIENTO No. FCAQ3 NOMBRE DEL PROCEDIMIENTO INSERTAR()		
ENTRADA	PROCESO	SALIDA

	else //abajo==inicio
	if(ycur==0)
	insertar al inicio
	de la lista;
	else
	insertar entre el
	nodo apuntado por
	abajo y el nodo
	apuntado por
	abajo->sig;
	6. abajo=primero;
	7. /*cerrar archivo apuntado
	por puntero*/
	fclose(puntero);
	//mostrar archivo insertado
	8. if(auxycur==0)
	limpiar pantalla a
	partir de xcur;
	else
	limpiar pantalla a
	partir de xcur + 1;
	9. while(abajo && xcur<=23){
	printf("%s\n",
	abajo->texto);
	abajo=abajo->sig;
	10. /*liberar punteros, lista
	vacia*/
	primero=final=NULL;
	11. actualizar valor de xcur
	y ycur;
	12. posicionar cursor;
	13. fin de proceso;	

... CONTINUACION

PROCEDIMIENTO No. ECA03 NOMBRE DEL PROCEDIMIENTO INSERTAR( ).			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	xcur	int	Almacenar posición fila del cursor.
	ycur	int	Almacenar posición columna del cursor.
	puntero	FILE	Referenciar el archivo abierto para lectura.
	primero	editor	Almacenar la dirección del primer nodo de la lista que contiene el archivo a insertar.
	final	editor	Almacenar la dirección del último nodo de la lista que contiene el archivo a insertar.
	inicio	editor	Almacenar la dirección del primer nodo de la lista.
	abajo	editor	Referenciar el nodo a partir del cual se enlaza la lista que contiene el archivo a insertar.
	ultimo	editor	Almacenar la dirección del último nodo de la lista.
VARIABLE LOCAL	auxxcur	register	Almacenar temporalmenta el valor de xcur.
	auxycur	register	Almacenar temporalmente el valor de ycur.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento IMPRIMIR.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP\93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. EC404	NOMBRE DEL PROCEDIMIENTO IMPRIMIR()

ENTRADA	PROCESO	SALIDA

	1. auxabajo=inicio;
	2. chequear impresora;
	3. whiie(auxabajo->sig){
	//imprimiendo
	fputs(auxabajo->texto,
	stdprn);
	auxabajo=auxabajo->sig;
	}
	4. //última línea
	fputs(auxabajo->texto,
	stdprn);
	5. auxabajo=NULL;
	6. fin del proceso
		Archivo impreso en papei.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	inicio	editor	Almacenar ia dirección dei primer nodo de ia lista.
VARIABLE LOCAL	auxabajo	editor	Recorrer la lista de inicio a último escribiendo cada nodo en ei dispositivo standard de saiida(PRINTER).
TIPOS DE DATO GLOBAL	editor	struct	Almacenar LISTA.
	FILE	struct	Aimacenar información acerca del archivo a ieer.

Procedimiento IGNORAR.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION	
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP\93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No.FCA05 NOMBRE DEL PROCEDIMIENTO IGNORAR C( )		
ENTRADA	PROCESO	SALIDA
	1. /*llamada al procedimiento leer()*/ 2. fin del proceso;	Despliega en pantalla el archivo.

3.1.6.5 PROCEDIMIENTOS DEL MODULO BLOQUE

Procedimiento MARCAR BLOQUE.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCB01	NOMBRE DEL PROCEDIMIENTO MARCAR( )

ENTRADA	PROCESO	SALIDA

	1. axp=axf=ax;
	2. xcurp=xcurf=xcur;
	3. auxprimero=auxfinal=abajo;
	4. cb=bb=F10B=0;
	5. sobreiluminar línea de texto en la posición actual del cursor;
	6. for(;;){
	//espera presionar tecla
	while(!bioskey(1)){
	capturar tecla pulsada;
	if(tecla pulsada normal)
	switch(tecla pulsada){
	case ESC:
	desmarcar bloque;
	auxprimero=NULL;
	auxfinal=NULL;
	fin de proceso;
	}
	else
	/*tecla presionada es especial*/
	switch(tecla pulsada){
	case 59: //tecla F1
	mostrar ayuda del FCEDIT;
	break;
	case 72: /*tecla flecha hacia arriba*/
	if(abajo==inicio)
	break;
	if(!cb && !bb){
	if((xcurp<xcur)
	if(xcur>3){
	escribir línea de texto en video normal;
		Bloque de texto especificado en video inverso o sobre fondo rojo, dependiendo de si el monitor es monocromático o a colores respectivamente.

... CONTINUACION

PROCEDIMIENTO No. ECB01 NOMBRE DEL PROCEDIMIENTO MARCAR()		
ENTRADA	PROCESO	SALIDA

	--xcur;--xcurf;
	}
	else{
	xcur=3;
	escrolar
	descendentemente
	21 líneas;
	escribir línea
	de texto en
	video normal;
	}
	else
	if(xcur>3){
	sobreiluminar
	línea de texto
	en la posición
	del cursor;
	--xcur; --xcurf;
	}
	else{
	xcur=3;
	escrolar
	descendentemente
	21 líneas;
	sobreiluminar
	línea de texto
	en la posición
	de1 cursor;
	if(xcurp<23)
	++xcurp;++axp;
	}
	auxfinal=abajo->ant;
	abajo=abajo->ant;
	--ax; --axf;
	}//fin if(!cb && !bb)
	else
	if(xcur>3){
	abajo=abajo->ant;
	--xcur; --ax;
	}
	else{
	xcur=3;
	escrolar descenden-
	temente 21 líneas;
	escribir línea de tex-
	to en video normal;	

... CONTINUACION

PROCEDIMIENTO No. ECB01. NOMBRE DEL PROCEDIMIENTO MARCAR().		
ENTRADA	PROCESO	SALIDA

	abajo=abajo->ant;
	--ax;
	}
	posicionar cursor;
	break;
	case 80:/*tecla flecha
	hacia abajo*/
	if(abajo==ultimo)
	break;
	if(!cb && !bb){
	if(xcur<xcurp)
	if(xcur>22){
	xcur=23;
	if(abajo==ultimo)
	break;
	escrolar
	ascendentemente 21
	líneas;
	escribir línea de
	texto en video
	normal;
	}
	else{
	if(abajo==ultimo)
	break;
	escribir línea de
	texto en video
	normal;
	++xcur; ++xcurf;
	}
	else
	if(xcur>22){
	xcur=23;
	if(abajo==ultimo)
	break;
	if(xcurp>3)
	--xcurp;
	escrolar
	ascendentemente 21
	líneas;
	sobreiluminar
	línea de texto en
	la posición del
	cursor;
	}	

... CONTINUACION

PROCEDIMIENTO No. FCB01 NOMBRE DEL PROCEDIMIENTO MARCAR()		
ENTRADA	PROCESO	SALIDA

	else{
	if(abajo==ultimo)
	break;
	sobreiluminar línea
	de texto;
	++xcur; ++xcurf;
	}
	auxfinal=abajo->sig;
	abajo=abajo->sig;
	}//fin if(!cb && !bb)
	else{
	if(xcur>22){
	xcur=23;
	if(abajo==ultimo)
	break;
	escrolar
	ascendentemente 21
	líneas;
	escribir línea de
	texto en video
	normal;
	}
	else{
	if(abajo==ultimo)
	break;
	++xcur;
	}
	abajo=abajo->sig;
	++ax;
	}
	posicionar cursor;
	break;
	case 24:/*Alt+O, copiar
	bloque*/
	copiar bloque;
	fin de proceso;
	case 47:/*Alt+V, mover
	bloque*/
	mover bloque;
	fin de proceso;
	case 19:/*Alt+R, borrar
	bloque*/
	borrar bloque;
	fin de proceso;	

... CONTINUACION

PROCEDIMIENTO No. FCB01. NOMBRE DEL PROCEDIMIENTO MARCAR( ).		
ENTRADA	PROCESO	SALIDA
	case 68:/*tecla F10 F10B=1; ir al menú principal del FCEDIT; }/*fin switch(tecla presionada)*/ }//fin for(:;) 3. fin de proceso;	

... CONTINUACION

PROCEDIMIENTO No. FCB01. NOMBRE DEL PROCEDIMIENTO MARCAR().			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	auxprimero	editor	Almacenar dirección del primer nodo de la lista marcada.
	auxfinal	editor	Almacenar dirección del último nodo de la lista marcada.
	abajo	editor	Referenciar el nodo de la lista a partir del cual se lleva a cabo la operación de marcar. Recorre la lista en las dos direcciones.
	xcur	int	Almacenar posición fila del cursor.
	xcurp,xcurf	int	Almacenar las posiciones inicial y final, respectivamente, de pantalla donde aparece marcado el bloque de texto.
	ax	int	Almacenar la posición final del cursor en la ventana del editor.
	axp,axf	int	Almacenar posiciones inicial y final respectivamente, en la ventana del editor, desde donde se empezó a marcar hasta el final.

... CONTINUACION

PROCEDIMIENTO No. FCB01 NOMBRE DEL PROCEDIMIENTO MARCAR( )			
OBJETO	NOMBRE	TIPO	PROPOSITO
	cb	int	Almacenar el valor 1 si la función copiar bloque está activada y el valor 0 si no lo está.
	bb	int	Almacenar el valor 1 si la función borrar bloque está activada y el valor 0 si no lo está.
	F10B	int	Almacenar el valor 1 si se ha pulsado la tecla F10 dentro de la función, marcar bloque.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento COPIAR BLOQUE.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. ECB02	NOMBRE DEL PROCEDIMIENTO COPIAR( )

ENTRADA	PROCESO	SALIDA

	1. tempauxprimero=auxprimero;
	tempauxfinal=auxfinal;
	2. desmarcar el bloque de texto a copiar;
	3. auxprimero=tempauxprimero;
	auxfinal=tempauxfinal;
	4. /*indica que la función copiar bloque está activada*/
	cb=1;
	5. liberar memoria utilizada por los punteros primero y final;
	6. /*para el primer nodo a copiar*/
	reservar memoria;
	7. if(xcurp<=xcur){
	strcpy(final->texto,
	auxprimero->texto);
	/*bloque a copiar 1 línea*/
	if(auxprimero==auxfinal){
	desplazar el cursor a la posición donde se va a copiar el bloque;
	insertar lista que contiene el bloque marcado en el archivo cargado en el entorno actual;
		Copia el bloque marcado, en la posición del cursor.

... CONTINUACION

PROCEDIMIENTO No. FCB02 NOMBRE DEL PROCEDIMIENTO COPIAR( )		
ENTRADA	PROCESO	SALIDA

	limpiar la pantalla a
	partir de la posición
	actual del cursor;
	desplegar el archivo a
	partir de la posición
	actual del cursor;
	fin del proceso;
	}
	do{
	auxprimero=
	auxprimero->sig;
	reserva memoria para
	más nodos;
	strcpy(final->texto,
	auxprimero->texto);
	}while(auxprimero!=
	auxfinal);
	}
	8. else{
	strcpy(final->texto,
	auxfinal->texto);
	do{
	auxfinal=auxfinal->sig;
	reservar memoria para
	más nodos;
	strcpy(final->texto,
	auxfinal->texto);
	}while(auxfinal!=
	auxprimero);
	9. auxprimero=auxfinal=NULL;
	10. fin de proceso;	

... CONTINUACION

PROCEDIMIENTO No. FCB02 NOMBRE DEL PROCEDIMIENTO COPIAR()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	auxprimero	editor	Almacenar la dirección del primer nodo a copiar.
	auxfinal	editor	Almacenar la dirección del último nodo a copiar.
	primero, final	editor	Almacenar las direcciones de inicio y final, respectivamente de la lista que contiene el bloque de texto a copiar.
	cb	int	Almacenar el valor 1 si la función copiar bloque está activada y el valor 0 si no lo está.
	xcur	int	Almacenar la posición fila del cursor.
	xcurp	int	Almacenar la posición de la primera fila del bloque marcado.
VARIABLE LOCAL	tempaux-primero, tempaux-final	editor	Almacenar temporalmente las direcciones de inicio y de final del bloque marcado.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento MOVER BLOQUE.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION	
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCB03 NOMBRE DEL PROCEDIMIENTO MOVER( )		
ENTRADA	PROCESO	SALIDA

	1. tempauxprimero=auxprimero;
	tempauxfinal=auxfinal;
	2. desmarcar el bloque de texto a mover;
	3. auxprimero=tempauxprimero;
	auxfinal=tempauxfinal;
	4. /*indica que la función mover bloque está activada*/
	mb=1;
	5. liberar memoria utilizada por los punteros primero y final;
	6. if(xcurp<=xcur){
	primero=auxprimero;
	final=auxfinal;
	}
	else{
	primero=auxfinal;
	final=auxprimero;
	xcurp=xcurf; axp=axf;
	}
	7. desplazar el cursor a la posición donde se va a mover el bloque;
	8. descenlazar los nodos comprendidos entre primero y final de la lista;
	9. insertar lista(primero-final) que contiene el bloque marcado en el archivo cargado en el entorno actual;
	10. desplegar el archivo a partir de la posición actual del cursor;
	11. fin de proceso;
		Mueve el bloque marcado, a la posición actual del cursor.

... CONTINUACION

PROCEDIMIENTO No. ECB03 NOMBRE DEL PROCEDIMIENTO MOVER()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	auxprimero, auxfinal	editor	Almacenar direcciones inicio y final respectivamente, del bloque a mover.
	primero, final	editor	almacenar las direcciones de inicio y final respectivamente, de la lista que contiene el bloque de texto a mover.
	mb	int	Almacenar el valor 1 si la función mover bloque está activada y el valor 0 si no lo está.
	xcur	int	Almacenar posición fila del cursor.
	xcurp,xcurf	int	Almacenar la primera y última fila de pantalla del bloque marcado.
	axp,axf	int	Almacenar las posiciones inicial y final respectivamente, en la ventana del editor, del bloque a mover.
VARIABLE LOCAL	tempaux-primero, tempaux-final	editor	Almacenar temporalmente las direcciones de inicio y de final del bloque marcado.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR BLOQUE.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCB04	NOMBRE DEL PROCEDIMIENTO BORRAR()

ENTRADA	PROCESO	SALIDA

	1. /*indica que la función borrar bloque está activada*/
	bb=1;
	2. liberar memoria utilizada por los punteros primero y final;
	3. if(xcurp<=xcur){
	primero=auxprimero;
	final=auxfinal;
	}
	else{
	primero=auxfinal;
	final=auxprimero;
	xcurp=xcurf;
	}
	4. descenlazar los nodos comprendidos entre primero y final de la lista que manejan los punteros cab, inicio, abajo y último;
	5. desplegar el archivo a partir de la posición actual del cursor;
	6. fin de proceso;
		Borra el bloque marcado (de pantalla y de LISTA).

... CONTINUACION

PROCEDIMIENTO No. FCB04 NOMBRE DEL PROCEDIMIENTO BORRAR( )			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	bb	int	Almacenar el valor 1 si la función borrar bloque está activada y el valor 0 si no lo está.
	xcur	int	Almacenar la posición fila del cursor.
	xcurp,xcurf	int	Almacenar la primera y última fila de pantalla del bloque a borrar.
	auxprimero, auxfinal	editor	Almacenar las direcciones de inicio y final respectivamente del bloque a borrar.
	primero, final	editor	Almacenar las direcciones de inicio y final respectivamente de la lista que contiene el bloque borrado.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento RESTAURAR BLOQUE.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.ECB05	NOMBRE DEL PROCEDIMIENTO RESTAURAR( )

ENTRADA	PROCESO	SALIDA
	1. /*bb=0 no hay nada que restaurar*/ if(!bb) fin de proceso; 2. /*posición actual del cursor*/ restaurar bloque borrado en la dirección apuntada por el puntero abajo; 3. desplegar el archivo a partir de la posición actual del cursor; 4. fin de proceso;	Restaura bloque borrado, en la posición actual del cursor.

... CONTINUACION

PROCEDIMIENTO No. FCB05 NOMBRE DEL PROCEDIMIENTO RESTAURAR( )			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	bb	int	Almacenar el valor 1 si se ha ejecutado la función borrar bloque y el valor 0 si no ha sido ejecutada.
	primero, final	editor	Almacenar las direcciones de inicio y final respectivamente, de la lista que contiene el bloque de texto restaurar.
	abajo	editor	Referenciar el nodo de lista en el posición fila(xcur) del cursor, en donde se ha de restaurar el bloque de texto.
	xcur	int	Almacenar la posición fila del cursor en el instante que se ejecuta la acción de restaurar bloque.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

3.1.6.6 PROCEDIMIENTOS DEL MODULO VENTANA.

Los procedimientos que forman este módulo son: dividir(), cambiar() y borrar(), fueron definidos con el propósito de manipular dos archivos en el entorno del FCEDIT pero por las razones antes expuestas (en 3.1.6.2) no fue posible su implementación.

3.1.6.7 PROCEDIMIENTOS DEL MODULO CURSOR.

Procedimiento CARACTER A LA IZQUIERDA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCCO2	NOMBRE DEL PROCEDIMIENTO CAR IZQ()

ENTRADA	PROCESO	SALIDA
	1./*posición actual del cursor (ax,1)*/ if(ay<2) interrumpir proceso; 2. --ycur; --ay; 3. posicionar cursor; 4. fin de proceso;	El cursor aparece una posición más a la izquierda.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ycur	int	Almacenar la posición columna del cursor en la pantalla.
	ay	int	Almacenar la posición columna del cursor en la ventana del editor.

Procedimiento CARACTER A LA DERECHA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC03	NOMBRE DEL PROCEDIMIENTO CAR_DER()

ENTRADA	PROCESO	SALIDA
	1./*posición actual del cursor (ax.80)*/ if(ay>79) interrumpir proceso; 2. ++ycur; ++ay; 3. posicionar cursor; 4. fin de proceso;	El cursor aparece una posición más a la derecha.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ycur	int	Almacenar la posición columna del cursor en la pantalla.
	ay	int	Almacenar la posición columna del cursor en la ventana del editor.

Procedimiento LINEA ARRIBA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. ECC04	NOMBRE DEL PROCEDIMIENTO FLECHA_A()

ENTRADA	PROCESO	SALIDA

	1. if(ax>1){
	if(xcur>3) --xcur;
	else{
	xcur=3;
	escrolar
	descendentemente 21
	líneas;
	if(abajo!=cab)
	printf("%s",
	abajo->ant->texto);
	}
	abajo=abajo->ant;
	--ax;
	posicionar cursor;
	}
	2. fin de proceso:
		El cursor aparece una posición más arriba o escrol hacia abajo mostrando línea de texto en primera fila de pantalla.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax	int	Almacenar la posición fila del cursor en la ventana del editor.
	xcur	int	Almacenar la posición fila del cursor en la pantalla.
	cab	editor	Almacenar la dirección del nodo cabecera de lista.
	abajo	editor	Recorrer LISTA.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento LINEA ABAJO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.FCC05	NOMBRE DEL PROCEDIMIENTO FLECHA AB().

ENTRADA	PROCESO	SALIDA

	1. if(ax>20 && xcur>22){
	if(abajo==ultimo)
	interrumpir proceso;
	escrolar
	ascendentemente 21
	lineas;
	printf("%s",
	abajo->sig->texto);
	}
	else{
	if(abajo==ultimo)
	interrumpir proceso;
	++xcur;
	}
	2. abajo=abajo->sig;
	3. ++ax;
	4. posicionar cursor;
	5. fin de proceso;
		El cursor aparece una posición más abajo o escrol hacia arriba mostrando línea de texto en última fila de pantalla.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax	int	Almacenar la posición fila del cursor en la ventana del editor.
	xcur	int	Almacenar la posición fila del cursor en la pantalla.
	abajo	editor	Recorrer LISTA.
	ultimo	editor	Almacenar la dirección del último nodo de LISTA.
TIPO DE DATO CLOBAL	editor	struct	Almacenar LISTA.

Procedimiento PALABRA IZQUIERDA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.		ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCC06 NOMBRE DEL PROCEDIMIENTO P_IZQ( )		
ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR*/	

Procedimiento PALABRA DERECHA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.		ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCC07 NOMBRE DEL PROCEDIMIENTO P_DER( )		
ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

Procedimiento INICIO DE LINEA

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.FCCOS NOMBRE DEL PROCEDIMIENTO INICIO_LIN()	

ENTRADA	PROCESO	SALIDA
	1. ycur=0; 2. ay=1; 3. posicionar cursor; 4. fin de proceso;	El cursor aparece en la posición (ax,1).

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax	int	Almacenar la posición fila del cursor en la ventana del editor.
	ay	int	Almacenar la posición columna del cursor en la ventana del editor.
	ycur	int	Almacenar la posición columna del cursor en la pantalla.

Procedimiento FINAL DE LINEA

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.FCCO9 NOMBRE DEL PROCEDIMIENTO FINAL_LIN( )	

ENTRADA	PROCESO	SALIDA
	1. ycur=0; ay=1; 2. incrementar ycur y ay hasta encontrar el caracter de fin de linea('\0'); 3. posicionar cursor; 4. fin de proceso;	El cursor aparece en la posición del caracter '\0' de la linea ax.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax	int	Almacenar la posición fila del cursor en la ventana del editor.
	ay	int	Almacenar la posición columna del cursor en la ventana del editor.
	ycur	int	Almacenar la posición columna del cursor en la pantalla.

Procedimiento INICIO DE ARCHIVO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION

PROYECTO EDITOR DE TEXTO ORIENTADO A OBJETOS FASE II

DISEÑO ROSA CRISTINA IRAHETA y FREDDY ARTURO GARCIA PORTILLO

PROCEDIMIENTO No. FOC10 NOMBRE DEL PROCEDIMIENTO INICIO_AR()

ENTRADA	PROCESO	SALIDA

	1. if(abajo==inicio)
	interrumpir proceso;
	2. if(ax<=21){
	abajo=inicio;
	xcur=3; ax=ay=1;
	ycur=0;
	else{
	desplegar los primeros
	21 nodos de lista;
	abajo=inicio;
	xcur=3; ax=ay=1;
	ycur=0;
	}
	3. posicionar cursor;
	4. fin de proceso;
		Despliega en pantalla la primera página de texto del archivo cargado en el entorno actual.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax,ay	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la ventana del editor.
	xcur,ycur	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la pantalla.
	inicio	editor	Almacenar la dirección del primer nodo de LISTA.
	abajo	editor	Recorrer LISTA.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento FINAL DE ARCHIVO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC11	NOMBRE DEL PROCEDIMIENTO FINAL AR()

ENTRADA	PROCESO	SALIDA

	1. if(abajo==ultimo)
	interrumpir proceso;
	2. while(abajo!=ultimo){
	abajo=abajo->sig;
	++ax;
	}
	3. if(ax<=21) xcur=ax+2;
	else
	desplegar los últimos
	21 nodos de lista;
	4. posicionar cursor;
	5. fin de proceso;
		Despliega en pantalla la última página de texto del archivo cargado en el entorno actual.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax	int	Almacenar la posición fila del cursor en la ventana del editor.
	xcur	int	Almacenar la posición fila del cursor en la pantalla.
	último	editor	Almacenar la dirección del último nodo de LISTA.
	abajo	editor	Recorrer LISTA.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento PAGINA ARRIBA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE C.C. NN. Y MAT.	ESCUELA DE MATEMATICA DEP. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL.	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC12 NOMBRE DEL PROCEDIMIENTO PaUp()	

ENTRADA	PROCESO	SALIDA

	1. if(abajo==inicio)
	interrumpir proceso;
	2. if(ax<=21){
	xcur=3; ycur=0;
	ax=ay=1;
	abajo=inicio;
	}
	else{
	if(xcur=23)
	decrementar abajo y ax 40 veces o hasta que abajo==inicio y ax==1;
	else
	if(xcur==23 &&
	abajo==ultimo)
	decrementar abajo y ax 20 veces o hasta que abajo==inicio y ax==1;
	else{
	auxxcur=20+xcur-3;
	decrementar abajo y ax auxxcur veces o hasta que abajo==inicio y ax==1;
	}
	xcur=3; ycur=0; ay=1;
	desplegar lista a partir del nodo apuntado por abajo;
	}
	3. posicionar cursor;
	4. fin de proceso;
		Despliega en pantalla página de texto anterior.

... CONTINUACION

PROCEDIMIENTO No. ECC12 NOMBRE DEL PROCEDIMIENTO FeUp()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax,ay	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la ventana del editor.
	xcur,ycur	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la pantalla.
	abajo	editor	Recorrer LISTA.
	inicio	editor	Almacenar la dirección del primer nodo de LISTA.
	ultimo	editor	Almacenar la dirección del último nodo de LISTA.
VARIABLE LOCAL	auxxcur	register	Almacenar entero positivo, que indica el numero de veces que debe decrementarse abajo y ax.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento PAGINA ABAJO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC13	NOMBRE DEL PROCEDIMIENTO PaDn()

ENTRADA	PROCESO	SALIDA

	1. if(abajo==ultimo)
	limpiar pantalla;
	xcur=3; ycur=0; ay=1;
	printf("%s",
	abajo->texto);
	posicionar cursor;
	fin de proceso:
	}
	2. if(x<23)
	incrementar abajo, xcur
	y ax hasta que
	abajo==ultimo y xcur<23;
	3. xcur=3; ycur=0; ay=1;
	4. desplegar lista a partir
	del nodo apuntado por
	abajo;
	5. posicionar cursor;
	6. fin de proceso;
		Despliega en pantalla página de texto siguiente.

... CONTINUACION

PROCEDIMIENTO No. FCC13 NOMBRE DEL PROCEDIMIENTO PgDn()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ax, ay	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la ventana del editor.
	xcur, ycur	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la pantella.
	abajo	editor	Recorrer LISTA.
	ultimo	editor	Almacenar la dirección del último nodo de LISTA.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento CENTRAR PAGINA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION	
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL FECHA SEP/93		
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCC14 NOMBRE DEL PROCEDIMIENTO CENTRA P()		
ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

3.1.6.8 PROCEDIMIENTOS DEL MODULO INSERTAR/BORRAR.

Procedimiento BORRAR CARACTER IZQUIERDO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A CORAL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.ECIRQ1 NOMBRE DEL PROCEDIMIENTO BKSP()	

ENTRADA	PROCESO	SALIDA

	1. if(ay<2)
	interrumpir el proceso;
	2. /*longitud de linea de texto*/
	if((strlen(
	abajo->texto)-1)>ycur){
	copiar en estring los caracteres desde ycur hasta el último caracter de la línea;
	abajo->texto[ycur-1]=
	'\0';
	strcat(abajo->texto,
	estring);
	}
	else{
	do{
	//decrementar
	--ycur; --ay;
	posicionar cursor;
	}while((strlen(
	abajo->texto)+1)<ycur);
	abajo->texto[ycur]=' ';
	abajo->texto[ycur-1]=
	'\0';
	}
	3. limpiar linea;
	4. printf("%s",
	abajo->texto);
	4. posicionar cursor;
	5. fin de proceso;
		Borra caracter a la izquierda de la posición del cursor.

... CONTINUACION

PROCEDIMIENTO No.FCI3Q1 NOMBRE DEL PROCEDIMIENTO BK_SP( )			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ay	int	Almacenar la posición columna del cursor en la ventana del editor.
	ycur	int	Almacenar la posiciones columna del cursor en la pantella.
	abajo	editor	Referenciar el nodo (linea de texto) en el que se borra el caracter.
VARIABLE LOCAL	estring	char	Almacenar temporalmente una cadena de caracteres.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR CARACTER DERECHO.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. ECIB02	NOMBRE DEL PROCEDIMIENTO DELETE()

ENTRADA	PROCESO	SALIDA

	1. if(strlen(abajo->texto)>
	ycur){
	copiar en estring los
	caracteres desde ycur+1
	hasta el último
	caracter de la línea;
	abajo->texto[ycur]='\0';
	abajo->texto[strlen(
	abajo->texto)+1]=' ';
	strcat(abajo->texto,
	estring);
	}
	else
	interrumpir el proceso;
	2. limpiar línea;
	3. printf("%s",
	abajo->texto);
	4. posicionar cursor;
	5. fin de proceso;
		Borra caracter a la derecha de la posición del cursor.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ycur	int	Almacenar la posiciones columna del cursor en la pantalla.
	abajo	editor	Referenciar el nodo (línea de texto) en el que se borra el caracter.
VARIABLE LOCAL	estring	char	Almacenar temporalmente una cadena de caracteres.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR LINEA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCIB03	NOMBRE DEL PROCEDIMIENTO BORRAR L()

ENTRADA	PROCESO	SALIDA

	1. auxabajo=abajo;
	2. if(auxabajo!=inicio &&
	auxabajo!=ultimo)
	borrar nodo(linea de
	texto) apuntado por
	abajo;
	else
	if(auxabajo==inicio &&
	auxabajo=ultimo){
	/*unico nodo (linea
	texto*/
	inicio->texto[0]='\0';
	xcur=3; ycur=0;
	ax=ay=1;
	limpiar linea;
	posicionar cursor;
	}
	else
	if(auxabajo==inicio)
	borrar primer nodo de
	lista;
	else
	//auxabajo==ultimo
	borrar último nodo de
	lista;
	3. desplegar lista a partir
	de la posición de la linea
	borrada;
	4. posicionar cursor;
	5. fin de proceso;
		Borra linea
		de texto en
		la posición
		del cursor.

... CONTINUACION

PROCEDIMIENTO No. FCIBO3 NOMBRE DEL PROCEDIMIENTO BORRAR L()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	xcur, ycur	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la pantalla.
	ax, ay	int	Almacenar las posiciones de fila y columna respectivamente, del cursor en la ventana del editor.
	inicio, ultimo	editor	Almacenar las direcciones del primero y último nodo de LISTA, respectivamente.
	abajo	editor	Referenciar el nodo (línea de texto) a borrar.
VARIABLE LOCAL	auxabajo	editor	Referenciar temporalmente el nodo a borrar, permitiendo establecer los nuevos enlaces en LISTA.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR HASTA FINAL DE LINEA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y EREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. ECIB04 NOMBRE DEL PROCEDIMIENTO B_FINAL_L()	

ENTRADA	PROCESO	SALIDA
	1. limpiar línea desde ycur hasta último caracter de línea; 2. auxycur=ycur; 3. while(abajo->texto [auxycur++])!='\0'; abajo->texto[auxycur]= ' '; 4. abajo->texto[ycur]='\0'; 5. fin de proceso;	Borra cadena de caracteres a la derecha del cursor.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ycur	int	Almacenar las posición columna del cursor en la pantalla.
	abajo	editor	Referenciar el nodo (línea de texto) en el que se va a efectuar el borrado.
VARIABLE LOCAL	auxycur	register	Almacenar el valor de ycur, para ser utilizado en el ciclo while.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR HASTA INICIO DE LINEA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No FCIR05 NOMBRE DEL PROCEDIMIENTO B_INIC_L()	

ENTRADA	PROCESO	SALIDA

	1. if(strlen(abajo->texto)<= xcur){
	llenar nodo(linea de texto) con el caracter blanco;
	abajo->texto[0]='\0';
	xcur=0; ay=1;
	}
	else{
	copiar en estring los caracters desde xcur+1 hasta el último caracter de la línea;
	llenar nodo(linea de texto) con el caracter blanco;
	abajo->texto[0]='\0';
	strcat(abajo->texto, estring);
	xcur=0; ay=1;
	printf("%s", abajo->texto);
	}
	2. posicionar cursor;
	3. fin de proceso;
		Borra cadena de caracteres a la izquierda del cursor

... CONTINUACION

PROCEDIMIENTO No FCIB05 NOMBRE DEL PROCEDIMIENTO B_INIC_L()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	ycur	int	Almacenar la posición columna del cursor en la pantalla.
	ay	int	Almacenar la posición del cursor en la ventana del editor.
	abajo	editor	Referenciar el nodo (línea de texto) donde se va a efectuar el borrado.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento BORRAR PALABRA IZQUIERDA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.		ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL		FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCIB06	NOMBRE DEL PROCEDIMIENTO B P IZQ()	
ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

Procedimiento BORRAR PALABRA DERECHA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.		ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL		FECHA SEP\93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO		
PROCEDIMIENTO No. FCIB07	NOMBRE DEL PROCEDIMIENTO B P DER()	
ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

Procedimiento INSERTAR LINEA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.EC1B08 NOMBRE DEL PROCEDIMIENTO INSERTA L()	

ENTRADA	PROCESO	SALIDA

	1. if(ax>20 && xcur>22){
	if(ycur==0)
	insertar al inicio de la linea 23;
	else
	if((strlen(abajo->texto)+1)>ycur)
	insertar en la posición ycur de la linea 23;
	else
	insertar al final de la linea 23;
	ycur=0; ay=1; ++ax;
	posicionar cursor;
	}
	else{
	if(ycur==0)
	insertar inicio de la linea xcur;
	else
	if((strlen(abajo->texto)+1)>ycur)
	insertar en la posición ycur de la linea xcur;
	else
	insertar al final de la línea xcur;
	ycur=0; ay=1;
	++xcur; ++ax;
	posicionar cursor;
	}
	2. fin de proceso;
		Inserta línea en la posición del cursor.

... CONTINUACION

PROCEDIMIENTO No.ECIROS NOMBRE DEL PROCEDIMIENTO INSERTA_L()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	xcur, ycur	int	Almacenar las posiciones de fila y columna respectivamente, del cursor en la pantalla.
	ax, ay	int	Almacenar las posiciones de fila y columna respectivamente, del cursor en la ventana del editor.
	abajo	editor*	Referenciar el nodo (línea de texto) donde se va a efectuar la inserción.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento INSERTAR CARACTER.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. ECIR09	NOMBRE DEL PROCEDIMIENTO INS CAR()

ENTRADA	PROCESO	SALIDA

	1.if(((strlen(abajo->texto)
	+1)>79) || ay>79)
	interrumpir proceso;
	2.if(isprint(caracter
	tecleado)){
	if((strlen(abajo->texto)
	+1)>ycur){
	copiar en estring
	los caracteres desde
	ycur hasta el último
	caracter de la línea;
	abajo->texto[ycur]=
	caracter tecleado;
	abajo->texto[ycur+1]=
	'\0';
	strcat(abajo->texto,
	estring);
	printf("%s",
	abajo->texto);
	}
	else{
	abajo->texto[strlen
	(abajo->texto)]=' ';
	abajo->texto[ycur]=
	caracter tecleado;
	printf("%c",
	abajo->texto[ycur]);
	}
	++ycur; ++ay;
	posicionar cursor;
	}
	3. fin de proceso;
		Inserta caracter en la posición del cursor.

... CONTINUACION

PROCEDIMIENTO No. EC1B09 NOMBRE DEL PROCEDIMIENTO INSCAR()			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	xcur, ycur	int	Almacenar las posiciones fila y columna respectivamente, del cursor en la pantalla.
	ax,ay	int	Almacenar las posiciones de fila y columna respectivamente, del cursor en la ventana del editor.
	abajo	editor	Referenciar el nodo (línea de texto) en el que se inserta el caracter.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

3.1.6.9 PROCEDIMIENTOS DEL MODULO ESPECIAL.

Los procedimientos que forman este módulo son: backups(), repetir_comando(), abortar_comando(), reemplazar_cadena(), busqueda_arriba(), busqueda_atras(), mayuscula_minuscula(), ASCII() y RATON(), fueron definidos con el propósito de que el usuario disponga de un conjunto de comandos que le permitieran ejecutar funciones especiales, como por ejemplo la búsqueda de cadenas de caracteres, pero por las consideraciones descritas en 3.1.6.2 no fue posible su implementación.

3.1.6.10 PROCEDIMIENTOS DEL MODULO COBOL.

Procedimiento ACTIVAR/DESACTIVAR FUNCIONES.

UNIVERSIDAD DE EL SALVADOR FACULTAD D E CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC001 NOMBRE PROCEDIMIENTO ACTIVAR/DES().	

ENTRADA	PROCESO	SALIDA

	1. //lista vacía
	if(inicio==NULL &&
	inicio->texto[0]=='\0'){
	abrir el archivo
	estruct.cbl para
	lectura;
	strcpy(nombre,
	"estruct.cbl");
	/*llamada a la función
	leer()*/
	cargar el archivo
	}
	2. almacenar en el arreglo
	axt la extensión dal
	nombre de archivo cargado
	en el entorno actual;
	3. if(!strcmp("cbl",ext) &&
	!ADC) ADC=1; //activar
	else ADC=0; /*desactivar
	funciones cobol*/
	4. posicionar cursor;
	5. fin de proceso;
		Despliega en pantalla la estructura general de un programa COBOL, o mensaje COBOL ACTIVO en la línea de estado, o desactivar las funciones para editar texto en COBOL.

... CONTINUACION

PROCEDIMIENTO No FCL001 NOMBRE PROCEDIMIENTO ACTIVAR/DES( )			
OBJETOS UTILIZADOS			
OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	cab	editor	Almacenar la dirección del nodo cabecera de LISTA.
	inicio	editor	Almacenar la dirección del primer nodo de LISTA.
	abajo	editor	Recorrer LISTA.
	ultimo	editor	Almacenar la dirección del último nodo de LISTA.
	nombre	char	Almacenar el nombre del archivo cargado en el entorno actual.
	ext	char	Almacenar la extensión del nombre del archivo cargado en el entorno actual.
	ADC	char	Almacenar el valor 1 si la función Activar/des está activada y el valor 0 si no lo está.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

Procedimiento COMPILAR PROGRAMA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCCO02 NOMBRE DEL PROCEDIMIENTO COMPILAR()	

ENTRADA	PROCESO	SALIDA

	1. almacenar en el arreglo ext la extensión del nombre de archivo cargado en el entorno actual;
	2. if(strcmp("cbl",ext)
	interrumpir proceso;
	3. salvar pantalla;
	4. //compila el programa
	system(compilador);
	5. getch(); //espera tecla
	6. restaurar pantalla;
	7. fin de proceso;
		Genera y almacena archivo con extensión COB en el directorio actual.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	nombre	char	Almacenar el nombre del archivo cargado en el entorno actual.
VARIABLE LOCAL	ext	char	Almacenar la extensión del nombre del archivo cargado en el entorno actual.
	compilador	char	Almacenar la orden compilador nombre.cbl.

Procedimiento EJECUTAR PROGRAMA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCCOO3	NOMBRE DEL PROCEDIMIENTO EJECUTAR()

ENTRADA	PROCESO	SALIDA

	1. almacenar en el arreglo ext la extensión del nombre de archivo cargado en el entorno actual;
	2. if(stromp("cbl",ext) interrumpir proceso;
	3. salvar pantalla;
	4. //ejecuta el programa system(compilador);
	5. getch(); //espera tecla
	6. restaura pantalla;
	7. fin de proceso;
		Resultados que genera el programa ejecutado.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	nombre	char	Almacenar el nombre del archivo cargado en el entorno actual.
VARIABLE LOCAL	ext	char	Almacenar la extensión del nombre del archivo cargado en el entorno actual.
	compilador	char	Almacenar la orden compilador nombre.cob.

Procedimiento PALABRAS RESERVADAS.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC004	NOMBRE DEL PROCEDIMIENTO RESERVA P( )

ENTRADA	PROCESO	SALIDA
	1.almacenar en el arreglo ext la extensión del nombre de archivo cargado en el entorno actual; 2. if(strcmp("cbl",ext) interrumpir proceso; 3. abrir el archivo palabras.obl para lectura; 4.salvar porción de pantalla; 5.desplegar el archivo palabras.obl; 6.restaurar porción de pantalla; 7. fin de proceso;	Ventana en la esquina superior derecha de la pantalla que muestra archivo palabras.cbl.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	nombre	char	Almacenar el nombre del archivo cargado en el entorno actual.
VARIABLE LOCAL	ext	char	Almacenar la extensión del nombre del archivo cargado en el entorno actual.

Procedimiento PLANTILLA.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No.FCCC005 NOMBRE DEL PROCEDIMIENTO PLANTILLA()	

ENTRADA	PROCESO	SALIDA

	1. almacenar en el arreglo ext la extensión del nombre de archivo cargado en el entorno actual;
	2. if(strcmp("cbl",ext)
	interrumpir proceso;
	3. if(!PC){
	PC=1;
	salvar menú principal:
	desplegar plantilla COBOL, en la posición del menú principal;
	else{
	PC=0;
	restaurar menú principal
	4. posicionar cursor;
	7. fin de proceso;
		Despliega en pantalla la plantilla COBOL o el menú principal.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	PC	int	Almacenar el valor 1 si la plantilla COBOL está activa y el valor 0 si no lo está.
VARIABLE LOCAL	ext	char	Almacenar la extensión del nombre del archivo cargado en el entorno actual.

Procedimiento VARIABLES.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC006	NOMBRE DEL PROCEDIMIENTO VARIABLES

ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

Procedimiento PARRAFOS.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP\93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCC007	NOMBRE DEL PROCEDIMIENTO PARRAFO()

ENTRADA	PROCESO	SALIDA
	/* PROCEDIMIENTO POR IMPLEMENTAR */	

3.1.6.11 PROCEDIMIENTOS DEL MODULO SALIR.

Procedimiento SALIR TEMPORALMENTE AL DOS.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCS01	NOMBRE DEL PROCEDIMIENTO SALDOS()

ENTRADA	PROCESO	SALIDA

	1. salvar pantalla;
	2. printf("Digite 'SALIR' para retornar al FCEDIT");
	//prompt de la unidad actual
	3. printf("%/c\\>", 'A'+getdisk());
	4. while(1){
	gets(comando);
	if(strcmp(comando, "SALIR") || strcmp(comando, "salir"))
	interrumpir proceso;
	//ejecuta orden DOS
	system(comando);
	printf(".%c\\>", 'A'+getdisk());
	}
	5. restaurar pantalla;
	6. fin de proceso;
		Transfiere el control al DOS.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE LOCAL	comando	char	Almacenar la orden de usuario la cual sirve de parámetro a la función system, para su ejecución.

Procedimiento SALIR AL DOS.

UNIVERSIDAD DE EL SALVADOR FACULTAD DE CC. NN. Y MAT.	ESCUELA DE MATEMATICA SEC. DE EST. Y COMPUTACION
PROYECTO EDITOR DE TEXTO ORIENTADO A COBOL	FECHA SEP/93
DISEÑO ROSA CRISTINA LEON y FREDYS ARTURO GARCIA PORTILLO	
PROCEDIMIENTO No. FCS02	NOMBRE DEL PROCEDIMIENTO SALIR()

ENTRADA	PROCESO	SALIDA
	1. liberar memoria; 2. limpiar pantalla; 3. posicionar cursor en (0,0) 4. exit(0); 5. fin de proceso;	Cancela la ejecución del FCEDIT, devuelve el control del PC al DOS.

OBJETOS UTILIZADOS

OBJETO	NOMBRE	TIPO	PROPOSITO
VARIABLE GLOBAL	cab, inicio, abajo, ultimo	editor	Referenciar memoria utilizada por el archivo cargado en el entorno actual.
	primero, final	editor	Referenciar memoria utilizada por texto borrado del archivo cargado en el entorno actual que aún no ha sido restaurado.
TIPO DE DATO GLOBAL	editor	struct	Almacenar LISTA.

3.2 Diseño físico.

3.2.1 Escritura de código fuente.

A partir del diseño lógico se realizó la programación del FCEDIT en lenguaje C, en esta fase se realizaron las actividades de elaboración de los programas para los procesos que se definieron en cada módulo del FCEDIT, las pruebas, la depuración de programas y la creación de archivos que soportan los procesos.

La técnica de programación utilizada en la elaboración de los programas es el METODO DE DISEÑO DESCENDENTE, por permitir la producción de un código claro y legible facilitando con ello la clarificación de la estructura general y de operación del programa.

Por acuerdo de los responsables del proyecto, el listado de archivos de programas fuente se entregó a la coordinación de la sección de computación, de la Escuela de Matemática.

3.2.2 Pruebas y evaluación de resultados.

Con el objeto de establecer si los programas elaborados para los procesos definidos generaban los resultados dados en el diseño lógico, se estableció un plan de pruebas de unidad, y una prueba de integración, las cuales se realizaron tomando en cuenta los requerimientos informáticos y operativos relevantes del FCEDIT.

El plan de pruebas de unidad consistió en un conjunto de

pruebas funcionales y de estructura para cada función o proceso, antes de la prueba de integración de todo el software, con el objeto de minimizar esfuerzos al realizar la revisión final luego de integrar todos los procesos en el producto de software en desarrollo.

Las pruebas de los programas se efectuaron conjuntamente con los responsables del proyecto y comprendió la generación de resultados importantes para el funcionamiento del FCEDIT.

Las pruebas realizadas pusieron de manifiesto la calidad de los programas y la obtención de resultados inesperados de acuerdo a la definición de requerimientos, lo cual fue corregido para alcanzar el objetivo deseado.

INTRODUCCION

El presente manual tiene como objetivo orientar al usuario en el uso del FCEDIT, que conozca la forma de utilizarlo correctamente para obtener los resultados deseados.

El manual se ha organizado en tres fases:

- Primero se presentan los requerimientos mínimos de hardware que deben tenerse para instalar el FCEDIT.

- Luego se presenta el proceso de instalación del FCEDIT, con lo cual se garantiza su funcionamiento.

- Al final se presenta una plantilla con los comandos (agrupados por módulo) que ejecutan cada función del FCEDIT, excepto aquellos que no se implementaron por razones de tiempo.

4.1 Requerimientos de hardware.

El FCEDIT fue diseñado en una computadora equipada con una tarjeta VGA, monitor a color y 1024K de memoria RAM, pero en la fase de pruebas se sometió a computadoras con otras características, obteniendo los mismos resultados.

Por tratarse de una herramienta de software que utiliza asignación de memoria en forma dinámica, depende directamente del tamaño de memoria RAM para aprovechar al máximo las

facilidades que el FCEDIT posee para el manejo de texto normal o código fuente en COBOL. El usar un monitor a color o monocromático, no impide el funcionamiento del FCEDIT, puesto que tiene la capacidad de determinar el tipo de monitor disponible, en el momento que se carga en la memoria de la computadora. Por lo tanto, la configuración mínima que debe tener su computadora es de dos unidades lectoras/escritoras de disco de doble cara y de doble densidad, 640K en memoria de acceso aleatorio (RAM) y una impresora de 80 caracteres. Esta configuración garantiza el funcionamiento del FCEDIT, para cumplir con los propósitos u objetivos que motivaron su diseño.

El control de los diferentes tipos de impresoras está fuera del alcance del FCEDIT. Para propósitos de la función imprimir, de la opción ARCHIVO del main menú, se programó en un impresor EPSON LX-810.

4.2 Proceso de instalación.

El FCEDIT es un programa sencillo, cuyo software se almacena en un único disquete, el cual no requiere de procesos complicados para su adaptación a un sistema particular de computación.

Esta herramienta de software la forman tres programas que se describen en la tabla-4.1 :

Tabla-4.1 Programas que forman al FCEDIT.

Programa	Propósito	Tamaño (en byte)
fc.exe	Permite al usuario cargar el FCEDIT en la memoria de la computadora.	129,519
estruct.cbl	Contiene una estructura general de un programa COBOL, la cual puede ser modificada por el usuario de acuerdo a sus necesidades de programación.	511
palabras.cbl	Contiene un conjunto de palabras reservadas en COBOL.	4,401

Los programas estruct.cbl y palabras.cbl se constituyen en archivos de soporte a los procesos que se realizan cuando se utiliza el FCEDIT en aplicaciones de COBOL, estos archivos deben de aparecer en el directorio del disco que contiene el programa fc.exe, para evitar errores de tipo funcional.

El FCEDIT para cumplir con las tareas de computación para las cuales fue creado, se dispone en dos disquetes.

Los nombres de los disquetes son:

1. El disquete del FCEDIT.

Este disco contiene los tres programas antes descritos (fc.exe, estruct.cbl y palabras.cbl).

2. El disquete del compilador COBOL.

Este disco contiene el programa completo del compilador COBOL.

El contenido de este disquete puede ser reemplazado, dependiendo del compilador que utilice el programador, el disco que se presenta aquí contiene los archivos necesarios para compilar código fuente escrito bajo la sintaxis del RM-COBOL.

Si su computador tiene características similares a las que se detallaron en la sección 4.1, la forma de instalar o poner en marcha el FCEDIT es la siguiente:

1. Proceda a la carga inicial de su computadora insertando el disco del sistema operativo en la unidad A.
2. Con el sistema operativo activo, inserte el disquete del FCEDIT en la unidad B y ejecute la orden:

B:FC

para cargar en la memoria de la computadora el FCEDIT.

3. Si utiliza el FCEDIT como un editor de texto de propósito general, no necesita el disquete del compilador COBOL; pero si procesa código fuente COBOL, retire el disquete del sistema operativo de la unidad A e inserte el disquete del compilador COBOL y continúe con su trabajo.

Si su computador tiene además disco duro siga los siguientes pasos para instalar o poner en marcha al FCEDIT.

1. Proceda a la carga inicial de su computadora.
2. Con el sistema operativo activado (C:\>_), proceda a crear un subdirectorio para el sistema de archivos del FCEDIT.

Ejecute la orden:

MD EDITOR\FC

para crear un subdirectorio actual.

3. Teclee:

CD EDITORFC

para introducirse al subdirectorio creado y convertirlo en el directorio activo.

4. Copie cada disquete original al disco duro de su computadora, proceda de la siguiente manera:

4.1 Inserte el primer disco en la unidad A, a continuación del indicativo C:\EDITORFC> del sistema operativo, teclee:

COPY A:.*/V*

4.3 Instrucciones de usuario.

Se presenta en esta sección un resumen de las funciones o comandos del FCEDIT, como una referencia instantánea que facilite al usuario el uso de esta herramienta de software.

RESUMEN DE COMANDOS DEL FCEDIT

COMANDOS DE ARCHIVO

. Leer archivo	Alt+L
. Escribir archivo	Alt+E
. Insertar archivo	Alt+I
. Imprimir archivo	Alt+P
. Ignorar cambios en el archivo	Alt+G

COMANDOS DE INSERTAR/BORRAR

. Borrar un caracter a la izquierda	BACKSPACE
. Borrar un caracter a la derecha	DELETE
. Insertar línea	ENTER
. Borrar línea	Alt+Y
. Borrar hasta el final de línea	Ctrl+END
. Borrar hasta el inicio de línea	Ctrl+HOME

COMANDOS DE BLOQUE

. Marcar bloque	Alt+M
. Copiar bloque	Alt+O
. Mover bloque	Alt+V
. Borrar bloque	Alt+R
. Restaurar bloque	Alt+S

COMANDOS DE COBOL

. Activar/Desactivar funciones COBOL	<F2>
. Compilar programa	<F3>
. Ejecutar programa	<F4>
. Palabras reservadas	<F5>
. Plantilla numerada para COBOL	<F6>

COMANDOS DE CURSOR

. Tabulador	TAB
. Mover cursor al inicio de línea	HOME
. Mover cursor al final de línea	END
. Mover cursor al inicio de archivo	Ctrl+PgUp
. Mover cursor al final de archivo	Ctrl+PgDn
. Mover cursor página hacia arriba	PgUp
. Mover cursor página hacia abajo	PgDn

COMANDOS DE SALIR

. Salir temporalmente al DOS (Os Shell)	<F8>
. Salir al DOS	<F9>

OTROS COMANDOS

<F1>: Ayuda
<F10>: Activa menú principal
<ESC>: Tecla de escape

COMANDOS DE VENTANA

*POR IMPLEMENTAR

COMANDOS ESPECIALES

*POR IMPLEMENTAR

Nota: Si COBOL está activo, el TAB desplaza el cursor a las columnas 8, 12 y 72, caso contrario tiene un funcionamiento normal.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

INTRODUCCION.

El FCEDIT es un editor de texto orientado a COBOL, con funciones de propósito general que permiten utilizarlo además, en la edición de código fuente escrito en otros lenguajes de programación.

El plan de solución que se siguió en el desarrollo del proyecto, está basado en la metodología sugerida para el desarrollo de un producto de informática. Este plan se inició con una fase de investigación a nivel de empresas (en el área de San Salvador) que desarrollan aplicaciones en lenguaje COBOL, para obtener toda la información posible sobre los editores utilizados en esta tarea.

Luego de esto se realizó un estudio y evaluación de las ventajas y desventajas que dan tales editores, y sobre otros editores más conocidos en el mercado, esto permitió seleccionar, definir y clasificar el conjunto de funciones que forman el FCEDIT.

Previo a la fase de diseño se hizo necesario corresponder a cada función un nombre mnemónico con el que sería identificado en todo el desarrollo del software, asignar una secuencia de teclas a cada función para ser ejecutada desde la ventana del editor y se seleccionó las estructuras de datos

que se utilizaron en el diseño.

El diseño del editor se llevó a cabo en dos etapas bien diferenciadas, como son el diseño lógico y el diseño físico.

El diseño lógico se elaboraron diagramas de tipo HIPO (Hierarchy-Input-Process-Output) para la documentación de procedimientos o funciones que forman los diferentes módulos del FCKDIT.

En el diseño físico se escribió el código fuente, en lenguaje C, de todas las especificaciones hechas para cada módulo.

La escritura de código fuente puso de manifiesto en un primer momento el bajo nivel de conocimiento que se tenía del lenguaje C, lo cual fué superado en poco tiempo de manera muy considerable.

Los problemas enfrentados en la elaboración de programas y que fueron resueltos con apoyo bibliográfico y entrevistas a expertos, se agrupan en tres categorías fundamentales como son:

- El manejo de memoria.
- El manejo de strings.
- El control de pantalla.

La prueba y evaluación de resultados se realizó conjuntamente con los responsables del proyecto, utilizando la técnica de pruebas de unidad, con el objeto de facilitar la revisión del editor como producto final, esto puso de

manifiesto, desviaciones de los resultados esperados según el diseño lógico, lo cual fue corregido para alcanzar el objetivo deseado.

5.1 Conclusiones.

Con base en el trabajo realizado y la experiencia adquirida en el desarrollo de este proyecto se concluye que:

- La estructura de los datos utilizada para alojar un archivo en la memoria del computador, es una lista doblemente enlazada con cabecera, a la que se identificó durante el desarrollo del proyecto con el nombre de LISTA. Esta estructura fue seleccionada por considerarse flexible a cualquier operación en la edición de texto y por facilitar su manejo en correspondencia con los cambios de posición del cursor en la pantalla. Esto es, el recorrido de LISTA en cualquier dirección, depende de si el cursor se desplaza en pantalla una línea hacia abajo o una línea hacia arriba, lo cual es posible dado que cada nodo de LISTA almacena una línea de texto del archivo.

- Para el control del cursor en la pantalla se utilizaron dos sistemas de coordenadas, puesto que es necesario manejar las posiciones de líneas en el rango de 0 a 24 (número de líneas de pantalla) y las posiciones de líneas del cursor que están en correspondencia directa con las líneas de texto del archivo que se maneja. Estos sistemas de coordenadas se manejan con la

declaración de cuatro variables globales a todo el programa, que permiten el control total de los movimientos del cursor en la pantalla. Para estos propósitos se decidió crear una función equivalente a la función predefinida de C: gotoxy(), que permitiera desplazar el cursor a cualquier posición de pantalla especificada en filas y columnas, en un rango de 0 a 24 para filas y de 0 a 79 para columnas.

- La estructura de datos que se utiliza para el manejo de los menús abatibles es un arreglo de punteros a cadenas de caracteres. Esta estructura fue seleccionada porque permite almacenar un número fijo de datos, y el acceso directo a cualquiera de sus elementos, con sólo especificar su posición. Para cada menú se define un arreglo cuyos elementos corresponden al número de items del menú.

- La técnica que se utiliza para la documentación de procedimientos en el diseño lógico, fue la técnica de diseño mediante diagramas de tipo HIPO (Hierarchy-Input-Process-Output), en los cuales se representan de forma clara y precisa las entradas, los pasos de los procesos y las salidas o resultados que genera cada procedimiento. La elección de los diagramas de tipo HIPO como esquemas para la documentación de procedimientos, se hizo de entre las diferentes técnicas de diseño que ofrece la Ingeniería de Software, por considerar que se trata de una técnica fácil en la cual se ilustra mediante pseudocódigo estructurado, los algoritmos

correspondientes a cada procedimiento o función.

- En la planificación del proyecto, se consideró en un principio la implementación de las funciones del DOS: **renombrar archivo, borrar archivo, cambiar directorio y ver directorio**, las cuales fueron eliminadas por considerar que su uso (y de todas los comandos del DOS) se habilita con la opción **salir al DOS**, la cual permite una salida temporal al sistema operativo sin cancelar la ejecución del FCEDIT.

- El uso del C, como lenguaje de programación en el que se elaboraron los programas, fue una decisión acertada, pues se trata de un lenguaje que permite programar a bajo y alto nivel, además posee un conjunto de funciones predefinidas que facilitan el manejo de memoria dinámica, de strings y de los dispositivos de almacenamiento. También porque el uso del C garantiza la portabilidad y transportabilidad de los programas a otros ambientes.

- El tamaño de los BUFFER's, para salvar y restaurar las porciones de pantalla en las que se visualizan los menús abatibles, se determinó mediante la siguiente ecuación:

$$TAMAÑO = líneas \times columnas \times 2$$

donde,

líneas: es el número de ítems del menú en cuestión.

columnas: es el número de caracteres de la cadena (ítem) con mayor longitud.

líneas x columnas: es el total de caracteres, este resultado se multiplica por dos (2), porque para almacenar cada caracter en memoria se requieren dos bytes: el primero para almacenar el caracter y el segundo para almacenar el atributo con que se visualiza en pantalla.

- La asignación de memoria para cada proceso, se hizo utilizando el sistema de asignación dinámica de C. En un primer momento se utilizaron las funciones predefinidas:

malloc(t): para asignar un bloque de memoria de por lo menos t bytes.

Esta función devuelve un puntero que referencia el espacio asignado. Si hay insuficiente espacio de memoria o si t es 0, la función retorna un puntero nulo (NULL).

free(p): para liberar el espacio de memoria apuntado por p.

Esto fue funcional mientras el tamaño del programa era menor que 64K y se utilizaba el modelo de memoria pequeño (SMALL), en el cual tanto el código como los datos son accedidos con **direcciones near**. Esto restringió el cargar en la memoria un archivo cuyo tamaño fuese mayor que 64K. Fue a raíz de esto que se optó por utilizar el modelo de memoria grande (LARGE), lo cual obligó el uso de **direcciones far**; es así como las funciones **malloc()** y **free()** fueron sustituidas por **farmalloc()** y **farfree()** respectivamente, y la dirección near se sustituyó por la **dirección far**, logrando con ello el

acceso a cualquier parte de la memoria.

- Para los propósitos del editor, no fue factible el uso de la función ***clrscr()*** *predefinida del C, para borrar la pantalla.*

Pues con esta función no es posible borrar partes de la pantalla, esto obligó la creación de la función ***cls()****, la cual resuelve esta tarea con sólo especificar las coordenadas del punto superior izquierdo y del punto inferior derecho de la porción de pantallla que se quiere borrar.*

- Las pruebas y evaluación de resultados, del FCKEDIT a nivel de usuarios, no se realizó debido a las limitantes de tiempo, que no fue posible superar.

De acuerdo a los planteamientos anteriores se hacen las siguientes:

5.2 Recomendaciones.

- Que para el diseño de editores, es apropiado el uso de estructuras dinámicas de datos, como lo es una lista doblemente enlazada con cabecera, en la cual se aloje el texto que se edita o el archivo que se transporte desde cualquier unidad de disco a la memoria de la computadora.

- Que para el manejo y direccionamiento de la memoria en un programa como el diseñado en este proyecto, debe utilizarse el modelo de memoria grande (LARGE), con un direccionamiento de tipo ***far****, para facilitar el acceso a cualquier parte de la memoria.*

- *Usar el modificador register (palabra clave de C) en la declaración de variables locales, para accesar lo más rápido posible a su contenido.*

- *Utilizar la opción proyect del menú principal del editor del C, para crear proyectos (pequeños programas que resuelven una tarea específica) que serán compilados individualmente y luego enlazados para crear el programa ejecutable. Esto evita un OVERFLOW en el proceso de compilación de programas muy grandes.*

- *Las pruebas y evaluación de resultados, deben hacerse utilizando la técnica de pruebas de unidad, para garantizar que el producto de software en desarrollo genera resultados acordes con la definición de requerimientos y para invertir menos esfuerzo en la revisión final, luego de integrar todos los procesos.*

- *Los criterios de programación modular deben ser determinantes en la elaboración de programas, sin importar que tan grandes sean, puesto que facilitan la depuración, las pruebas, el ajuste y la modificación del software.*

- *Debe incluirse la participación del usuario en la prueba y evaluación de resultados del software como producto final, para demostrarle hasta que punto la programación satisface sus necesidades.*

BIBLIOGRAFIA.

LIBROS

- Achaerandio S. J, Luis. Iniciación a la Práctica de la Investigación. Editorial Universitaria URL. Guatemala, C. A. 1989.

- Ceballos Sierra, Francisco Javier. Curso de Programación "C" (Microsoft C) Macrobit Editores. México. 1990.

- Claybrook, Billy G. File Management Techniques. Editorial John Wiley & Sons, Inc. USA. 1983.

- Dale, Nell y Lilly, Susan C. Pascal y Estructura de Datos. Editorial McGraw-Hill. México. 1988.

- Duncan, Ray. Guía de Referencia Rápida para Programadores. Funciones del MS-DOS. Editorial Anaya Multimedia-Microsoft. Madrid, España. 1989.

- Fairley, Richard. Ingeniería de Software. Editorial McGraw-Hill. México. 1988.

- García de Sola, Juan F. y Garcerán Hdez, Vicente. Lenguaje C y Estructura da Datos. Aplicaciones Generales y de Gestión.

Editorial McGraw-Hill. Madrid, España. 1992.

- Gottfried, Byron S. Programación en C. Serie Schaum. Editorial McGraw-Hill. Madrid, España.

- Godfred, J. Terry. Lenguaje Ensamblador para Microcomputadoras IBM, para principiantes y avanzados. Editorial Prentice-Hall. México. 1992.

- Ibeas, F Franco. Diccionario Tecnológico Inglés-Español. Editorial Alhambra. España. 1989.

-Jourdain, Robert L. Programmer's Problem Solver for the IBM PC, XT & AT. Editorial Brady Books. USA. 1986.

-Kernighan, Brian W. y Ritchie, Dennis M. El lenguaje de Programación "C". Editorial Prentice-Hall. México. 1991.

- Kruse, Robert L. Estructura de Datos y Diseño de Programas. Editorial Prentice-Hall. México. 1988.

-Lawrence R. Newcomer, M. S. Programación en Cobol Estructurado. Editorial McGraw-Hill. México. 1991.

- Lafore, Robert. Object-Oriented Programming in MICROSOFT C++. Editorial The Waite Group. USA. 1992.

- Leventhal, Lance. Guía de Programación 80386. Editorial Macrobit. México. 1991.

- Lipschutz, Seymour. Estructura de Datos Serie Schaum en Computación. Editorial McGraw-Hill. México. 1988.

- Loomis, Mary E. S. Estructura de Datos y Organización de Archivos. Editorial Prentice-Hall. México. 1991.

- Murray, Willian H. y Pappas Chris H. 80386/80286 Programación en Lenguaje Ensamblador. Editorial McGraw-Hill. México. 1987.

- Neibuer, Alan R. El ABC del Wordperfect 5.1. Editorial Ventura. México. 1991.

- Pappas, Chris H. y Murray Willian H. Manual del Microprocesador 80386. Editorial Osborne/McGraw-Hill. Madrid, España. 1987.

- Philippakis, Andreas S. y Kazmier, Leonard J. Cobol Estructurado. Editorial McGraw-Hill. México. 1989.

- Pressman, Roger S. Ingeniería del Software un Enfoque Práctico. Editorial McGraw-Hill. México. 1988.

- Saybold, Patricia B. y Marshak Ronnit T. Procesamiento de Texto Software para el IBM/PC. Editorial McGraw-Hill. México. 1986.

- Schildt, Herbert. ANSI C a su Alcance. Editorial Osborne/McGraw-Hill. Madrid, España. 1991.

-Schildt, Herbert. C: Guía para Usuarios Expertos. Editorial Osborne/McGraw-Hill. Madrid, España. 1989.

-Schildt, Herbert. Lenguaje C Programación Avanzada. Editorial Osborne/McGraw-Hill. Madrid, España. 1988.

-Schildt, Herbert. Aplique Turbo C++. Editorial Osborne/McGraw-Hill. Madrid, España. 1991.

TESIS

- Armas Mata, Myrna Astrid y otros. Editor de Pantalla para el Sistema Operativo Unix. UCA. 1989.

- Guandique Cordon, Elisa Carolina y otros. Paquete Generador de Aplicaciones en Cobol. UCA.

OTRAS FUENTES

- Entrevista personal con el Lic. Salvador Urías, Coordinador del laboratorio de cómputo del Departamento de Ingeniería Eléctrica de la Universidad Centroamericana "José Simeón Cañas" (UCA), 1993.

Asunto: Manejo de estructuras dinámicas en lenguaje C y el direccionamiento de la memoria de acceso aleatorio (RAM).

APENDICES

ANEXO #1

1.1 Cuestionario.

1.2 Listado de empresas e instituciones en donde se paso la encuesta.

1.3 Tabla de frecuencia de uso de los editores que utilizan las empresas encuestadas.

1.4 Cuadro resumen de cada editor.

1.5 Exigencias de programación en COBOL satisfechas y no satisfechas por los editores utilizados en las empresas encuestadas.

1.6 Necesidades de codificación en lenguaje COBOL, que los editores utilizados no las satisfacen.

1.7 Tipos de aplicaciones que desarrollan las empresas encuestadas.

1.8 Tipo de softwares utilizados en las empresas encuestadas.

1.9 Sugerencias dadas por las personas entrevistadas.

1.10 Editores de mayor uso en las empresas encuestadas.

1.11 Listado de empresas consultadas, las cuales no aportaron información útil al proyecto.

1.1 CUESTIONARIO

INVESTIGACION SOBRE HERRAMIENTAS DE SOFTWARES UTILIZADOS PARA LA CODIFICACION Y COMPILACION DE PROGRAMAS EN COBOL.

OBJETIVO: ***"DETERMINAR QUE EDITORES DE TEXTO SE ESTAN UTILIZANDO EN EL MERCADO NACIONAL, VENTAJAS Y DESVENTAJAS QUE ESTOS OFRECEN Y NECESIDADES QUE SE DESEARIAN FUESEN SUBSANADAS PARA EL DESARROLLO DE APLICACIONES EN LENGUAJE COBOL."***

EMPRESA (o INSTITUCION).
NOMBRE: _____

TELEFONO: _____

ENTREVISTADO.
NOMBRE: _____

AÑOS DE EXPERIENCIA DE PROGRAMAR EN LENGUAJE COBOL: _____

GRADO ACADEMICO: _____

ENTREVISTADOR(ES): _____
_____

1.- Qué editores utilizan para codificar programas en COBOL?. Explique las ventajas y desventajas que presentan cada uno de ellos y en una escala de cero a cien, qué dominio de conocimiento tiene usted sobre el editor?

VENTAJAS

_____ _____
nombre del editor _____
_____

DESVENTAJAS

_____ _____
dominio de conocimiento _____
_____

VENTAJAS

_____ _____
nombre del editor _____
_____

DESVENTAJAS

_____ _____
dominio de conocimiento _____
_____

nombre del editor _____

dominio de conocimiento _____

VENTAJAS

_____

_____

_____

DESVENTAJAS

_____

_____

_____

2.- Los editores que utilizan satisfacen las exigencias de la programación en COBOL?, en cuanto a:

Limitantes en el tamaño de texto _____
Funciones de borrado _____
Facilidades de uso _____
Posee ayudas (HELP) _____
Manejo de menús o comandos _____
Si está integrado el compilador dentro del editor _____
Si el editor puede permanecer en memoria _____
Inserción de código _____
Posee indentación _____
Conjunto de carácteres gráficos _____
Funciones de edición normal _____
Facilidades para el movimiento de cursor _____
Otros, explique _____
_____

3.- Qué necesidades demanda la codificación de programas en COBOL que los editores que ustedes utilizan no las satisfacen?

4.- Qué tipo de aplicaciones son las que ustedes desarrollan?

Manejo de inventarios _____
Contabilidad _____
Control financiero _____
Planilla _____
Otros, explique _____
_____

5.- Qué software (compilador o intérprete, librerías utilitarias de programación extra, versión) utilizan en las aplicaciones que desarrollan?

6.- Qué sugerencias daría?, si lo que se pretende es hacer un EDITOR cuya característica particular sea "facilitar la codificación de programas en lenguaje COBOL".

1.2 LISTADO DE LAS EMPRESAS E INSTITUCIONES EN DONDE SE PASO LA ENCUESTA.

UES - CENTRO DE COMPUTO.

UES - FIA.

CENTRAL DE FIANZAS.

EMPRESARIOS JUVENILES.

CIPC.

MAG.

ANDA.

ANTEL.

TRIBUNAL SUPREMO ELECTORAL.

INPEP.

WANG - INTERDATA.

ISSS.

SIMAN CONSTRUCTORES.

BANCO AGRICOLA COMERCIAL.

AHORROMET.

1.3 TABLA DE FRECUENCIA DE USO DE LOS EDITORES QUE UTILIZAN LAS EMPRESAS EN DONDE SE PASO LA ENCUESTA.

NOMBRE DEL EDITOR	NUMERO DE EMPRESAS QUE LO UTILIZAN
Q	4
EDITOR DE WANG	6
SIDEKICK	4
LSE/SCA	1
EDIT	2
VI-UNIX	2
Dx10	1
EDITOR DEL XTPRO	1
SEE	2
SPF/PC	1
RED	1
TOPICS (INTRAK INC PARA NCR)	1
ICCF-IBM	2
WORDSTAR	1
EE	1

1.4 CUADRO RESUMEN DE CADA EDITOR.

1. Editor Q
2. Editor de WANG
3. Editor de SK
4. Editor Dx10
5. Editor XTPRO
6. Editor SEE
7. Editor SPF/PC
8. Editor RED
9. Editor TOPICS
10. Editor ICCF-IBM
11. Editor WORDSTAR
12. Editor EE
13. Editor LSE/SCA
14. Editor EDIT (del DOS)
15. Editor VI-UNIX

1.- EDITOR Q

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
CENTRO DE COMPUTO UES	RENE HECTOR MARTINEZ	3 AÑOS	EGRESADO LIC. EN MATEMATICA	90%	-ES PEQUEÑO -FACILIDAD PARA LA GESTION DE BLOQUE -ES LATENTE EN EL SISTEMA. -EXPORTA E IMPORTA FICHEROS O PARTES -ES INTELIGENTE	-SOLO PERMITE TRABAJAR UN FICHERO A LA VEZ
CENTRAL DE FIANZAS	LEONARDO CRUZ	2 AÑOS	BACHILLER	100%	-MANEJA MAS DE UN ARCHIVO -FACILIDAD EN MENUS -SALIDA AL DOS CON OS SHELL	-FALTA DE TABLAS DE CODIGO -TABULACIONES
	CARLOS ORLANDO CARTAGENA	1 AÑO	PROGRAMADOR	75%	-QUE EL MANEJO A NIVEL DE BUSQUEDA ES BASTANTE EXACTO	
	CARLOS ALVAREZ	20 AÑOS	LIC. EN ADMON. DE EMPRESAS	100%	-ES VERSATIL -TIENE EXCELENTES OPCIONES	
EMPRESARIOS JUVENILES	MANUEL PEREZ	7 AÑOS	ANALISTA PROGRAMADOR	100%	-MANEJO DE HERRAMIENTAS POR MENUS -CONTROL DE MAS DE UN PROGRAMA EN MEMORIA -FUNCION INTERACTIVA CON EL DOS	
	MIGUEL ERNESTO LANDAVERDE	1 AÑO	BACHILLER	75%	-FACIL MANEJO -OFRECE MANEJO DE VARIAS VENTANAS -FACIL COPIADO DE BLOQUES	-AL SER MUY POBRE LA MEMORIA SE NECESITA ESTAR ENTRANDO Y SALIENDO CADA VEZ QUE SE DESEA USAR
	MOISES ELIAS ARRIAZA	2 AÑOS	ANALISTA PROGRAMADOR	95%	-EXCELENTE MANEJO DE BLOQUES	-NO RESIDE EN MEMORIA COMO EL SK

2. EDITOR DE WANG

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
M A G	OLGA DE GARCIA	7 AÑOS	Ing. INDUSTRIAL	70%	-SIN SALIRSE DEL EDITOR PUEDE ENTRAR AL PROCESADOR DE COMANDOS	-NO PUEDE MANEJAR SIMULTANEAMENTE DOS ARCHIVOS
A N D A	DANIEL ALVARADO ECHEGOYEN	4 AÑOS	2do. DE Ing. EN SISTEMAS	85%	-FACIL SU MANEJO POR EL USO DE MENUS	
A N T E L	JORGE ALEX FUNES	10 AÑOS	5to. DE Ing. EN COMPUTACION	90%	-COMPILA DENTRO DEL EDITOR -EJECUTA PROGRAMAS DESDE EL EDITOR -TRABAJA Y COMPILA EN ASSEMBLER, COBOL, etc.	
TRIBUNAL SUPREMO ELECTORAL	OSCAR ROLANDO ENRIQUEZ B.	9 AÑOS	Ing. PROGRAMADOR	100%	-EL TUTORIAL VA LLEVANDO DE LA MANO AL USUARIO	
I N P E P	VICTOR EDGARDO ZALDAÑA	2 AÑOS	Ing. PROGRAMADOR	100%	-RAPIDEZ EN UTILITARIOS DE EDICION -COMPILACION Y EJECUCION DE PROGRAMAS INTEGRADOS -DEBUGER INTEGRADO	-NO CREA ARCHIVOS DE RESGUARDO (BACKUPS)
INTERDATA	ALVARO JAVIER ARANA	5 AÑOS	NINGUNO	90%	-COMPILAR, ENLAZAR Y EJECUTAR UN PROGRAMA -CORREGIR LOS ERRORES DE UN PROGRAMA -EL DEBUGGER DE UN PROGRAMA -FORMAS PREESTABLECIDAS PARA BASIC, COBOL, LP/I, RPG2, ASSEMBLER	-NO POSEE MACROS -NUMERACION DE LINEAS -SOLAMENTE SE PUEDE EDITAR EL CONTENIDO DEL ARCHIVO FUENTE QUE SE ENCUENTRA ACTUALMENTE DESPLEGADO EN LA PANTALLA

D.- EDITOR SK

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
U E S - F I A	MARIO E. LOPEZ R.	1 AÑOS	TEC. PROGRAMADOR	40%	-ES PORTATIL Y RAPIDO	
C I P C	GERARDO A. QUINTANILLA	3 AÑOS	TEC. PROGRAMADOR	70%	-SE CARGA EN MEMORIA -VERSATILIDAD EN VENTANAS -SE PUEDE INVOCAR EN CUALQUIER MOMENTO	-NO MANEJA MAS DE UN ARCHIVO AL MISMO TIEMPO -NO DETECTA ERRORES -NO REESTRUCTURA BLOQUES DE INFORMACION -NO TRABAJA EN BASE A MARGENES
EMPRESARIOS JUVENILES	MIGUEL ERNESTO LANDAVERDE	1 AÑO	BACHILLER	50%	-RAPIDO, BUENA PRESENTACION Y ES EFICIENTE	-COMO ES UN TSR OCACIONA PROBLEMAS CON OTROS PROGRAMAS DEL MISMO TIPO AL SER ACTIVADOS
	MANUEL PEREZ	7 AÑOS	ANALISTA PROGRAMADOR	100%	-VENTANA INTERACTIVA AL MS-DOS -HERRAMIENTAS ADICIONALES PARA EL MANEJO DE INFORMACION	-SOLO PUEDE MANEJAR UN PROGRAMA EN MEMORIA
	MOISES ELIAS ARRIAZA	2 AÑOS	ANALISTA PROGRAMADOR	100%	-RESIDENTE EN MEMORIA -POSEE CALCULADOR TABLA ASCII	-LENTO EN ACCESO A LA UNIDAD MAGNETICA -CONFLICTOS EN MEMORIA CON OTROS PROGRAMAS RESIDENTES
CENTRAL DE FIANZAS	CARLOS ORLANDO CARTAGENA	1 AÑO	PROGRAMADOR	60%	-BUEN MANEJO DE LINEAS	-TIENE UN TOPE EN EL MANEJO DE PROGRAMAS
	LEONARDO CRUZ	2 AÑOS	BACHILLER	75%	-PUEDE PERMANECER EN MEMORIA -POSEE TABLA DE CODIGOS ASCII	-SATURACION DE MEMORIA CUANDO ESTA EN USO -NO POSEE LA OPCION DE DESCARGA DE MEMORIA

4. EDITOR DX10

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
CENTRO DE COMPUTO U E S	CESAR A. MELENDEZ	5 AÑOS	BACHILLER	90%	-ES VERSATIL, AMIGABLE, FACIL DE USAR	-NO PUEDE MANEJAR SIMULTANEAMENTE DOS ARCHIVOS

5. EDITOR XTPRO

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
CENTRO DE COMPUTO U E S	RENE HECTOR MARTINEZ	2 AÑOS	EGRESADO LIC. EN MATEMATICA	90%	-FACILIDAD PARA GESTION DE BLOQUES -INTELIGENTE -PERMITE EXPORTAR E IMPORTAR SEGMENTOS DE FICHEROS -HACE BACKUPS	-ESTA INCORPORADO AL XTREE, POR LO QUE HAY QUE CARGAR EL PAQUETE PRIMERO -NO ES LATENTE EN EL SISTEMA

6. EDITOR SEE

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
I S S S	JORGE GARCIA	2 AÑOS	LIC. EN COMPUTACION	90%	-FACILIDAD DE USAR -VARIEDAD DE COMANDOS	

7. EDITOR SPF/PC

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
U E I - F I A	MARIO E. LOPEZ R.	2 AÑOS	ING. PROGRAMADOR	40%	-TIENE UTILIDADES INCORPORADAS -PUEDE ACCESARSE DESDE CUALQUIER DIRECTORIO	-NO ES CONOCIDO EN EL AMBIENTE DE PC'S

8 . EDITOR RED

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
M A G	OLGA DE GARCIA	7 AÑOS	Ing. INDUSTRIAL	60%	-MANEJO DE VENTANA EN ARCHIVOS DIFERENTES	-LOS COMANDOS DE EJECUCION NO ESTAN A LA VISTA

9 . EDITOR TOPICS

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
BANCO AGRICOLA COMERCIAL	CARLOS RENE FLORES CHAVEZ	9 AÑOS	---	95%	-FACILIDAD DE COMUNICACION -ACCESO A SPOOL DE IMPRESION -DISPOSICION DE INFORMACION FISICA EN DISCO	-NO HAY PRIVACIDAD DE INFORMACION

10 . EDITOR ICCF-IBM

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
A N T E L	JORGE ALEX FONES	10 AÑOS	Est. Ing. en Computacion	80%	-TODAS LAS OPCIONES MANEJABLES CON EL TECLADO -COMPILAR DESDE EL EDITOR	-EL COPY Y LO MANEJA POR BLOQUES NO POR NUMERO DE LINEA
	JORGE GARCIA	2 AÑOS	Lic. en COMPUTACION	90%	-VERSATILIDAD EN EL MANEJO DE COMANDOS DE EDICION	

11 . EDITOR HORDSTAR

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
U E S - F I A	MARIO E. LOPEZ R.	2 AÑOS	Ing. PROGRAMADOR	60%	-ES PORTATIL -CREA UNA COPIA BACKUP	-ES MUY TARDADO

12 - EDITOR XIC

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
DE EVARIOS SUC MILES	MANUEL PEREZ	7 AÑOS	ANALISTA PROGRAMADOR	100%	- POSEE UN MENU DE [illegible] - [illegible]	- HAY QUE ESTAR [illegible]
	MOISES ELIAS ARRIMAZA	2 AÑOS	ANALISTA PROGRAMADOR	100%	- UN ADECUADO [illegible]	- NO POSEE LA [illegible]

13 - EDITOR LSE/SCA

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
MORROMEI	CARLOS RIVERA CUELLAR	6 AÑOS	---	100%	-GENERA FORMATOS DE PROGRAMAS COBOL	

14 - EDITOR EDIT

EMPRESA	ENTREVISTADO	EXPERIENCIA	GDO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
CENTRO DE COMPUTO O E S	RENE HECTOR MARTINEZ	2 AÑOS	EGRESADO LICA EN MATEMATICA	90%	- FACILIDAD PARA [illegible] - EXISTENTE - INTELIGENTE	- NO PERMITE IMPORTAR [illegible] - NO GENERA BACKUPS - INCORPORADO AL DOS
	CESAR A. MELENDES	8 AÑOS	BACHILLER	30%	-EN CUANTO A [illegible]	
CENTRAL DE FINANZAS	CARLOS ALVAREZ	20 AÑOS	Lic. EN ADMON. DE EMPRESAS	100%	-SE CONSIDERA PRACTICO	-ES DEMASIADO LENTO

15. EDITOR VI-UNIX

EMPRESA	ENTREVISTADO	EXPERIENCIA	GPO. ACADEMICO	DOMINIO	VENTAJAS	DESVENTAJAS
CENTRO DE COMPUTO U. E. S	CESAR A. MELENDEZ	0 AÑOS	BACHILLER	20%	-SE PUEDE TRABAJAR CON VARIOS USUARIOS A LA VEZ, UNIENDO PARTES DE ELLOS.	-TRABAJA BIEN SOLO EN LOS PROFINES CONSOLAS
AHORRONET	CARLOS RIVERA CUELLAR	6 AÑOS	---	100%	-FACIL DE USAR, MULTIBUFFER, etc.	-CONSUME MUCHOS RECURSOS DEL SISTEMA

1.5 EXIGENCIAS DE PROGRAMACION EN COBOL SATISFECHAS Y NO SATISFECHAS POR LOS EDITORES UTILIZADOS EN LAS EMPRESAS DONDE SE PASO LA ENCUESTA.

	No. DE EMPRESAS QUE CONTESTARON SE SATISFACE	NO SE SATISFACE	NADA
1.- Limitantes en el tamaño de texto	16	4	0
2.- Funciones de borrado	19	0	1
3.- Facilidades de uso	19	0	1
4.- Posee ayudas (HELP)	16	3	1
5.- Manejo de menús o comandos	18	2	0
6.- Si está integrado el compilador dentro del editor	8	12	0
7.- Si el editor puede permanecer en memoria	17	3	0
8.- Inserción de código	17	3	0
9.- Posee indentación	13	7	0
10.-Conjunto de caracteres gráficos	8	10	2
11.-Funciones de edición normal	19	0	1
12.-Facilidades para el movimiento de cursor	19	0	1

OTRAS EXIGENCIAS

- Compatibilidad en el uso con otros editores y procesadores de palabra.

- El Q le permite por ejemplo salir del sistema momentáneamente.

- *La combinación de uno o más archivos.*
- *Manejo de ventanas para programas cargados en memoria.*
- *Copia de programas, fusión de 2 ó más programas.*
- *Debugger integrado; compilador integrado.*
- *Podemos copiar otro programa o parte de él, que no este siendo usado, dentro del programa en memoria.*

1.6 NECESIDADES DE CODIFICACION EN LENGUAJE COBOL QUE LOS EDITORES UTILIZADOS NO LAS SATISFACEN.

- Revisión y corrección de indentación.
- Listado de variables y nombres de párrafos.
- Mostrar indicadores de nivel.
- Manejo de directorios para programas fuente objetos y listados de compilación.
- Incorporación del compilador.
- Que esté integrado al compilador y versátil como el Q.
- La tabla de códigos ASCII.
- Las tabulaciones no se fijan por ejemplo en columna 8 ó 12.
- Tener un controlador de línea para aplicar indentación.
- La identificación de errores de sintaxis.
- Editores más rápidos que posean ventanas para compilar más fácil los programas fuentes.
- Cada vez que se abre una línea nueva, el cursor debería ubicarse donde empezó la anterior.
- La estructura de un programa en COBOL, debería ser tomada en cuenta.

1.7 TIPOS DE APLICACIONES QUE DESARROLLAN LAS EMPRESAS DONDE SE PASO LA ENCUESTA.

NOBRE DE APLICACION	No. DE EMPRESAS QUE LAS DESARROLLAN
Manejo de inventarios	12
Contabilidad	16
Control financiero	14
Planillas	17

OTRAS APLICACIONES

- Cuentas corrientes.
- Facturación.
- Sistema de registro académico.
- Control de cheques pagados.

  Recibos de escolaridad, nuevo ingreso.
- Inscripción de asignaturas.
- Recolector de notas.
- Elaboración de causes, fianzas y seguros.
- Control de cálculos de mortalidad.
- Pérdidas aplicadas a granjas.
- Control en expedientes de alumnos o en el registro

  académico.
- Control de bancos (SALDOS).
- Cuentas por cobrar.
- Manejo de cuentas pendientes o saldos por cobrar.
- Cuentas corrientes, patronal y de trabajadores.
- Sistema de estadística agropecuaria.
- Censos agropecuarios.

1.8 TIPO DE SOFTWARES UTILIZADO EN LAS EMPRESAS DONDE SE PASO LA ENCUESTA

- COBOL 85 VS WANG
- COBOL 74 VS WANG
- MS-COBOL 2.20
- RM-COBOL 1.12
- VS-7110 WANG
- RM-COBOL 2.1
- Compilador WANG-VS
- Compiladores todos los que dispone el sistema operativo
- Aplicadas al compilador MS-COBOL 2.20., con opciones de ejecución de programas y funciones de versiones revisadas que otros no lo hacen.
- VAX - COBOL ,ITX - COBOL, RM 85 COBOL
- NCR V 8595 (MAINFRAME)
- IBM DOS/VS COBOL versión 3.0
- Manual de errores para COBOL 85/74
- Manual de comandos para COBOL 85/74

1.9 SUGERENCIAS DADAS POR LAS PERSONAS ENTREVISTADAS EN LAS EMPRESAS DONDE SE PASO LA ENCUESTA, ENTORNO AL DISEÑO DE UN EDITOR QUE DE FACILIDADES PARA LA CODIFICACION DE PROGRAMAS EN COBOL.

- Facilidad para la gestión de bloques.

- Facilidad para exportar e importar párrafos.

- Permita trabajar con varios ficheros a la vez (2 como mínimo).

- Incorporar el compilador (que tan ventajoso es).

- Un help (on line) con el formato de los comandos, explicación y un ejemplo.

- Un help on line con los errores más conocidos en la ejecución y compilación.

- Inteligencia (indentación automática).

- Poder mantener el editor en memoria a fin de facilitar la edición despues de compilar.

- Generalmente para ejecutar un programa se hace lo siguiente: a) compilar; b) link, ambas deberán hacerse juntas, de forma transparente al usuario.

- Incluir palabras reservadas en mi código fuente presionando una secuencia de teclas.

- Facilidad de generar código fuente que sirve como esqueleto básico de un programa.

- Generar programas fuentes de mantenimiento para un archivo dado.

- Que posea edición de archivos con importación y exportación de bloque en archivos externos.

- Que incluyera el compilador y debugger para programas COBOL.
- Que incluyera una comunicación entre los diferentes utilitarios al sistema.
- Que permita extraer un rango de líneas de otro programa y las inserte en el programa que estamos codificando.
- La estructura de un programa en COBOL debería ser tomada en cuenta.
- Pensar en todas las exigencias que pide el lenguaje COBOL, pero no solamente el lenguaje sino también en la facilidad de manejar el editor.
- Que posea funciones de búsqueda de texto, para renumerar líneas, compilación dentro del editor dando errores sin salirse de la sesión, que cree el texto en disco sin salirse de la sesión, que reemplace el texto sin salirse de la sesión, que tenga menús con facilidad de manejo, para modificar bloque, cambiar alguna palabra o frase por otra, inserción de línea con enumerado automático, eliminación de líneas, que muestre la posición donde se encuentre el cursor, etcétera.
- Cargar en memoria, menús de ayuda, corrección de sintaxis, que maneje la estructura de COBOL.
- Un editor que guarde automáticamente la estructura propia del lenguaje.
- Tabla de códigos ASCII, compilador adicionado y facilidad en tabulaciones.
- Tener un controlador de línea.

- *Que posea un diseñador de pantalla.*
- *Que maneje de forma interactiva las bases de datos.*
- *Que posea la facilidad de copia de sentencias que se usan con frecuencia al codificar programas en COBOL.*
- *Incluir en el editor, áreas de trabajo independientes para cada programador, con un PASSWORD de acceso personal.*
- *Que el editor sea fácil de manejar, que posea manejo de ventanas, es decir, abrir varias ventanas para cada programa dentro de la pantalla y además otros programas dentro de la memoria.*
- *Que el editor contenga todas las herramientas para un editor llamado comúnmente TURBO, es decir que en el editor lleve consigo el compilador y una mezcla de todas las ventajas de los editores investigados.*

1.10 EDITORES DE MAYOR USO EN LAS EMPRESAS ENCUESTADAS

EDITOR Q.

VENTAJAS:

- Da facilidad para movimiento de bloques (podría ser a nivel interno o externo).
- Maneja más de un fichero.
- Exportar e importar ficheros o partes.
- Residente en memoria.
- Ofrece comandos interesantes.

DESVENTAJAS:

- Indentación automática.
- Falta de tabla de códigos.

EDITOR SideKick

VENTAJAS:

- Es rápido y pequeño.
- Residente en memoria.
- Posee tabla de códigos ASCII, calendario, calculadora.
- Versatilidad de comandos.

DESVENTAJAS:

- No maneja más de un archivo.
- Origina conflicto en memoria, debido a que está residente, no posee descarga de la memoria utilizada.
- No indentación automática.

EDITOR WANG

VENTAJAS:

- *Compilar, corregir y ejecutar dentro del editor.*
- *Facilidad de edición de texto de otros lenguajes (BASIC, C, COBOL, ASSEMBLER, FORTRAN, PL/I, RPG II).*
- *Facilidad de uso a través de menús.*

DESVENTAJAS:

- *Sólo se puede editar y compilar un archivo a la vez.*
- *No crea copias de resguardo.*
- *No posee macros.*

1.11 LISTADO DE EMPRESAS CONSULTADAS, LAS CUALES NO APORTARON INFORMACION UTIL AL PROYECTOS.

QUEVEDO SANCHEZ ASOCIADOS.

ALAN SOFTWARE S. A.

CORPAC.

MACORP, S. A. DE C. V.

SISTEMAS ASOCIADOS.

SERVICIOS Y PROCESOS DE COMPUTOS.

NCR CORPORATION.

FM ASOCIADOS.

ANGLOSAL.

ANEXO #2

Listado de funciones predefinidas en C, que se utilizaron en la fase de programación (diseño físico) del FCEDIT.

bioskey()	*isprintf()*
cprintf()	*main()*
exit()	*printf()*
farfree()	*puttext()*
farmalloc()	*strcpy()*
fclose()	*strlen()*
feof()	*strcmp()*
fgets()	*system()*
fopen	*sizeof()*
fputs()	*_setcursortype()*
gets()	*textcolor()*
getch()	*textbackgraund()*
gettext()	*tolower()*

ANEXO #3

DISQUETES

Los disquetes con el software se describen en la siguiente tabla, los cuales por acuerdo de los responsables del proyecto fueron entregados a la coordinación del área de computación, de la escuela de matamática.

DISQUETE	CONTENIDO
1	Biblioteca de los archivos de programas fuente del FCEDIT.
2	Disquete del FCEDIT, contiene los programas ejecutable (fc.exe) y de soporte (estruct.cbl, palabras.cbl) a las aplicaciones en COBOL.
3	Disquete del compilador COBOL.

GLOSARIO

Abortar: Terminar un proceso antes de que concluya.

Acceso: Llamar a un función.

Almacenar: Escribir un fichero en el disco.

ASCII: Código Normalizado Americano para Intercambio de Información.

Atributo: Cualidad o característica asignada a un caracter.

AYUDA: Juego de pantallas de ayuda que se activan con la tecla de función <F1>.

Asignación Dinámica de Memoria: Asignación de memoria para las componentes individuales de una estructura dinámica de datos al tiempo que estas son creadas durante la ejecución del programa en vez de hacer la asignación durante la compilación del mismo.

Backups: Hacer copia de seguridad en disco, del fichero en memoria.

Buffer: Bloque de memoria (temporal) asignado para entrada y salida.

Bloque (de texto): Texto consecutivo que puede operarse con él como una entidad única, moviéndolo, copiándolo o borrándolo.

Borrar: Eliminar texto.

Búsqueda: Moverse directamente sobre un fichero hasta una cadena específica de texto.

Caracter: Cualquier letra, número o símbolo tecleado desde el

teclado alfanumérico.

Cadena de caracteres: Secuencia de caracteres.

Código fuente: Código de lenguaje de alto nivel que usa el programador para escribir las instrucciones.

Codificación: Conversión de los diagramas de flujo de programa en lenguaje utilizado por la computadora.

Código ASCII: Código binario de 7 bits con 128 caracteres que incluyen las teclas del alfabeto (may/min), los 10 numerales, códigos de control no imprimibles y signos de puntuación.

Código mnemotécnico: Conjunto de símbolos que dan la facilidad de recordar.

Compilador: Programa que sirve para traducir un lenguaje de programación simbólico más alto a un lenguaje de máquina que sea comprensible para el procesador.

Comando: Es una instrucción dada por el operador que puede ejecutarse de inmediato.

Cursor: Puntero sobre la pantalla que indica la posición que se verá afectada al activar el teclado.

Dato: Una palabra de computadora: una unidad de información.

Datos: Números, letras, símbolos o hechos que describen un objeto, idea, condición, situación u otro factor. Elementos básicos de la información que pueden ser procesados o producidos por una computadora.

Delimitar: Establecer los límites.

Depurar: Identificar y eliminar un error.

Despliegue: *Representación visible de información en una pantalla de consola.*

Diagrama: *Representación gráfica de las soluciones presentadas en una forma abstracta o simbólica.*

Disco de doble densidad: *Técnica de grabación o registro utilizada para duplicar la cantidad de datos almacenados en un disco magnético.*

Disco duro: *Medio de almacenamiento masivo en disco que utiliza un disco de material rígido sobre el cual está depositado el medio magnético que almacena los datos.*

DOS: *Siglas del programa sistema operativo de disco.*

Editar: *Modificar el contenido o formato de un documento.*

Ejecución: *Realización de una instrucción u operación por la computadora.*

Ejecutar: *Proceso de computadora que interpreta una instrucción de computadora y realiza las operaciones especificadas por la instrucción.*

Escribir: *Transmitir datos de la memoria de la computadora a un medio externo de almacenamiento como disco duro o disco flexible.*

Error: *Discrepancia entre una condición o valor calculado y la condición real especificada, o valor correcto teórico.*

Hardware: *Las componentes mecánicas y electrónicas de un sistema computador.*

HIPO: *Jerarquía-Entrada-Proceso-Salida.*

Impresión: *El documento se imprime sobre el papel.*

Indentación: *Posicionar un margen temporal que se reflejará al imprimirlo, e idealmente también en la pantalla cuando se alinee el texto.*

Insertar: *Introducir nuevo texto entre el texto previamente introducido sin borrar éste.*

Identificación: *Número o nombre de código que etiqueta exclusivamente una unidad de información.*

Lenguaje C: *Lenguaje de programación estructurado de propósito general.*

Lenguaje COBOL: *Lenguaje orientado a procedimientos, que se diseñó específicamente para satisfacer las necesidades asociadas con el procesamiento de datos administrativos.*

Menú: *Listado de posibles opciones de comandos.*

Mover: *Llevar un bloque de texto a una nueva posición.*

Memoria de acceso aleatorio (RAM): *Arreglo de almacenamiento interno de datos a partir del cual puede recuperarse información a una velocidad que es relativamente independiente de la localización de la información almacenada.*

Mnemotécnico: *Uso de un símbolo abreviado para ayudar a la memoria humana.*

Portabilidad: *Propiedad del software (programas) que le permite ser usado en diferentes computadoras.*

Programa: *Conjunto detallado y explícito de instrucciones de computadora para realizar algún trabajo.*

Programa fuente: Aquella forma que tiene el programa tal como lo escribe el programador. Por lo general se escribe en lenguaje de alto nivel.

Programación estructurada: Un programa estructurado es como un conjunto de notas escritas en forma de esquema.

Programador: Persona que prepara y planea la secuencias de actividades que una computadora debe realizar a fin de resolver un problema.

Puntero: Registros en una UCP (Unidad Central de Procesamiento) que contienen direcciones de memoria o valores de compensación que apuntan a datos, subrutinas, sectores de disco, etc.

Seudocódigo: Descripción con palabras o fórmulas de un algoritmo o programa; resumen general de las funciones de un programa que se utiliza con un diagrama de flujo para diseñar o documentar un programa.

Sintaxis: Conjunto de reglas que se aplican para construir expresiones o frases válidas y significativas en un lenguaje.

Simulación: Técnica para demostrar o imitar situaciones físicas empleando computadoras.

Sistema operativo: Conjunto de programas y rutinas que guia una computadora en el cumplimiento de sus tareas, auxilian a los programas con funciones de apoyo e incrementan la utilidad del hardware.

Software: Programas y rutinas (instrucciones secuenciales)

que indican a la computadora qué hacer y cuándo hacerlo.

Tabulador: *Punto que indica el limite de una columna de textos con múltiples columnas.*

Unidad de disco: *Dispositivo periférico electromecánico de entrada/salida que alberga y hace girar un disco duro o flexible para almacenamiento auxiliar de datos.*

ANEXO #4

Descripción de la estructura de datos utilizada para el manejo de bloques.

La estructura de datos utilizada en la gestión de bloques, es una lista doblemente enlazada, tal como se muestra en la figura-4.1,

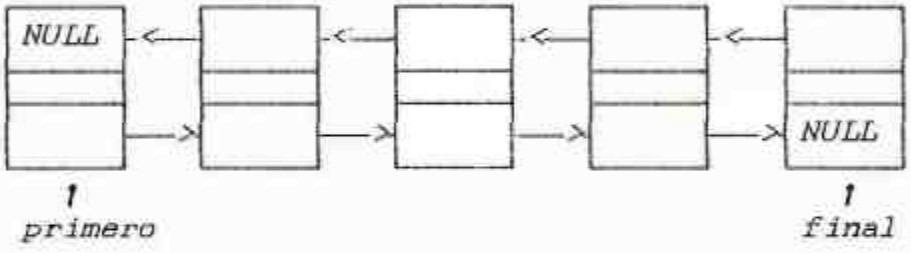


figura-4.1

donde: **primero** y **final** son punteros externos que referencian el primero y el último nodo de la lista. Esta estructura almacena en cada uno de sus nodos, información que se va a mover, copiar, borrar o restaurar en el texto que se maneja actualmente en memoria, a través de una lista doblemente enlazada con cabecera.

Supóngase que el archivo que se maneja actualmente en memoria, se aloja en una estructura como la que se muestra en la figura-4.2,

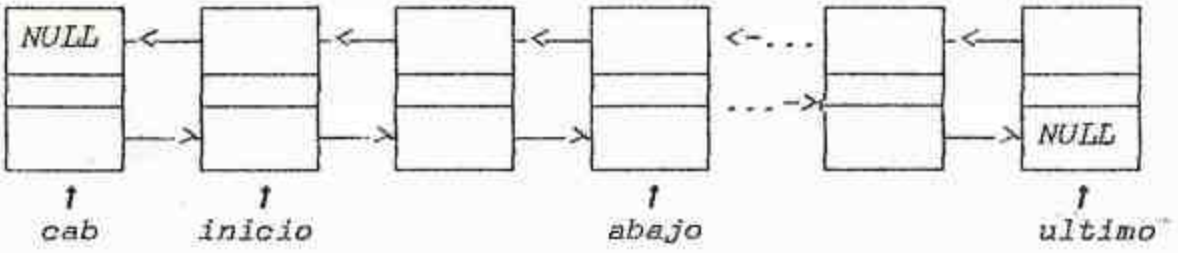


figura-4.2

donde: cab, inicio, abajo y ultimo son punteros externos que permiten efectuar cualquier operación en el archivo.

Esta estructura y los punteros que permiten manipularla se explicó en la sección 3.1.4, a la que se identificó con el nombre de LISTA.

Los cambios de dirección en LISTA, dependen directamente de los movimientos del cursor en la pantalla.

Supóngase que se ejecuta la función marcar bloque y que se marcan tres líneas de texto (de arriba hacia abajo), después de esta operación el estado de LISTA es el siguiente:

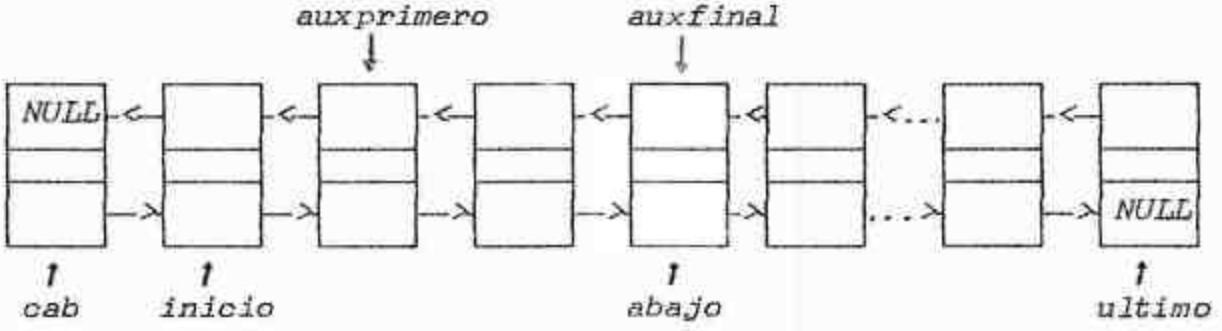


figura-4.3

donde: auxprimero y auxfinal son punteros externos que referencian el primero y el último nodo (línea de texto)

marcado.

Bajo este supuesto, pueden ejecutarse cualesquiera de las siguientes funciones:

- ESC. Si se presiona la tecla ESC, se desactiva la función ***marcar*** *bloque y los punteros* ***auxprimero*** *y* ***auxfinal*** *se inicializan con la dirección nula (NULL).*

- Copiar. Si se ejecuta la función ***copiar*** *bloque, se recorre la lista desde el nodo referenciado por* ***auxprimero*** *hasta el nodo referenciado por* ***auxfinal,*** *a la vez que se hacen asignaciones de memoria para crear simultáneamente una lista doblemente enlazada (manipulada por los punteros,* ***primero y final),*** *en la que se almacena (temporalmente) una copia de la información contenida en cada nodo referenciado por* ***auxprimero;*** *esto es:*

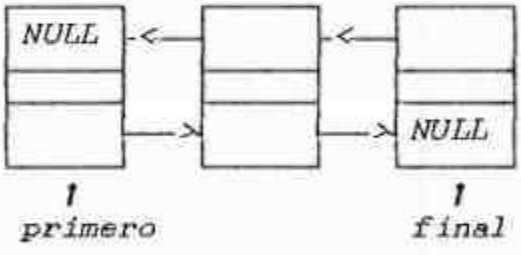


figura-4.4

y los punteros ***auxprimero*** *y* ***auxfinal*** *se inicializan con la dirección nula (NULL). Posteriormente se desplaza el cursor hasta la posición donde se quiere copiar el bloque marcado.*

Supóngase que los desplazamientos del cursor en la pantalla, producen los siguientes cambios en LISTA.

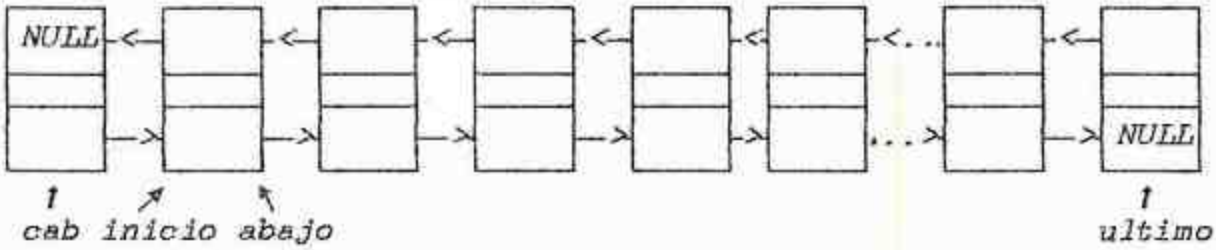


figura-4.5

lo que indica que la copia se efectuará al inicio del archivo y se ejecuta con sólo presionar la tecla ENTER.

La figura-4.6 muestra lógicamente como se realiza esta operación (inserción de la lista de la figura-4.4 en la lista de la figura-4.5).

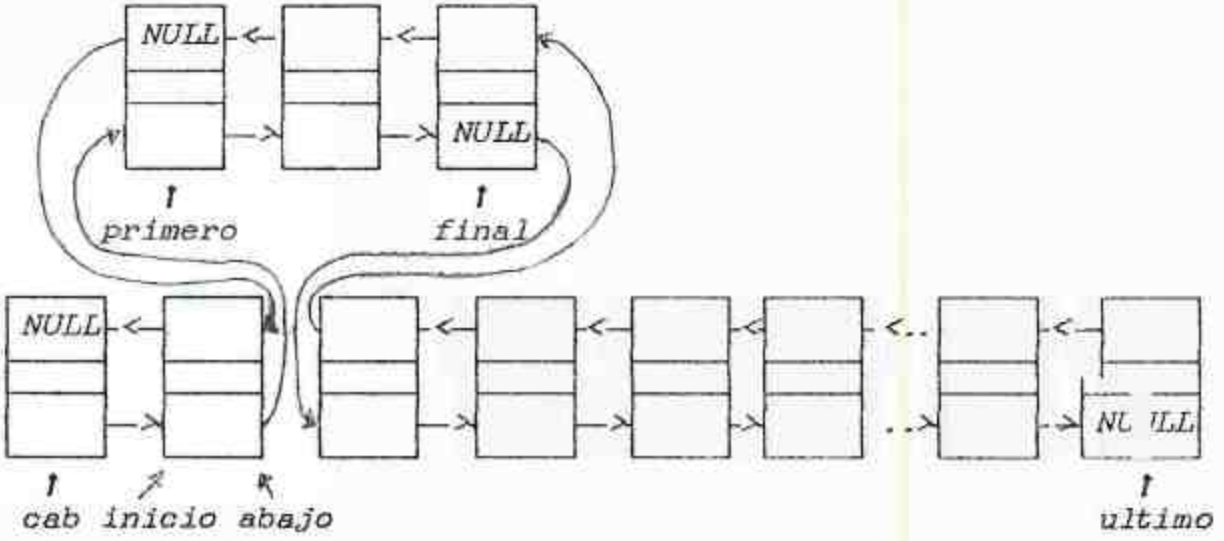


figura-4.6

Finalmente los punteros primero y final se inicializan con la dirección nula (NULL).

- Mover. Si se ejecuta la función mover bloque, los punteros primero y final apuntan a los nodos referenciados por auxprimero y auxfinal (ver figura-4.3), inicializando estos

últimos con la dirección nula (NULL). Posteriormente se desplaza el cursor hasta la posición donde ha de moverse el bloque marcado. Supóngase que el estado actual de LISTA es el siguiente:

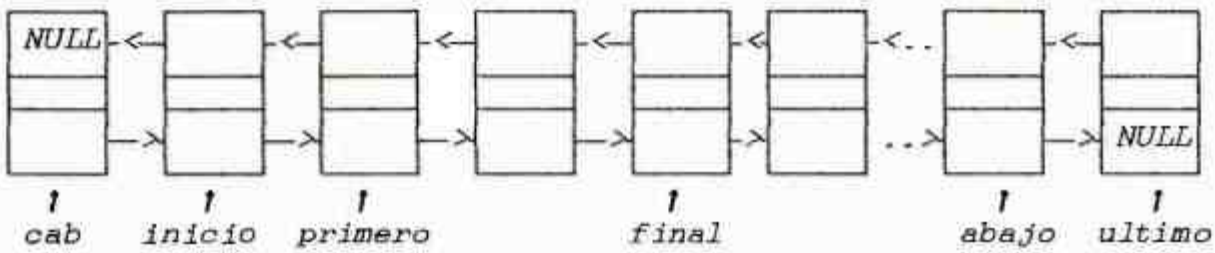


figura-4.7

el puntero abajo referencia la dirección actual.

La figura-4.8 muestra la LISTA luego de eliminar el bloque marcado de la posición original.

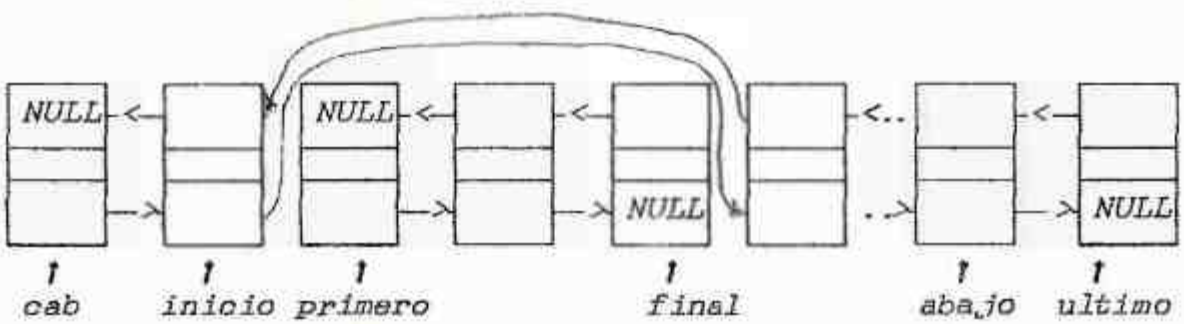


figura-4.8

Luego la lista manejada por los punteros primero y final es insertada en la dirección referenciada por abajo. Este proceso es el mismo que se presenta en la figura-4.6, finalmente primero y final se inicializan con la dirección nula (NULL).

- Borrar. Si se ejecuta la función borrar bloque, y se parte

del mismo supuesto que en la función anterior, el estado de LISTA es el que se presenta en la figura-4.8.

La lista manejada por los punteros **primero** y **final** permanece en memoria hasta que se ejecuten otros procesos en los que tenga que ser eliminada del entorno actual.

- Restaurar. La función **restaurar** bloque es exitosa sólo en el caso que la operación previa haya sido un borrado; esta función inserta en LISTA (en la dirección referenciada por el puntero **abajo**), la lista que manejan los punteros **primero** y **final**, luego estos últimos se inicializan con la dirección nula (NULL).