Ej. 12

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

PROCESADOR DE PALABRAS EN ESPAÑOL PARA MICROCOMPUTADORA

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N

MARIA DEL CARMEN GODOY COVARRUBIAS

CARLOS RODRIGUEZ LUCATERO

DANIEL ZAMORA OLVERA

Director: M. en C. Ing. Efrain Pardo

Ciudad Universitaria **Julio 1987**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Contenido

1. Introducción

1.1 ¿ Qué ha sido un procesador de palabras ?

1.2 Aceptación de los procesadores de palabras.

1 INTRODUCCION

1.1 ¿Qué ha sido un procesador de palabras?

El procesamiento de palabras es sólo uno de los trabajos que las computadoras o microcomputadoras hacen actualmente. Las funciones elementales siguen siendo procesos numéricos, invisibles para el ususario, pues lo que éste tiene que hacer, es teclear palabras de un texto que simultáneamente visualiza en una pantalla.

A lo que nos referimos cuando hablamos de un procesador de palabras, es a un conjunto de programas de computadora que nos dan la facilidad de crear, corregir, mejorar, guardar o pasar a papel cualquier documento. Se nos permite también borrar parte del texto, incluir páginas enteras de información, y algunas otras funciones más complicadas como alinear todas las palabras a un margen, incluir gráficas o corregir ortografía, pero de una manera sencilla.

Otra ventaja que nos ofrece este tipo de sistema es que se disminuye el desperdicio de papel, y la creatividad, productividad y claridad de la personas que lo usan se incrementa.

Ahora bien, ¿quiénes son las personas que usan un procesador de palabras?; pues cualquier persona que tenga relación con documentos, textos, notas, correspondencia, etc., pero en especial es útil para escritores, o en oficinas y editoriales.

Dado que ningún procesador de palabras reune todas las características deseadas, generalmente lo que sucede es que se usa un procesador para introducir texto a la computadora fácilmente, y para cuando se necesita dar forma, usar columnas, o utilizar alguna otra función más sofisticada, se usa otro.

Desarrollo

A continuación se incluyen algunos datos históricos para aquellos con un poco de curiosidad, como nosotros.

Desde un principio se ha comparado a los procesadores de palabras con las máquinas de escribir, pero lo único que tienen en común es el teclado. A pesar de ello, incluímos algo de la historia de estas máquinas por ser parte de su pasado.

Un procesador de palabras puede hacer todo lo que una máquina de escribir, claro, pero más rápido, mejor, y de una manera más eficiente.

Aún así, el primer acercamiento hacia los procesadores de palabras es por medio de las máquinas de escribir (pues todo mundo las conoce). Y si recordamos un poco más en el pasado... En el siglo XV se inventó la imprenta, uno de los eventos más importantes en la historia, y con él empezó la difusión de información en forma escrita.

Aunque esto solucionaba en gran parte los problemas de transmisión de información entre las personas, el escritor seguía con sus mismos problemas.

Un poco después, en 1867, se inventó la máquina de escribir, y ésto se acompañó de algunas implicaciones sociales como la creación del oficio de mecanógrafa.

Las cosas siguieron así por casi cien años, hasta que las máquinas de escribir eléctricas se inventaron, mejorando en mucho a las antiguas máquinas mecánicas. Pero la escencia de las máquinas de escribir no cambió; si se cometían errores, la corrección seguía siendo igual que antes.

Ultimamente se puede aplicar un líquido sobre el papel en lugar de borrar, pero de todas maneras, el corregir es mucho más complicado si el trabajo incluye copias. Algunas veces, el aplicar el líquido no es suficiente, y ya lo habrán sufrido, pues las correcciones usan más espacio que las palabras originales, y se necesita teclear toda la página de nuevo.

En 1964 se dió, por la compañía I.B.M., el primer paso hacia lo que se llama "procesamiento de palabras". Este consistía en una grabadora de cinta magnética que se añadía a una máquina eléctrica. Cinco años después, se mejoró el sistema cambiando la cinta por tarjetas, también magnéticas, donde cada tarjeta representaba una página, facilitando el manejo del documento.

Ya acercándonos más a lo que sería nuestro tiempo, en la década de los setentas, las compañías "Lexitron" y "3M" presentaron al público lo que se conoció como un "procesador de palabras". Incluían en él una pantalla que permitía a los

usuarios ver y cambiar texto directamente sobre ella, usando el papel sólo como un paso final. En 1973, la compañía "Vydec" presentó una máquina similar, usando un disco flexible magnético, lo que incrementó la velocidad de procesamiento. Este tipo de máquinas son conocidas como "procesadores de palabras dedicados", relativamente fáciles de usar, rápidas y muy buenas para manejar diferentes tipos de documentos. Ya que nuestra area de acción son las microcomputadoras, podemos incluir que las micros se fueron desarrollando paralelamente a ese tipo de máquinas, solamente que ahora son más comunes las primeras que los procesadores dedicados.

Para las grandes computadoras también se empezaron a desarrollar programas casi al mismo tiempo que los procesadores dedicados, pero el resultado en las primeras no se obtenia de forma tan sencilla, ni tan rápida. Los manuales y la documentación en general no eran suficientes, así que la mayoría de los usuarios regresaban a sus máquinas de escribir.

Finalmente las microcomputadoras se cruzaron con el camino de los procesadores de palabras dedicados (por el tiempo en que empezábamos la licenciatura). Se cruzaron en el camino porque las microcomputadoras dieron origen a editores sencillos de texto. Al evolucionar, estos editores se fueron pareciendo a los procesadores dedicados.

Ya en nuestros dias, un conjunto de programas para el procesamiento de palabras, son un módulo de los paquetes del tipo **software integrado**. En los sistemas de software integrado se tienen algunas caracteristicas, como que generalmente los otros módulos son de la misma calidad, se manejan similarmente y se trata de que sean una familia de programas. En ocasiones se incluye un módulo de gráficas u otro tipo de representación, que son de gran utilidad para ejecutivos y directores de empresas (o de tesis).

Otra corriente que ha surgido es el **procesamiento de ideas**. Aquí se unen un organizador de ideas, en donde cada una se desarrolla como un documento, y un administrador de base de datos para la organización de éstas.

El procesamiento de palabras es una de las operaciones más utilizadas en las computadoras personales. Cualquier persona que tenga una computadora, y la use, quisiera tener un procesador de palabras. En un principio, los costos no permitieron que esta herramienta fuera de uso común, pero ahora no es solo su precio lo que nos acerca a ellas, sino que también su facilidad de uso y su versatilidad. Desgraciadamente, solo muy pocos programas pueden manejar las caracteristicas especiales del español y lo hacen dificilmente, o no están terminados, o son de baja calidad.

Por último, al analizar el desarrollo de los procesadores de palabras, se espera que en algún momento éstos se unan con la inteligencia artificial, y que el procesamiento de palabras influya también en la forma de expresarse. Tal vez, hasta nos den comentarios sobre nuestros trabajos escritos, y se puedan integrar reconocedores de voz y al fin librarnos de teclear. Para entonces esperamos que el correo electrónico y las bases de datos sean comunes.

1.2 Aceptación de los procesadores de palabras.

La aceptación de un producto es uno de los indicadores más significativos de la calidad del mismo, aunque otros factores sean su publicidad o su suerte. Un factor decisivo es el ser innovador, como lo fueron en su momento "APPLE WRITER" o "WORD STAR" (procesadores de los que se encuentra más información adelante).

En el campo de las microcomputadoras, un hecho decisivo para que ciertas marcas llevaran la delantera, fue la introducción de un medio de almacenamiento de información masivo, pero más rápido y confiable que las grabadoras de cassettes, conocido ahora como diskettes.

Para aquellos que usan un procesador de palabras, éste les representa un ahorro considerable en horas de escritura, y un cambio en su proceso creativo. El problema reside en elegir uno adecuado a sus necesidades, adaptable a su computadora, y a un costo razonable. El primer paso que se recomienda es definir muy bien las actividades a realizar, después acudir a un distribuidor y probar varios programas hasta encontrar uno adecuado.

Los procesadores de palabras más sencillos se usarían en actividades como escribir cartas o memorandums de poca longitud. Si quisiéramos tener correo electrónico, necesitaríamos paquetes accesorios de comunicación. Pero si se trata de tener cientos de reportes o documentos con mucha calidad, se requiere de un procesador de palabras que soporte una gran variedad de impresoras. La característica de facilidad de comunicación entre otros sistemas es muy importante, pues una de las nuevas tendencias es el de hacer programas integrados o acoplados, pero también porque así se aprovechan las capacidades de otros paquetes.

Entonces llegamos a la conclusión de que no hay un procesador de palabras que cubra totalmente las necesidades del usuario, y por ello algunos procesadores en función al mercado hacia el cual están orientados llegan a ser best-sellers en Norteamérica y empiezan a extenderse hacia latinoamérica. En general, se ha detectado que los procesadores más versátiles son los más costosos y difíciles de usar, provocando una disyuntiva entre versatilidad, costo y facilidad de uso.

2 Análisis

2 ANALISIS

2.1 Procesadores de Palabras Especificos

2.1.1 Introducción

A continuación se presenta una serie de reportes, resultado del estudio de procesadores de palabras existentes. Nuestras fuentes fueron revistas y experiencias personales. Este punto es importante en especial, porque a partir de este estudio se identificaron las caracteristicas principales y las limitaciones de un procesador de palabras.

Probablemente algunos datos parecerán imaginados, no lo son, lo que pasa es que este estudio se hizo hace dos años. Se incluyen datos como el lenguaje en que fueron programados, la computadora para la cual se hicieron y facilidad de uso.

2.1.2 Descripciones de algunos Procesadores

Nombre : The Executive Secretary
Tipo : procesador de palabras
Configuración básica: Apple II+ con Applesoft en ROM, 48 K RAM, lectora de discos, adaptador para letras minusculas y/o tarjeta de 80 columnas.
Lenguaje : Applesoft BASIC y lenguaje de máquina.

Características :

Comandos.- sus comandos son sencillos y las letras de las teclas de función están relacionadas con la función a realizar. Algunas de sus funciones son solo accesibles a nivel de menú de funciones. Cuenta con comandos para realizar funciones de bases de datos y generación de correspondencia.

Facilidad de uso.- es un paquete fácil de usar pues sus comandos de funciones son consistentes con la función que van a realizar por lo cual, también es fácil de aprender.

Interfase de Usuario.- su interfase de usuario es a base de menús lo cual llega causar descontrol al usuario al estar cambiando de una pantalla de trabajo a una pantalla de menú de funciones.

Documentación.- cuenta con un buen manual.

Conclusiones.- un aspecto interesante en este procesador es la comunicabilidad con otros paquetes comerciales y las funciones adicionales de bases de datos y generador de correspondencia.

Nombre : Easy Writer
Tipo : procesador de palabras
Configuración básica: IBM PC, 64 K RAM pantalla de 80 columnas, lectora de discos, sistema operativo MS-DOS

Lenguaje : lenguaje de máquina.
Productor : Information Unlimited Software, INC.

Características :

Comandos.- muchos de sus comandos se piden solamente desde menú y las teclas asignadas a las funciones no están relacionadas con la función a realizar.

Facilidad de uso.- dado que las teclas de comando no fueron asignadas con cuidado, este paquete resulta difícil de usar y de aprender, pues además de no ser mnemónicas con respecto a la función, son inconsistentes.

Interfase de Usuario.- se lleva a cabo por medio de menús, pero es navegacional, por lo cual se pierde la referencia de lo que se está haciendo, pues para realizar una función se cambia a una pantalla de menú.

Documentación.- cuenta con un manual amplio, que es necesario, en ocaciones las cosas son tan complicadas que requieren de una práctica extensiva además de éste. También cuenta con una tarjeta de referencia rápida.

Conclusiones.- es un paquete muy malo y antiguo (1982). Lo que podemos aprender de él, es que tiene características que debemos evitar en nuestro procesador de palabras. Un comentario importante es que este programa era muy eficiente y bueno en Apple, es decir que fue al pasarlo a IBM que el programa decae, pues no considera el potencial de esta máquina.

Nombre : THOR
Tipo : Paquete integrado.
Configuración básica: IBM PC, 64 K RAM pantalla de 80 columnas, lectora de discos, sistema operativo MS-DOS

Características :

Comandos.- los nombres de los comandos que utiliza tienen más significado para los usuarios nuevos, por ejemplo "pensamiento nuevo ", "recolección", "olvido" en lugar de "Apertura de archivo", "merge", "borrar".

Facilidad de uso.- sus pantallas de menús son muy representativas y ayudan al usuario cuando esta trabajando. Esta característica aunada al significado claro de sus comandos lo hacen un paquete fácil de aprender y usar.

Interfase de usuario.- en cuanto a este aspecto se puede decir que trabaja por medio de menús y pantallas de ayuda que sirven mucho al usuario. Maneja 16 colores en la pantalla lo cual le da más vista a la interfase.

Documentación.- la documentación es muy deficiente, su manual está muy mal organizado. A pesar de que cuenta con una gran variedad de funciones, no se pueden aprovechar porque no se cuenta con información confiable.

Conclusiones.- es deseable que nuestro procesador de palabras sea fácil de aprender. Este sistema se ayuda de nombres que no son muy técnicos, de menús y de ayudas en pantallas muy bien presentadas, lo cual es también deseable.

Otra enseñanza que nos deja este paquete es que no debemos entregar un paquete sin manuales.

Nombre : Word Mark 4.0
Tipo : Software integrado
Configuración básica: IBM PC ó compatibles, 256 K RAM, 2 unidades de disco.

Características :

Comandos.- sus funciones se realizan por medio de comandos que son de caracter y las funciones para las cuales no alcanzo el teclado se pueden invocar en menús bastante explícitos. Los comandos son consistentes con la función que se va a realizar. Contempla el uso de comandos para correo electrónico y corrección de ortogrrafía.

Facilidad de uso.- es un paquete difícil de usar pues tiene una variedad tan grande de comandos y funciones que difícilmente se pueden aprender todos.

Interfase de Usuario.- utiliza menús y líneas de ayuda para cada función.

Documentación.- su manual es deficiente.

Conclusiones.- tiene características que son un ejemplo a seguir para cualquier procesador de palabras, como lo es la confiabilidad; pero también cuenta con algunas características indeseables como lo son la dificultad de uso y la falta de una documentación de calidad.

Nombre : Display Writer II
Tipo : Procesador de palabras
Configuración básica: IBM PC, 192 K RAM, 2 lectoras de disco, Sistema operativo MS-DOS 2.1

Características:

Comandos.- todos se realizan a través de menús.

Facilidad de uso.- es un paquete de fácil uso y de rápido aprendizaje.

Interfase de Usuario.- su presentación es agradable y contiene menús que sirven de guía al usar cada operación; en las tres primeras y en la última líneas de la pantalla nos da información útil, ayudas y mensajes de error.

Conclusiones.- este es un paquete de muy alta calidad, como lo que se espera de IBM, y debe de servirnos de ejemplo en cuanto a la calidad del producto; pero no como base, ya que este paquete consume mucha memoria.

2.1.3 EXPERIENCIAS PERSONALES

Nombre : Apple Writer //e (Apple Computer, Inc.)
Tipo : Procesador de Palabras
Configuración básica: Apple IIe 128 K RAM
Sistema operativo DOS 3.3

Características:

Comandos.- Realiza la mayoría de sus funciones por medio de teclas de control y las que no puede realizar de esta forma las hace por medio de un lenguaje de comandos propios, llamado "WPL" (word processor language). Este lenguaje es un conjunto de 20 instrucciones. Los comandos son consistentes con las funciones a realizar. Algunas funciones de uso poco frecuente o del sistema operativo se realizan a través de 2 menús secundarios.

Facilidad de Uso.- Es un procesador de palabras muy sencillo de utilizar debido a la consistencia de sus comandos tanto de control como sus instrucciones de WPL. A pesar de que algunas funciones tengan que ser realizadas a través de un menú secundario, esto no descontrola al usuario, pues dichas funciones están agrupadas de tal forma que sean operaciones que se realicen cuando se va a terminar de trabajar.

Interfase de usuario.- Cuenta con dos menús auxiliares y para cada comando de control existe un mensaje guía en la parte inferior de la pantalla.

Documentación.- Cuenta con un manual de referencia del procesador de palabras y un manual de lenguaje de comandos.

Conclusiones.- Las características de este procesador de palabras son muy buenas, tomando en cuenta las limitaciones del hardware para el cual fue diseñado; pero la principal de ellas es la incorporación de un lenguaje de comandos que le da una gran flexibilidad y poderío al sistema.

Nombre : Jack 2
Tipo : Software integrado
Configuración básica: IBM PC ó compatibles, 128 K RAM, 2 unidades de disco.
Lenguaje : Pascal USDC
Productor : Business Solutions Inc. Kings Park N.Y.

Características:

Comandos.- Sus funciones se accesan a través de menús jerárquicos y las funciones son consistentes en todos los niveles. Utiliza las teclas de funciones para definir macros.

Facilidad de uso.- Debido a la consistencia en la forma de invocar a las funciones en todos los niveles del paquete (nivel de disco, nivel de directorio, nivel de archivos, nivel de texto o registros de un archivo) este paquete es fácil de aprender.

Interfase de usuario.- En cuanto a su interfase de usuario, se puede decir que es revolucionaria, ya que introduce el concepto de archiveros para cargar documentos, el cual corresponde a un modelo real y común para el usuario. Es el primer paquete en el mercado que comienza a utilizar las líneas de ayuda o líneas "hint" para evitar el tener que leer pantallas largas de ayudas al usuario.

Documentación.- Cuenta con manual de referencia técnica, manual para usuario avanzado, manual para principiante, manual tutorial y un disco tutorial con lecciones muy bien planeadas.

Conclusiones.- Este es uno de los paquetes que se adelantó a su tiempo en el aspecto de la interfase de usuario. Una de sus características más importantes es el uso de líneas guía y el manejo del concepto de archiveros, que posteriormente, el paquete de software integrado llamado Apple-Works, utilizó con mucho éxito. Es un paquete digno de tomarse en cuenta para tratar de alcanzar su facilidad de uso. Otro aspecto importante fue el hecho de que a pesar de no hacer uso de los caracteres alternos de la IBM PC para desplegar letras acentuadas tiene una forma muy fina de resolver el problema, que es poniendo la letra con un color más intenso, evitando así el despliegue de caracteres de control.

Nombre : Think Tank
Tipo : Organizador de Ideas y Proc. de Palabras
Configuración Básica: Cualquier modelo
Lenguaje : Pascal

Características :

Comandos.- Todas las funciones se pueden realizar por medio de menús, que se localiza en la parte inferior de la pantalla. Entre sus comandos se encuentran los que manejan el bosquejo del trabajo, los que manejan el procesamiento de palabras y los generales, que se encargan de las funciones que relacionan al trabajo con el disco o impresora.

Facilidad de Uso.- Es un paquete extremadamente fácil de usar, ésto es porque siempre se encuentra en la pantalla información suficiente para no tener problemas al dar el siguiente comando. A pesar de no existir ayudas, los menús y sus explicaciones son suficientes.

Interfase de Usuario.- Su presentación en la pantalla es fácil de interpretar, además, las versiones que existen son muy similares, de manera que si se conoce ya una, la otra también. Si la computadora lo permite o si se puede conectar un "Mouse", el programa lo reconoce y su operación es más sencilla.

Conclusiones.- Es una herramienta muy útil para organización. Como procesador de palabras, es muy bueno también. Su punto débil es que en sus impresiones no es fácil de lograr algún formato en especial.

Nombre : Multimate
Tipo : Software integrado
Configuración básica: IBM PC ó compatibles, 256 K RAM, 2 unidades de disco.

Características:

Comandos.- Utiliza ampliamente las teclas de funciones de la IBM PC. Fuera de las funciones de edición las operaciones de parámetros de edición e impresión se realizan llenando pantallas tipo cuestionario y los módulos principales se accesan a través de menús y las demás funciones se piden con secuencias de teclas. Cuenta con comandos para correo electrónico y corrección de ortografía, librerías de frases de uso frecuente, y manejo de archivos tipo "batch". Sus comandos son muy consistentes en todas las partes del programa.

Facilidad de uso.- Tiene una gran cantidad de comandos lo cuál puede llegar a complicar el uso y aprendizaje de un programa pero para solucionar este problema, utiliza mensajes de explicación en todas sus pantallas en los márgenes superior e inferior, además cuenta con un disco tutorial, un manual de lecciones y un sistema de ayudas muy bueno.

Interfase de usuario.- Es un sistema con interfase de usuario a través de menús con línea de ayuda ó línea guía. Para las funciones secundarias que no intervienen de manera directa en la edición en pantalla, este procesador de palabras hace uso de pantallas tipo cuestionario.

Documentación.- Cuenta con manual de referencia técnico, manual para principiantes, manual para usuarios avanzados, prontuario de funciones y una plantilla.

Conclusiones.- Multimate es un procesador de palabras muy bueno, pero ésto va de acuerdo con su precio, que es muy elevado. Una característica que es preciso hacer notar, es el manejo o enfoque de páginas que le permite ahorrar espacio en memoria. Soporta un gran número de impresoras y permite dar de alta alguna impresora que no tenga en librerías de una manera muy sencilla. En general este propcesador de palabras es muy bueno.

Nombre : Framework II
Tipo : Software integrado
Configuración básica: IBM PC ó compatibles, 320 K RAM, 2 unidades de disco.

Características :

Comandos.- Utiliza ampliamente las teclas de funciones de la IBM PC.

Facilidad de uso.- Tiene una gran cantidad de comandos que pueden llegar a complicar el uso y aprendizaje de un programa; pero para solucionar este problema, utiliza mensajes de explicación en todas sus pantallas en los márgenes superior e inferior. Además cuenta con un disco tutorial, un manual de lecciones y un sistema de ayudas muy bueno.

Interfase de usuario.- Es un sistema con interfase de usuario a través de "pull-down menus" con línea de ayuda o línea guía. Cada opción puede ser obtenida mediante movimientos de cursor o por medio de la letra inicial de la función a ejecutar. Cada "pull-down menu" puede ser usado mediante la letra inicial de la opción más la tecla de <CTRL> presionadas simultáneamente. Es un paquete muy fino en su interfase de usuario, utiliza lo último en tecnología de interfase de usuario, que es el manejo de ventanas con todas las operaciones que se pueden hacer sobre ellas (mover, agrandar, desaparecer, crear, etc). El manejo del cocepto de "desktop", y el uso de "pull-down menus". A parte de que hace un manejo muy interesante de lo que son las "líneas de guías" y "líneas de status".

Conclusiones.- FRAME WORK es un paquete de software integrado muy fino que cubre casi todas las nececidades de cualquier usuario de IBM PC o compatibles. Contiene una hoja tabular, una base de datos, un módulo de graficación, un procesador de palabras y un procesador de ideas. Su interfase de usuario es una de las más "amigables" y avanzadas que puede existir en el mercado. Además, permite el despliegue de tipo de letras en la pantalla y en lo que se refiere a manejo de memoria, permite el uso del concepto de memoria virtual de tal forma que la única limitación es el tamaño del disco de datos.

2.2 ANALISIS DE HARDWARE

2.2.1 Introducción.

A continuación, se incluye un pequeño análisis de las partes físicas de las que se forman una microcomputadora. Se dice que un procesador de palabras es tan bueno como el hardware en el que funciona, y por ello la interconección de sus elementos (teclado, monitor, impresora y dispositivos de almacenamiento) es determinante en la productividad del sistema.

El teclado y el monitor son las partes más importantes para un usuario, pues se encuentra frente a ellas el mayor tiempo. El monitor debe desplegar caracteres nitidos, claros y brillantes. Un caracter está formado en la pantalla por pequeños puntos, y mientras más puntos lo formen, su imagen está mejor definida. Este conjunto de puntos es conocido como matriz de puntos, y las medidas más comunes son 6x8, 7x10 y 14x10 puntos. Otro dato acerca del monitor es el número de lineas y de columnas que caben en la pantalla. Los sistemas mejores despliegan ochenta caracteres por linea y veinticuatro o más renglones por pantalla. En un sistema del tipo que nos encontramos estudiando, seria recomendable poder ver en la pantalla toda una hoja de texto a la vez, y por ello cincuenta y cinco lineas suelen ser comunes, pero en ocaciones estorbosas. Y en cuanto al color, las estadisticas muestran que las pantallas de fósforo ámbar son más cómodas a la vista, aunque otra buena posibilidad es el fósforo verde.

El teclado, de preferencia debe ser tipo máquina de escribir, con letras grandes y ligeramente curvas. Si se trabaja a menudo con números, es recomendable que tenga un teclado numérico a la derecha del teclado regular, y como última especificación, es deseable que sea independiente o móvil, para ponerlo donde sea conveniente o más cómodo.

En cuanto a la impresora. Existen tres principales tipos de impresoras, las de matriz de puntos, las de caracteres fijos y las laser. Las impresoras laser, en el momento, son extremadamente caras, y por lo tanto no muy comunes. Los otros dos tipos son muy comunes y más accesibles. Las impresoras de matriz de puntos cuestan menos e imprimen más rápido, pero la calidad obtenida es baja. Las impresoras de caracteres fijos son más caras y lentas, pero la calidad de las páginas terminadas, es como la de las máquinas de escribir eléctricas.

De este breve análisis, es preferible una impresora de caracteres fijos por su tipo de impresión, ya que casi cualquier escrito se necesita hacer con calidad.

Ahora, cuál impresora será la mejor? Podemos escoger entre las máquinas eléctricas convertidas, las de margarita, las de esfera y las de cadena de caracteres. Las máquinas de escribir son muy lentas y requieren de mucho mantenimiento. Cuando en una impresora de margarita se usa una margarita de metal nos da resultados mejores que la plástica que es más rápida. La impresora de esfera nos da máxima velocidad (55 carac. por min.), excelente calidad, pero a un mayor costo.

Unidades de Disco. Un procesador de palabras podría trabajar con cassettes, pero para un aplicación seria, su lentitud y capacidad nos restringirían mucho. Una aplicación seria implica un texto mayor a tres páginas, donde se necesiten mover e insertar bloques grandes a velocidad mayor. Entonces, vemos que por lo menos necesitamos un disco flexible.

Las operaciones, en general, se pueden ejecutar con un solo disco, pero con dos sería más cómodo. Con dos unidades de disco se puede tener más información en línea o textos mayores, las copias o respaldos se hacen en un abrir y cerrar de ojos y el programa se desenvolvería en un ambiente mayor.

Entre los diferentes tipos de discos, los de 5 1/4" son los más comunes en computadoras personales, por su precio. Pero, si el costo no nos fuera importante, el punto para decidir sería qué tanto almacenamiento necesitamos al mismo tiempo. Podemos establecer que un "K" son 1024 letras (aproximadamente media página mecanografiada a doble espacio). En nuestro caso, los trabajos a efectuar se enfocan a resolver problemas de pequeños negocios y profesionistas que pueden trabajar desde dos páginas (o menos) a cincuenta páginas, por ello, con dos unidades de discos flexibles sería suficiente. Por supuesto, también existe la posibilidad de trabajar con discos duros, pero para la mayoría de las aplicaciones de procesamiento de palabras no es necesario.

2.2.2 Presentación de dos Microcomputadoras

En este capítulo nos restringimos a dos microcomputadoras, a la IBM PC y a la Apple, una de las razones es que sólo contábamos con este tipo de máquinas y la otra es que son las microcomputadoras más usadas y conocidas en México.

IBM PC

¿Es en verdad una computadora revolucionaria?
A primera vista, parece un producto revolucionario, sin embargo, analizándola parte a parte, ofrece todas las características que se pueden encontrar en otras computadoras personales.

La verdadera distinción o innovación de esta computadora, es la forma en que reune las características de las otras microcomputadoras, para formar un producto final, que es definitivamente una computadora personal.

Dado que I.B.M. se dedicó durante mucho tiempo a hacer perforadoras y a perfeccionar los periféricos con los cuales se comunican los capturistas con la computadora, sus pantallas y teclados son de muy alta calidad y nos ofrecen una gran variedad.

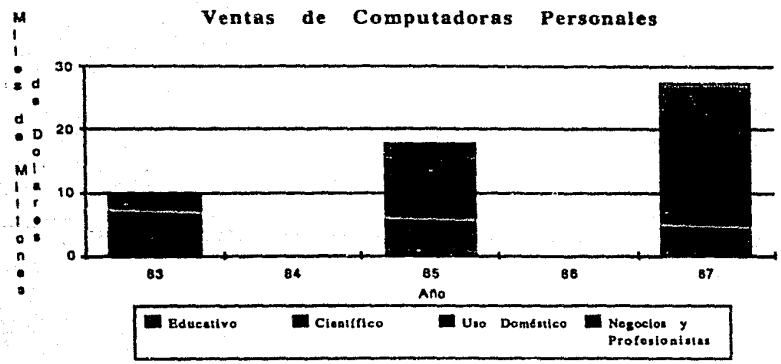
En su diseño interno, los elementos son de mejor calidad, pero el resultado final no es representativo de estas mejoras, pues no aprovecha estas características en su totalidad. Un ejemplo claro está en su manejo de memoria, el procesador en sí puede direccionar hasta un mega de memoria, pero lo hace por bloques de 64 K. Externamente, si muestra varios avances, uno de ellos es el aumento en la capacidad de almacenamiento en floppy. Su juego de caracteres es extenso, incluyendo las letras acentuadas, las eñes, y otros símbolos, pudiéndose mostrar en pantalla con varios tipos de letra.

Apple

Mientras otras marcas de computadoras llenaban el mercado con características estándares, los usuarios de Apple no tenían estas ventajas. En poco tiempo decayó. A pesar de ésto, algunos puntos muy fuertes apoyan a la Apple comparada con estas máquinas. Estos son: una amplia gama de programas, generalmente los más creativos disponibles en el mercado, un arreglo de ranuras de expansión que permite modificar la configuración del sistema de acuerdo a lo que necesite cada usuario, gráficas de alta resolución.

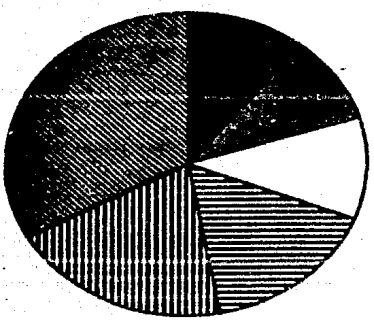
Dentro de las características estándares de la Appple //e hay:

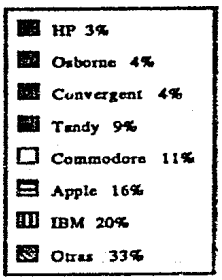
- un teclado totalmente rediseñado, que incluye sesenta y tres teclas, con autorepetición en cada tecla.
- uso de mayúsculas y minúsculas, usando la tecla de <Caps-Lock>
- tarjeta opcional de ochenta columnas.
- Una memoria de 64 K bytes de RAM, incluyendo un administrador de memoria y 16K bytes de la tarjeta de lenguaje.
- 16 K bytes ROM que contiene el BASIC de Applesoft y un sistema de software que soporta las ampliaciones hechas para la Apple.
- La capacidad de expansión de 64 K adicionales y su propia unidad de administración de memoria.
- Microprocesador 6502A de 8 bits.
- Interfases para cassette, bocina. Capacidad de conectar dispositivos para juegos, capacidad de manejar gráficas a color.
- Lenguajes disponibles: BASIC, Pascal, FORTRAN, LISP, FORTH, PILOT, COBOL, LOGO, C.


Ventas de Computadoras Personales
Miles de Millones de Dolares
30
20
10
0
83
84
85
86
87
Año
Educativo
Científico
Uso Doméstico
Negocios y Profesionistas

Ventas por Compañia 1982

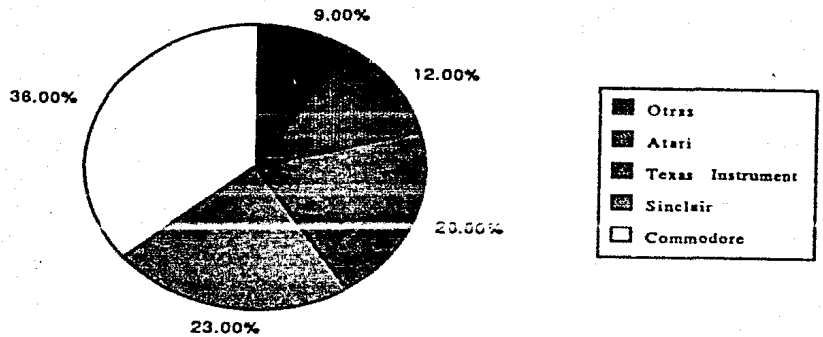
$4,100 Millones de Dolares





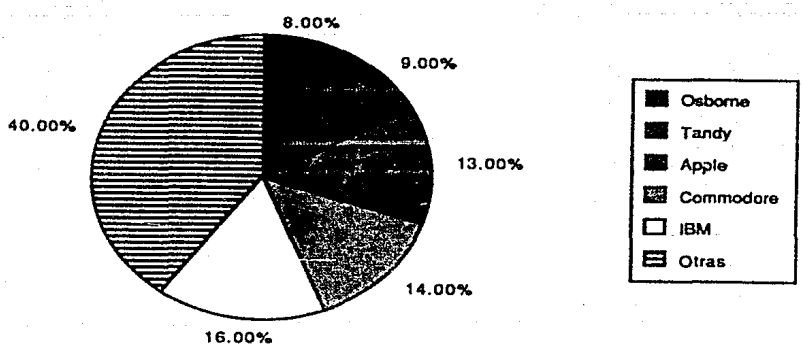
Ventas Uso Doméstico

1,977,000 Unidades



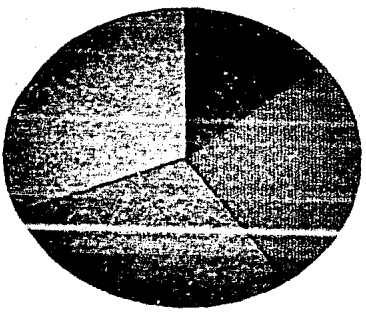
Ventas Uso Profesional y Empresas

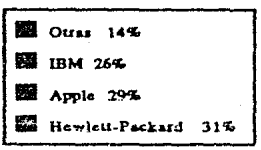
225,000 Unidades



Ventas Uso Científico

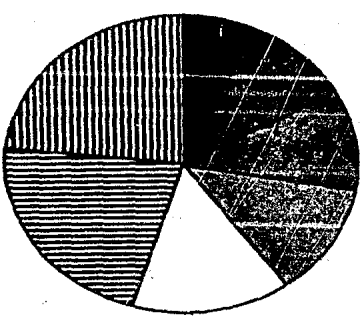
80,000 Unidades

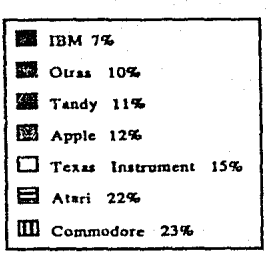




Ventas Uso Educativo

147,000 Unidades





2.2.4 Configuración ideal

Finalmente, como un resumen de lo que sería una configuración ideal para un sistema procesador de palabras, lo siguiente:

Procesador de 16 bits por palabra, a pesar de que el tamaño de la palabra no es elemento que influya determinantemente en la ejecución total de una microcomputadora. Pero que sea lo más rápido posible.
Monitor de fósforo ámbar o verde, de once o doce pulgadas, capaz de desplegar aproximadamente ochenta columnas por venticuatro renglones con alta resolución.
Teclado independiente o móvil, con teclas grandes, ligeramente cóncavas y de estructura firme.
Capacidad para trabajar como terminal inteligente o para comunicarse fácilmente con alguna red de computadoras.
Más de 128 K de memoria de Acceso Aleatorio.
Capacidad de manejo de impresoras en serie o en paralelo, para preveer el caso de contar con varios tipos de impresoras.
Impresora de caracteres fijos, de preferencia de esfera.
Dos o tres unidades de discos flexibles, que almacenen de 170 a 350 K individualmente.

2.3 ANALISIS DE SOFTWARE

2.3.1 Introducción

Tan importante como escoger la computadora en la cual se va a desarrollar un sistema, es la elección del lenguaje que se va a emplear. Aunque en cualquier lenguaje se pueden escribir programas legibles y bien estructurados, algunos tendrán ventajas en otros aspectos. Así como en la selección de la máquina se puede aportar mejoras en la velocidad o poderio de calculos, al escoger el lenguaje más conocido para los programadores o el que cuenta con más utilerias puede ser lo mejor, también puede ocurrir que se tenga que usar el único lenguaje con que se dispone.

A continuación se incluye una clasificación de lenguajes de programación y las partes importantes que se deben observar al escoger un lenguaje de programación para desarrollo de un sistema, que son: declaraciones, tipos, módulos, estructuras para el control del flujo de programas, manejo de errores y de excepciones, y como elemento muy importante, la capacidad de realizar compilaciones separadas (que implica pruebas separadas).

2.3.2 Clasificación de Lenguajes

Actualmente existen muchos lenguajes de programación con características muy diferentes entre ellos, pero se pueden clasificar en cinco grandes gupos:

1.-**Lenguajes Ensambladores.** Con equivalencia directa al lenguaje de máquina, muy poco recomendables excepto en aplicaciones muy especiales.

2.-**Lenguajes de Desarrollo de Sistemas (Systems Implementation Languages).** Se derivaron de los lenguajes ensambladores cuando se notaron las grandes dificultades que presentaban éstos. Proveen facilidades de estructuras y control de flujo en los programas, y hasta funciones de alto nivel, como la revisión de tipos en el uso de las variables, pero son considerados de bajo nivel porque dan la capacidad de accesar directamente las operaciones que ejecuta la máquina. Son altamente recomendables, si lo que se desea realizar es muy especial y complicado. Un ejemplo muy conocido de estos lenguajes es "C".

3.-**Lenguajes de Alto Nivel.** Dan al programador todas las facilidades para manejar variables y estructuras, así como amplia variedad de instrucciones de control de flujo. Se caracterizan por la disposición estática de sus variables y código. Estos fueron los primeros lenguajes de alto nivel : FORTRAN y COBOL.

4.-**Lenguajes de Alto Nivel Estructurados.** Se derivan de los lenguajes de alto nivel, pero se distinguen por tener construcciones de control variadas, tales como "IF-THEN-ELSE", "WHILE", "REPEAT", y por poder clasificar objetos (valores) como de cierto tipo. También tienen disposición dinámica de la memoria, permitiendo rutinas y funciones con parámetros y variables locales, esta característica, junto con sus construcciones de control, permiten a estos lenguajes gran modularidad y facilitan la programación estructurada. Estos lenguajes no son tan ampliamente usados como los anteriores por ser más nuevos, pero se están volviendo más populares a medida que se reconocen sus ventajas. Ejemplos bien conocidos son: PASCAL y ALGOL.

5.-**Lenguajes de Alto Nivel Dinámicos.** Se caracterizan porque la disposición de la memoria se hace por completo dinámicamente, cada instrucción ejecutada provoca una serie de asignaciones y desasignaciones. En general, la lógica de estos lenguajes es muy

diferente a la de los otros lenguajes de programación, y aún entre ellos suelen ser muy diferentes. Aunque estos lenguajes son especialmente buenos en algunas aplicaciones de desarrollo e investigación, su uso es limitado. Ejemplos de este tipo de lenguajes son APL y LISP.

Por la capacidad de los lenguajes de alto nivel de utilizar instrucciones lógicas en lugar de instrucciones de máquina, y nombrar a porciones de memoria destinadas para almacenar valores (variables) y a partes de código (subrutinas) con nombres significativos, Los programas escritos en estos lenguajes, además de haber sido más fáciles de crear, son mas fáciles de leer y de mantener que los escritos en lenguaje ensamblador. realmente el uso de lenguajes ensambladores se debe limitar para ocaciones en que se tengan necesidades especiales de velocidad o espacio en memoria.

2.3.3 Comparativo de lenguajes.

Ahora hablaremos de algunos lenguajes populares en el desarrollo de programas en microcomputadoras, prefiriendo los que más conozcan los programadores.

C

"C" es un lenguaje relativamente nuevo que ha tenido una gran aceptación y que continúa haciéndose más popular. Fue hecho originalmente para escribir UNIX, sistema operativo ampliamente usado y que marcó pautas que han imitado muchos otros sistemas operativos. Actualmente "C" se ha usado para proyectos de mucha importancia, como escribir sistemas operativos, realizar animaciones para películas de ficción y hacer programas muy sofisticados, tanto en microcomputadoras como en computadoras grandes, y todo esto sucede porque "C" proporciona código extremadamente rápido y compacto. Este lenguaje es capaz de aprovechar las ventajas de la computadora en la que reside, por su manejo de instrucciones de bajo nivel, y aún ser claro, eficiente y productivo como Pascal, por manejar varias estructuras de control y de datos. Entre otras cosas permite una gran portabilidad, tiene un pequeño KERNEL de 30 palabras reservadas, sus rutinas de entrada y salida están en una biblioteca que es fácil de hacer (en "C" es muy fácil hacer módulos independientes de cualquier programa). Por estas razones implementar un sistema en "C", y después reescribirlo para otras computadoras es muy fácil.

"C" cuenta con una gran variedad de operadores con lo que se ajusta a las ventajas de cada máquina. Aunque en un principio la gran cantidad de operadores que maneja lo hacen difícil de aprender, por ejemplo: "++" significa incrementar (muchos procesadores pueden realizar esta operación con una sola instrucción de máquina).

Otra ventaja es que cuenta con operadores de modo bit, tales como AND, OR, SHIFT, etc. También cuenta con un manejo de apuntadores bastante amplio, inclusive se puede resolver el problema que se mencionó cuando se habló de apuntadores, ésto es, permite referenciar una variable a través de su nombre o a través de un apuntador, lo que no pasa en Pascal, por ejemplo. La validación de tipo si se realiza, porque un apuntador está definido para un tipo específico de objetos. Además, se pueden realizar cálculos con apuntadores para direccionar arreglos, por ejemplo: si X es un arreglo de reales y P apunta a X[0], P+1 apunta a X[1], a pesar de que la longitud de un real no es 1. Esto puede llegar a causar problemas, como accesar accidentalmente memoria no utilizada, pero nos da muchas ventajas en cuanto

a facilidad de programación y libertad para usos muy especiales y totalmente prohibidos en muchos lenguajes, tales como accesar memoria directamente y no a través de variables, ésto es, como la capacidad de direccinamiento directo que tiene BASIC con los "PEEKs" y "POKEs", pero de manera más formal y elegante.

Soporta variables estructuradas (como los registros de Pascal), y permite convertir de cualquier tipo a otro para aplicaciones especiales, por ejemplo: usar un real como un entero para trabajar sobre el bit que se quiera.

En cuanto a la portabilidad, "C" ha marcado toda una pauta, porque, además de relizar programas muy eficientes aprovechando las características de cada máquina en especial al estilo de los lenguajes ensambladores, los programas escritos en "C" frecuentemente son muy fáciles de usar en otra computadora y hasta bajo otro sistema operativo.

"C" permite una gran modularidad , puesto que todo se maneja a través de funciones (o subrutinas), además de que se pueden conservar valores de variables locales entre una llamada y otra de una función. Además, en "C" es fácil de armar librerías de funciones o implementar rutinas a nivel de sistema operativo.

Actualmente se consiguen una gran variedad de compiladores "C" para microcomputadoras bajo diferentes sistemas operativos y varias herramientas de programación y desarrollo para "C".

Una desventaja de "C" es que no se sabe el orden en que realiza las operaciones en una fórmula o el orden en que se evalúan varias expresiones que se requieren a un mismo tiempo, como por ejemplo, los parámetros que se pasan a una función, ésto puede causar resultados imprevistos si se programa descuidadamente, y además este tipo de errores son muy difíciles de detectar.

"C" permite mucha mucha libertad, demasiada, si se piensa usar por gente sin mucha experiencia, "C" es un lenguaje para programadores profesionales.

"C" es una buena elección como lenguaje de desarrollo para proyectos sofisticados, pero tiene la desventaja de requerir programadores muy experimentados y con buenas costumbres de programación, para no caer en vicios peligrosos por la libertad que ofrece, o cometer errores difíciles de detectar que suelen ocurrir cuando se programa despreocupadamente en él.

LOGO

Desarrollado en el MIT (Massachusetts Institute of Technology) a partir de LISP, aunque popular como lenguaje de introducción a la programación y ser muy poderoso (modular, rutinas con parámetros, recursividad, comunicación con ensamblador, etc), aleja mucho al programador de la computadora, no permitiendo aprovechar las ventajas que ofrece la misma.

a facilidad de programación y libertad para usos muy especiales y totalmente prohibidos en muchos lenguajes, tales como accesar memoria directamente y no a través de variables, ésto es, como la capacidad de direccionamiento directo que tiene BASIC con los "PEEKs" y "POKEs", pero de manera más formal y elegante.

Soporta variables estructuradas (como los registros de Pascal), y permite convertir de cualquier tipo a otro para aplicaciones especiales, por ejemplo: usar un real como un entero para trabajar sobre el bit que se quiera.

En cuanto a la portabilidad, "C" ha marcado toda una pauta, porque, además de relizar programas muy eficientes aprovechando las características de cada máquina en especial al estilo de los lenguajes ensambladores, los programas escritos en "C" frecuentemente son muy fáciles de usar en otra computadora y hasta bajo otro sistema operativo.

"C" permite una gran modularidad , puesto que todo se maneja a través de funciones (o subrutinas), además de que se pueden conservar valores de variables locales entre una llamada y otra de una función. Además, en "C" es fácil de armar librerías de funciones o implementar rutinas a nivel de sistema operativo.

Actualmente se consiguen una gran variedad de compiladores "C" para microcomputadoras bajo diferentes sistemas operativos y varias herramientas de programación y desarrollo para "C".

Una desventaja de "C" es que no se sabe el orden en que realiza las operaciones en una fórmula o el orden en que se evalúan varias expresiones que se requieren a un mismo tiempo, como por ejemplo, los parámetros que se pasan a una función, ésto puede causar resultados imprevistos si se programa descuidadamente, y además este tipo de errores son muy difíciles de detectar.

"C" permite mucha mucha libertad, demasiada, si se piensa usar por gente sin mucha experiencia, "C" es un lenguaje para programadores profesionales.

"C" es una buena elección como lenguaje de desarrollo para proyectos sofisticados, pero tiene la desventaja de requerir programadores muy experimentados y con buenas costumbres de programación, para no caer en vicios peligrosos por la libertad que ofrece, o cometer errores difíciles de detectar que suelen ocurrir cuando se programa despreocupadamente en él.

LOGO

Desarrollado en el MIT (Massachusetts Institute of Technology) a partir de LISP, aunque popular como lenguaje de introducción a la programación y ser muy poderoso (modular, rutinas con parámetros, recursividad, comunicación con ensamblador, etc), aleja mucho al programador de la computadora, no permitiendo aprovechar las ventajas que ofrece la misma.

Es un lenguaje interpretado y por lo tanto lento, y el sistema ocupa una parte importante de la memoria, por lo tanto no es recomendable para desarrollar sistemas muy complicados con problemas de velocidad y de espacio en memoria.

FORTH

Pertenece a la corriente de "Threaded Interpretative Languages" que consiste en lenguajes que actuan como intérpretes (como BASIC), ésto es, se desarrolla y se prueba al mismo tiempo, se pueden estar dando órdenes directamente y la computadora responde como lo haría al encontrar esas instrucciones en un programa. FORTH además, tiene la característica de permitir definir nuevos comandos (en la misma manera que LOGO), y de poder trabajar como compilador para ganar rapidez y espacio en memoria, y ésto hacerlo modularmente, ésto es, los módulos que ya han sido probados ampliamente se compilan para ganar velocidad mientras se siguen desarrollando o probando otros, todo esto interactivamente, con comandos directos del teclado durante una sesión de trabajo.

ADA

Se deriva del Pascal en 1978, con la inclusión de conceptos muy nuevos de programación, un ejemplo es su capacidad de sustituir eficientemente programas que antes sólo se hacían en ensamblador en microcomputadoras, y con su tendencia a permitir gran portabilidad.

Al estilo de Pascal, permite hacer una amplia gama de estructuras de datos de programación, pero todo restringido dentro de un marco rigido que evita y marca muchos errores, pero que quita la libertad en algunos casos o aumenta el tiempo de ejecución y el tamaño del programa por todo el código de supervisión que genera (igual que Pascal), aunque muchos de estos códigos de supervisión se pueden omitir con las opciones al compilador, no se recomienda mucho su uso, por hacer los programas propensos a errores y dificiles de transportar.

La forma de escribir ADA es de formato libre como Pascal (excepto por los comentarios, que terminan con el fin de línea), lo cual permite escribir los programas de una manera elegante y comprensible.

La forma de un programa es una serie de rutinas, al tipo de "C" o "Modula", que se pueden compilar por separado, de hecho la forma de escribir subrutinas recuerda la forma de escribir bibliotecas en otros lenguajes, primero una especificación de las rutinas y sus parámetros (la parte de comunicación), y después el cuerpo de las rutinas. También se pueden anidar las rutinas dentro de otra global, como en Pascal, en donde todas las rutinas están anidadas en el programa principal, pero esta forma le resta

modularidad al programa y facilidad para depurarlo, además de que muchos datos son accesibles desde distintas rutinas, pudiéndose generar muchos efectos colaterales.

Actualmente no se consiguen muchas utilerías para trabajar con ADA en microcomputadoras, ni siquiera existen muchos compiladores ADA para ellas, por lo que su uso no es fácil y la documentación (bibliografía) relacionada es muy reducida.

Pascal

Diseñado por Niklaus Wirth a partir de ALGOL, con la finalidad de conseguir una herramienta para enseñar los conceptos de la programación estructurada en un lenguaje compacto, de manera que cupiera en casi cualquier máquina para que se pudiera convertir en un estándar de programación.

Realmente resultó compacto y fácil de dominar totalmente después de cierto tiempo de utilizarlo, ésto, más el hecho de que es totalmente estructurado y de que tiene una forma de escribirse libre de formato, hizo que se usara y que se use aún ampliamente para escribir algoritmos.

Actualmente es muy usado, y tiene muchas ventajas para sistemas grandes, una es el hecho de que evita muchas de las malas prácticas en programación, otra es su facilidad de expresar ideas de una manera natural y similar a como se expresaría en lenguaje natural, los programas son mucho muy legibles, y no se tienen que manejar claves obscuras, o hacer un manejo de datos complicado, gracias a su capacidad de definir tipos de variables, asignando a esos tipos los valores que pueden tomar o la forma de la variable en sí (éste es el caso de los registros).

Pascal tiene muchas ventajas en cuanto a programación clara, confiable, mantenible y transportable, sus desventajas son en el desarrollo de sistemas que tengan mucha relación con la computadora y con el sistema operativo en general. Pascal aleja mucho al programador del nivel de máquina, facilitando el trabajo común, pero restándole libertad. En Pascal es casi imposible hacer algo que se salga del modelo para el que fue hecho, el programador no tiene casi control sobre el código que resultará, ni tiene ninguna forma de acceso directo a las instrucciones de máquina ni a la memoria (excepto por el uso de funciones escritas en ensamblador, ya que muchas implementaciones de Pascal tienen todas las herramientas para cominicarse con ellas).

Pascal es una muy buena elección en cuanto a lenguaje para desarrollar sistemas, siempre y cuando el sistema no tenga limitaciones críticas en cuanto a su eficiencia del código (tamaño y velocidad), o requiera operaciones muy especiales de la computadora.

BASIC

Aparece como un lenguaje para microcomputadoras, pequeño (en cuanto al espacio que el sistema ocupa en memoria) y fácil de aprender y usar, no es un lenguaje estructurado ni poderoso, se puede clasificar como lenguaje de alto nivel estático, es tal vez el lenguaje más popular actualmente (en microcomputadoras).

La principal razón del éxito de BASIC es el hecho de que es interpretado y de que viene como lenguaje estándar en muchas computadoras. El ser interpretado le da la ventaja de que el desarrollo de programas es interactivo, en el sentido de que se puede probar cada parte de un programa por separado, aún una línea, y observar como trabaja inmediatamente, ésto representa una gran ventaja sobre los lenguajes compilados, en donde la realimentación es muy lenta por el proceso que se tiene que seguir y además se tiene que escribir código para observar la mayoría de los procesos. En BASIC es posible detener la ejecución de un programa cuando se quiera y monitorear el estado general del proceso u observar cualquier variable en especial, después, realizar la operación que se quiera sobre los datos o sobre el programa, o continuar la ejecución del mismo, lograr ésto en otros lenguajes requiere el uso de herramientas muy sofisticadas de programación y muy difíciles de usar.

Una gran desventaja de los lenguajes interpretados es su lentitud, pero ésto se puede resolver actualmente en muchos caso gracias a la aparición de compiladores para muchos BASICs.

Otra desventaja es el hecho de que existen muchísimas versiones de BASIC. Al notar las deficiencias de este lenguaje, muchas empresas le hicieron ampliaciones hasta el grado de que es el lenguaje que cuenta con más utilerías actualmente (sólo se pueden comparar COBOL y "C" en este aspecto, COBOL por la misma razón de ser popular y tener muchas deficiencias, y "C" por la gran facilidad que da este lenguaje para escribir utilerías eficientes), a tal grado que se pueden conseguir muchos BASICs muy poderosos, tanto como los lenguajes estructurados, pero toda esta cantidad de versiones son incompatibles, teniendo que realizar siempre una pequeña adaptación, o a veces casi volver a escribir el programa cada que se quiere transportar a otra máquina o inclusive a otra versión de BASIC de la misma máquina, algunas veces resultando imposible esta traducción.

Otra desventaja que tiene es la forma en si del lenguaje, BASIC es desestructurado, y aunque actualmente se consiguen versiones que supuestamente son estructuradas, ésto solo es en algunos aspectos, y aún en las más poderosas y nuevas implementaciones se sigue caraciendo de muchas facilidades y algunas limitaciones que hacen tan claros y mantenibles los programas escritos en lenguajes estructurados. BASIC es muy propenso a generar errores en la lógica de los programas, algunos muy difíciles de corregir. Algunas de las peores características

de los lenguajes de programación existen en BASIC, como el de no requerir una declaración de las variables y el de que las variables no tienen un área de acción limitada.

Una gran ventaja de BASIC es el dominio total que ofrece sobre la máquina y del sistema en que habita (en máquinas pequeñas, puesto que es el lenguaje residente en muchas de ellas), aunque no necesariamente fácil ni elegante.

BASIC tiene acceso directo a la memoria a través de las funciones PEEK, POKE y CALL, ésto le da posibilidades muy poderosas aunque también puede hacer los programas terriblemente difíciles de leer.

En general, los programas en BASIC no son legibles, aún cuando se puede modularizar y usar mensajes (comentarios), ni siquiera estos pueden existir en cualquier parte, por el hecho de que no tienen un formato libre de programación, requiriendo inclusive números de línea.

BASIC es un lenguaje con muchos defectos pero todos ellos se pueden evitar con un buen estilo de programación y tiene las ventajas del mejor ambiente de desarrollo, por el hecho de ser interpretado, y el hecho de ser el más popular y contar por lo tanto con muchísimas herramientas de programación, amplia bibliografía en libros y revistas y gran experiencia por parte de los programadores (o facilidad de aprendizaje y práctica en caso contrario), lo hacen ser una de las mejores opciones en cuanto a la elección de lenguaje para desarrollo.

2.3.4 Lenguaje ideal

No existe un lenguaje ideal, ni siquiera se puede hablar del mejor de los lenguajes, lo que sí es cierto es que existen lenguajes con facilidades para ciertas aplicaciones, o funciones que facilitan la codificación de algunos programas.

Después de haber hablado de las características generales de los lenguajes de programación y de haber comentado algunas ventajas de algunos de ellos, ahora vamos a enumerar algunas características y funciones deseables en el lenguaje usado en la implementación de un procesador de textos.

Estructurado.- es una de las características que siempre son deseables para hacer programas legibles y mantenibles, aunque en este caso en especial, el lenguaje que se use también debe permitir cierto grado de desestructuramiento para poder manejar los datos de manera natural, dado que tenemos una única y gran estructura de datos, que es todo el archivo que se está trabajando en memoria y en disco, y además con un gran conjunto de pequeños datos que se refieren directa o indirectamente a esa estructura, todos ellos pasándose de una rutina a otra y usándose en muchos lados, dándole un aspecto muy desestructurado a los datos.

Estructuras de datos.- Debe de tener un manejo amplio de caracteres y cadenas de éstos, puesto que es la base de los datos que se manejan.

Manejo de periféricos.- Debe tener amplio manejo sobre el despliegue en pantalla y libertad en cuanto a su comunicación con la impresora, puesto que sobre estos dos medios se va a trabajar ampliamente y algunos casos serán muy especiales (otra vez el manejo amplio y libre de caracteres en general).

Manejo de memoria.- Es deseable que pueda manejar la memoria directamente y que se comunique o modifique fácilmente con el sistema en que habita y que los programas producidos no requieran un gran sistema que los soporte en memoria, pués esto limita aún más la memoria disponible. Es bueno que el programador pueda decidir totalmente en donde va a residir su programa y en donde sus datos; la comunicación con el sistema es para poder manejar varios archivos y poder ejecutar todos los comandos necesarios para administrar eficientemente una colección de archivos del usuario en disco.

Herramientas de desarrollo.- Como se va a desarrollar un sistema grande, es muy conveniente que se cuente con amplias herramientas de depuración y desarrollo, un buen editor de programas, un compilador rápido y que permita compilaciones

parciales, y una buena herramienta para observar el programa durante su ejecución, al estilo de los "debuggers", o mejor que todo ésto, un lenguaje interactivo, el cual constituye el mejor depurador por sí mismo, pero que dé la opción de compilar para mejorar eficiencia y rapidez.

Otras herramientas.- Que existan utilerías ya escritas, las cuales se puedan utilizar directamente para varias apliaciones específicas, por ejemplo, se necesita un amplio manejo de dispositivos exteriores, tales como una impresora, memorias extras, terminales extras, un paquete que manejara señales de este tipo sería útil, otro ejemplo es el hecho de que pocos dispositivos de salida soportan algunos elementos del español, tales como los acentos, las eñes, etc., un paquete que simulara estos caracteres sería muy útil también.

Comunicación.- Que se pueda comunicar con otros lenguajes, especialmente con ensamblador, para eficiencia y rapidez de algunas rutinas críticas, o algunos manejos especiales y no soportados por el lenguaje.

Manejo de "overlays".- Además de que el código generado sea pequeño, porque se requiere mucho espacio de memoria para el archivo, es deseable que maneje "overlays" o intercambios ("swap"), todo ésto bajo el control del programador.

Formato.- Que tenga un formato de escritura libre, que permita escribir programas claros y legibles.

3 Definición de la Interfase

3 DEFINICION DE LA INTERFASE DE USUARIO

3.1 Interfase de Usuario

3.1.1 Diseño de la Interfase para el Usuario

El término Interfase de Usuario quiere decir la forma en que el programa se comunica con el usuario e incluye la forma en que el usuario puede manejar cualquier programa o sistema, la presentación del mismo y la documentación que lo acompaña.

Anteriormente este elemento no se había considerado como importante, pero como las computadoras cada vez son más independientes del técnico que las programa, no podemos dejar que pase desapercibido.

El éxito con el que se diseña cualquier programa depende de la habilidad que se tenga para predecir su comportamiento. Si se es muy optimista en estas estimaciones, seguramente se llegará a tener un programa que no cumpla con los requerimientos originales.

La comunicación con el usuario es el elemento de un programa interactivo más difícil de predecir y el que determina principalmente la aceptación de un producto.

Es importante tener mucho cuidado con el diseño de interfases interactivas. Las interfases deficientes son difíciles de aprender, pero también hacen que el programa sea ineficiente al usarlo, aún cuando el usuario tenga mucha experiencia. En casos extremos, el programa probará ser imposible de usar, ineficiente o poco confiable, de manera que el costo que provoque el uso del mismo no será razonable. La interfase debe ser familiar al usuario, utilizar con buen rendimiento los elementos y no saturar el uso de las teclas.

En el caso particular de nuestro Procesador de Palabras, el diseño de la interfase del usuario fue laborioso y tardado, y como resultado obtuvimos una interfase de aspecto familiar para una secretaria, de operación guiada por menús, y de funcionamiento consistente en los diferentes niveles del programa. A pesar de que esta interfase era adecuada, se optó por usar otra, la interfase del Mouse para Apple. La razón principal por la cual se usó esta interfase fue por consistencia

con otros desarrollos para la misma microcomputadora, las otras razones se detallan más adelante en el capítulo de decisiones tomadas.

Características de la Interfase de Usuario usada.

Dada la importancia que ha llado a tener la interfase de usuario, la compañía Apple desarrolló todo un conjunto de rutinas básicas para que los desarrolladores de software las puedan unir a sus programas. La principal caracteristica de este conjunto de rutinas es que nos permite aprender y usar fácilmente los sistemas. El fin que se persigue es el de crear un estándar en interfases, de manera que cualquier usuario que haya manejado esta interfase anteriormente no tendrá ningún problema para manejar un nuevo programa.

Algunas caracteristicas de esta interfase son el manejo de ventanas, manejo del ratón, representaciones gráficas de algunas funciones, menús del tipo "pull-down". Estas funciones permitirán al usuario sentirse más a gusto con una aplicacion, pues se trata de que las habilidades del usuario se desarrollen en lugar de forzarlo a aprender nuevas habilidades.

Las principales cualidades de esta interfase son:
Responsividad, es decir que las acciones del usuario producen resultados directos, y se podrán ver reflejados de inmediato.
Permitividad, las aplicaciones permiten al usuario realizar cualquier cosa bajo su responsabilidad. Por esta característica, se trata de evitar que los sistemas se dediquen a analizar las operaciones que el usuario pudiera hacer, por muy definitivas que sean. Si debido a alguna operación no se puede recuperar el estado anterior, teóricamente, para la siguiente vez que se use, se hará con más cuidado.
Consistencia, las funciones o teclas se usan de manera muy parecida, si no es que igual, a través de todos los módulos de una aplicación, e idealmente en todas las aplicaciones. Esto se realiza así, pues una persona divide su tiempo entre varias aplicaciones, y se irrita por el hecho de que cada aplicación tiene siempre una manera diferente de manejarse.
Lo que usted ve es lo que usted obtine, este es otro aspecto importante. De acuerdo a esta característica lo que aparece en la pantalla, y la forma en que se ve el documento, es exactamente lo que se imprimirá.

3.1.2 Componentes de la Interfase del Usuario

La interfase del usuario se divide en forma natural en cuatro partes, el modelo del usuario, el lenguaje de comandos, la realimentación, y la visualización de la información. El diseño de la interfase no puede pasar por alto ninguno de estos elementos, y se tiene que realizar tomando en cuenta los recursos de los cuales disponemos y las necesidades del usuario.

A partir de la primera parte, "modelo del usuario", es fácil entender la existencia de las otras tres. Es un modelo conceptual, que el ususario se forma de la información que maneja y los procesos que le aplica a ésta.
El modelo es útil para que se entienda mejor lo que el programa está haciendo, sin necesidad de tener conocimientos en computación. En ocasiones, el diseño del modelo del usuario, es sólo una simulación del sistema real.

Cuando el usuario ha entendido el modelo, necesita instrucciones para manejarlo, por lo que la segunda parte de la interfase del ususario es el "lenguaje de instrucciones o comandos", que acompaña al programa. Se desea que los programas de computadora tengan lenguajes de instrucciones naturales, es decir, que se puedan aprender a usar sin estar consciente de ello.

El tercer elemento es la "realimentación", con la cual el usuario se ayuda al manejar el programa. La realimentación puede ser de varios tipos: reconocimiento de la recepción de instrucciones, mensajes de explicación, indicación de los elementos seleccionados y ecos de los caracteres tecleados. Unas formas de realimentación se usan para ayudar a los usuarios inexpertos, otras son necesarias para el uso de las instrucciones.

La cuarta parte, "visualización de la información", es necesaria dado que la realimentación no muestra el efecto de las instrucciones. Esta última parte muestra al usuario el estado de la información que está manejando y trata de dar una imagen organizada de la información. Con esta imagen el usuario se asegura de que su modelo sea correcto. Si el modelo depende de su realismo, la imagen visualizada debe ser así, si el modelo es sintético, se debe reforzar por medio de símbolos y de una imagen gráfica, muy bien escogida, en la información mostrada.

Los cuatro elementos de la interfase están muy relacionados entre sí, pero esta división ayuda en el diseño y en el análisis. Antes de poderlos estudiar por separado, es necesario hacer notar la importancia de un buen análisis de las funciones requeridas; por lo tanto recomendable comenzar el diseño con éste.

También es necesario establecer los modos de operación que existirán. Un modo es la parte de una aplicación que un usuario use de manera formal, en la que se rentrinjan las operaciones que pueda realizar mientras el modo esté en función.

Lenguaje de Instrucciones

Dado que todas las funciones se ejecutan por medio de menús pull-down, la parte principal del lenguaje de instrucciones es el acceso a éstas, éste se puede lograr por medio del ratón o por el teclado. Cuando se usa el ratón, su representación en pantalla es una flecha en el area de menú y una barra vertical, simulando al cursor en el area de trabajo. La selección por medio del ratón se efectua al oprimir el boton del mouse, conociéndose esta acción como "click". Si se usa el teclado, se considera una simulación del ratón. Como en toda la aplicación las acciones del usuario son iguales se logra consitencia en las funciones de todos los módulos.

Para que una operación o comando se realiza, deberá se seleccionarse la información anticipadamente, ahorrándose así algunos pasos.

Realimentación

La realimentación en esta interfase nos permite siempre ver en la pantalla el efecto de las acciones tomadas, ya sea una inserción, una selección, un borrado, etc.

El cursor siempre nos permite ver donde se realizan las operaciones, y en el caso de que nos encontremos en el menú o cambiemos de modo, también nos lo hace notar debido a que cambia de forma.

Los errores también se manejan como un cierto tipo de avisos, pudiendo ser sonoros o visuales, dependiendo de la gravedad del error.

Visualización de la información

Mientras el usuario está tnabajando, la única forma en que se puede visualizar un documento es por medio de una ventana, y la manera de realizar las funciones es a través de un menú.

La primera linea de la pantalla está reservada para la linea de menús, en esta línea se incluyen los titulos de los menús asociados. Cuando se accesa la línea, ya sea por ratón o por teclas, cada menú se abre hacia abajo mostrando todas las opciones que incluye y sus abreviaturas o comandos alternativos,

si existen. Cuando se trata de usar una función, se necesita pasar el cursor por encima y cuando esto sucede, las funciones por donde pasa el cursor se visualizan en inverso. Algunas funciones sólo tienen dos valores, y por lo tanto o están prendidas o apagadas; este tipo de funciones se representan en el menú por medio de un indicador junto al nombre, si la opción esta prendida este indicador aparece en forma de una palomita.

En el caso de ser seleccionada una función que requiera de algunos parámetros, se abre otra pequeña ventana, ventana de diálogo, donde se prodrá cambiar estos valores por medio de botones o tecleando los valores.

Una vez seleccionada la opción de editar un documento, se abre una ventana. Las partres de la ventana son el título, la barra de scroll vertical (a la derecha), la barra de scroll horizontal (en la parte inferior, y que no estará en funcionamiento) y por último, la zona de trabajo.

En el título se encuentra el nombre del documento, en las barras de scroll se ve la posición relativa del cursor en el documento, y en la zona de trabajo el texto. El texto se representa en forma gráfica, con el fin de poder representar los diferentes tipos de letras que se manejan y los acentos.

Existe la opción de ver los valores de márgenes y pies de página. La representación de éstos se realiza por medio de una regleta, similar a las de las máquinas de escribir, que se intercala en el texto y que puede ser modificada.

Cada ventana de trabajo puede mostrar vistas parciales de un documento. La parte de la pantalla que contiene a las ventanas es llamado escritorio ("desktop").

3.1.3 Generalidades de su manejo

La interfase desarrollada por Apple es lo suficiente poderosa para poderse usar sin el ratón, pero es mucho más fácil usarse con ratón.

Los menús pull-down se pueden accesar por medio de la tecla <Esc> y movimientos del cursor, y por medio de combinaciones de las manzanas y algunas letras.

En cuanto al uso del ratón, existen tres acciones básica:
Clicking, esto es cuando se posiciona el cursor con el mouse, y entonces brevemente se oprime y se deja el ratón sin moverlo.
Oprimiendo, se posiciona el cursor con el mouse, manteniendo oprimido el botón del ratón sin moverlo.
Dragging, o desplazando, se mueve el cursor manteniendo oprimido el botón del ratón mientras se mueve.

Las selecciones del texto u objetos se realizan con el fin de distinguirlos de lo demás justo antes de realizarse alguna operación. Esta interfase proporciona los siguientes modos de selección:
Selección por clicking, es la selección de un solo elemento en un arreglo de éstos, y se realiza apretando el botón del mouse sobre el elemento.
Selección por rango, se posiciona el cursor en el extremo inicial del area o rango elegido, se mueve el cursor apretando el botón, se posiciona al final y después se suelta el botón.
Extensión de una selección previa, se realiza al oprimir una de las manzanas y cambiando de posición el botón del ratón.
Selección discontinua, es cuando se selecciones diferentes porciones de un documento de manera aislada.

Las operaciones que se pueden realizar con las ventanas:
Abrir y Cerrar, ésto es crear una ventana en nuestra area de trabajo o borrarla.
Activar ventana, el usuario solo puede trabajar en una ventana a la vez y dicha ventana es la activa, cuando se activa una, las demás inmediatamente se desactivan.
Mover, se puede mover una ventana para dejar libre algun espacio o descrubrir alguna ventana que esté abajo de otra.
Cambiar de tamaño, las ventanas tienen un botón de cambio en la parte inferior derecha, moviéndola se cambian las dimensiones de la pantalla hasta sus límites.
Scroll (deslizar), por medio de esta función se desliza la información contenida dentro de la ventana con el fin de visualizar otra parte del mismo documento.
Partir ventana, se permite separar una región de la ventana o panel, y aunque en sus desplazamientos son independientes, el conjunto de los paneles siguen siendo un solo elemento.

Otros objetos de control se diseñaron para ser parte de la interfase y aunque ya se habló algo de ellos, aquí se resumen en:
Botones, son objetos pequeños etiquetados y que al oprimirlos realizan lo que dice la etiqueta.
Check boxes, son elementos por medio de los cuales se prende o apaga algún parámetro.
Radio buttons, son elementos para seleccionar una de varias funciones, y como son excluyentes entre sí, al oprimir uno se apagan los demás.
Dials, es un despliegue analógico donde una cantidad se compara con su límite.

3.1.4 Manejo de Errores

Debido a que en esta interfase se permite al usuario ser libre al operar la aplicación, los errores probables son muy pocos. Aún así, existen dos tipos de manejo de error.

El primer tipo de error, detecta movimientos más alla del límite del documento. Dado que no es un error muy importante sólo se emite un sonido sencillo ("beep"), y se regresa al estado inmediato anterior.

El segundo tipo de manejo es un poco más formal. En este tipo de errores, no se puede seguir trabajando normalmente, y antes de regresar al estado anterior, se pide al usuario una confirmación de que está enterado. Esta confirmación se realiza por medio de una ventana que incluye una pequeña descripción de lo ocurrido.

3.2 Decisiones tomadas

3.2.1 Computadora Seleccionada

Ya que tuvimos una configuración ideal, estudiamos lo que existía en el mercado mexicano, y buscando estas características nos llevó a las microcomputadoras "Apple" y a las "IBM-PC". Nos decidimos a trabajar para "Apple" porque no existían procesadores de palabras en español para ésta, porque existía un conjunto de utilerías extenso y porque de este equipo se tenía mayor disposición.

3.2.2 ¿Porqué Pascal?

Existe una gran cantidad de razones para usar el lenguaje de programación Pascal, en nuestro caso, una razón fuerte fue que es un lenguaje que nos es familiar y en el caso particular de "Apple Pascal UCSD", porque es un compilador que puede reconocer la existencia de 128K Bytes en la Apple. Debido a la poca cantidad de memoria de esta microcomputadora, necesitamos de un lenguaje que ofreciera la facilidad de no tener todos los procedimientos siempre cargados en memoria ocupando un espacio, es decir, que tuviera manejo de "overlays", facilidad que ofrece el Pascal de Apple.

También, como es sabido, este compilador no genera código de máquina, sino un código intermedio, optimizado, llamado p-code, el cuál es interpretado por la máquina "p", y este tipo de código es muy compacto, lo cual redunda en un ahorro considerable en espacio de memoria RAM.

También es pertinente hacer notar que exite una interfase sencilla de utilizar, desde el punto de vista de programación, para hacer uso de las rutinas del "Mouse Tool Kit" desde Pascal.

Existe dos razones de peso para usar este lenguaje de programación sin tomar en cuenta las ventajas que ofrece una máquina en particular. La primera es que es más sencillo escribir programas en un lenguaje de alto nivel y estructurado como lo es Pascal, la segunda y tal vez la más importante es que es más fácil de leer y entender un programa escrito en un lenguaje estructurado.

Los programadores profesionales e ingenieros en software dedican gran parte de su tiempo a revizar y depurar programas escritos por ellos mismos y por otras personas, esto quiere decir que si ellos no entienden completamente estos programas, sus intentos de depuración y mantenimiento terminan por arruinar a un sistema completo. Por eso es tan importante el aprender a programar como el poder entender y modificar un programa.

Esto nos lleva a concluir que Pascal es el lenguaje ideal para programar un sistema en equipo.

Por el lado de la eficiencia de ciertas funciones criticas que requieren velocidad, Pascal permite de manera legible y sencilla hacer uso de rutinas escritas en lenguaje ensamblador. Como comentario importante al respecto, las rutinas que realizan el "swapeo" hacia los bancos de memoria de la tarjeta "SUPER RAM" están escritos en lenguaje ensamblador y son llamados por programas escritos en Pascal que realizan la función de administración de memoria.

Podemos incluír también que PASCAL cuenta con la facilidad de utilizar estructuras de datos dinámicas, es decir que crecen conforme lo necesite un proceso, lo cual da un aprovechamiento muy eficiente de la memoria.

Otra ventaja que ofrece este lenguaje es portabilidad; pués si quisiéramos pasar el programa de procesamiento de palabras a IBM solo tendriamos que reteclear los listados haciendo uso de un buen editor de texto. Claro está que necesitamos tener algo equivalente al MOUSE TOOL KIT para IBM; sin lo cual no podria corre el programa.

3.2.3 ¿Porqué la compatibilidad con ProDOS?

ProDOS es un sistema operativo que mejora en muchos aspectos al sistema operativo DOS de Apple pues contempla el uso de directorios y maneja los archivos en una organización arborea, lo cual permite tener más organizada la información en un diskette. Debido a estas mejoras muchos desarrolladores comenzaron a hacer sistemas bajo este sistema operativo, entre los cuales se destaca un paquete de software integrado de muy alta calidad y cuyo nombre es "Apple Works". Este paquete de software integrado cuenta entre otras cosas con un procesador de palabras muy poderoso.

Apple Works por su calidad y buen precio se fue haciendo cada vez más popular con el paso del tiempo, y esto lo ha convertido en un estandar. Como muchos paquetes manufacturados en USA, o contempla las necesidades de los mercados latinoamericanos es decir que le falta el uso de acentos en pantalla. Entonces, pensamos en ser compatibles con los archivos generados con este procesador de palabras, para ofrecer a los usuarios de APPLE WORKS una alternativa viable en cuanto a procesamiento de palabras se refiere.

3.2.4 Herramientas de Software

Las herramientas de software que se utilizaron para el procesador de palabras fueron: el mouse tool kit y el ProDOS and profile support tools.

El mouse tool kit es una interfase de usuario sumamente atractiva y fácil de usar. En esta interfase de usuario se plasman todos los conceptos e ideas surgidas de los proyectos "Smalltalk" de Xerox, Lisa y Macintosh de Apple. Estos proyectos se llevaron a cabo con la idea de ofrecer al usuario sistemas de manejo mas ágil y autónomo, y permitir a un usuario tener acceso a todas y cada una de las partes de un sistema en cualquier momento.

Entre sus principales características se encuentran el uso de los pull-down menús, el manejo de ventanas, y el manejo del concepto de escritorio o desktop. Los pull-down menús son un tipo de menús que se despliegan sobre la pantalla de manera vertical cuando se trabaja en ellos, y cualquiera de los elementos que esten abilitados se pueden ejecutar en ese momento; y cuando se deja el menu, se compacta ésta y se recupera la pantalla.

El manejo de ventanas se refiere a la facilidad que esta interfase ofrece para poder desplegar información diferente demanera simultanea. Por ejemplo, desplegar documentos diferentes, uno en cada ventana, o desplegar varias partes de un documento en varias ventanas.

Como comentario al respecto, la versión actual del procesador de palabras carece de esta características debido a restricciones en el espacio de memoria. Entre las operaciones que se permiten realizar con las ventanas estan el cerrarlas, abrirlas, moverlas, y cambiarlas de tamaño.

Otra característica que es importante de dicha interfase de usuario es el uso del dispositivo externo llamado "mouse" o ratón. Este es un periférico con el cual se pueden seleccionar los menús, mover el cursor, y hacer la operaciones de ventanas, simplemente deslizándolo sobre una superficie plana, tal como la mesa de trabajo.

El mouse tool kit permite trabajar con el teclado si no se cuenta con el raton, las rutinas de dicha interfase de usuario fueron implementadas por Apple Computer Inc. y están implementadas en lenguaje de máquina del 6502, la interfase puede ser utilizada desde APPLESOFT o como libreria desde Pascal.

La otra herramienta que se utilizó fue la librería de ProDOS and ProFILE supports tools. Sirve para poder tener acceso desde Pascal a diskettes de ProDOS, y si se usa, pues es importante

mencionar que Pascal UCSD de Apple tiene su propio sistema operativo y formato de disco, el cual no es compatible ni con ProDOS ni con DOS 3.3. Estas librerías contemplan el uso de directorios y subdirectorios, así como el uso de prefijos de ProDOS, además, permite hacer cualquier operacion sobre un archivo de ProDOS (crear, leer, escribir y borrar). También permite el uso de un disco duro externo llamado ProFILE.

3.2.5 Herramientas de Hardware Utilizadas.

La única herramienta de hardware que utilizamos fue una tarjeta de expansión de memoria llamada SUPER RAM que nos ofrece 512 Kbytes de memoria.

La razón por la que usamos esa tarjeta fue que debido a la gran cantidad de memoria que ocupa la interfase de usuario "MOUSE TOOL KIT", sumado a la memoria que ocupan las rutinas propias del procesador de palabras y aunado a los fonts para manejar diferentes tipos de letra, el tamaño del documento que era capaz de manejar este programa, sin tomar en cuenta la tarjeta de expansión de memoria, se reducia notablemente (30 lineas aproximadamente).

En principio de cuentas, los 128 Kbytes que dice tener la microcomputadora APPLE IIe es un espejismo, pues en realidad el procesador de esta máquina no puede direccionar más de 64 Kbytes. Lo que se hace es, utilizar localidades de memoria como switches para seleccionar cual de los bancos de memoria de 64 Kbytes se quiere utilizar.

Es importante hacer notar, que la manera en que la APPLE //e logra la pantalla de texto de 80 caracteres es usando el concepto de "interleaving", es decir que las columnas pares de la pantalla se encuentran mapeados en la página de texto de uno de los bancos y las columnas nones en el otro banco de memoria, lo mismo es utilizado para lograr la doble alta resolución pero se hace a nivel de pixels.

La forma en que se utiliza la memoria de 512 Kbytes es utilizando una localidad especifica de memoria en la cual se deposita el valor del banco de memoria seleccionado y cuyo valor va de 0..7 (localidad C073 HEX), es decir que son realmente 8

bancos de 64 Kbytes cada uno. De esta forma podemos tener cargado en memoria en el banco 0 hasta 30 líneas de texto que representan la parte del documento donde nos encontramos posicionados y la parte restante del documento se distribuye en los demás bancos. Lo anterior nos permite tener un documento de hasta 35 páginas aproximadamente (350 Kbytes) y 6 fonts diferentes.

Como ya se ha mencionado en alguna parte del escrito, al hablar de las limitaciones de la APPLE se mencionó que el generador de caracteres se encuentra bastante limitado si lo comparamos con el juego de caracteres de una IBM PC. y concretamente hablamos de la ausencia de letras acentuadas en pantalla. Esto nos llevo a pensar en desarrollar un procesador de palabras para la APPLE que utilizara caracteres dibujados en la pantalla de doble alta resolución. Dadas estas condiciones la parte de interfase de usuario que se utilizó fue la referente a gráficos.

4 Manual del Procesador

4 MANUAL DEL PROCESADOR

A continuación se presentan los elementos que formarán parte del procesador de palabras, junto con una pequeña explicación de la forma en que se maneja. Las funciones específicas que cada elemento realiza se pueden encontrar en uno de los apéndices.

4.1 Movimientos de cursor

Movimientos por caracter:

Flecha-Derecha	Avanza un caracter, en el extremo de la línea, el cursor se pasa a la primera columna después del margen izquierdo de la siguiente linea.
Flecha-Izq.	Retrocede un caracter, en el inicio del renglón el cursor pasa a la última columna antes del margen derecho en el renglón anterior.
Flecha-Arriba	Sube un renglón y se coloca en la misma columna, cuando está en la primera linea se coloca en la primera columna.
Flecha-Abajo	Baja un renglón y se coloca en la misma columna, cuando está en la última línea sólo se coloca al final del renglón.

Movimientos amplificados; usando manzana hueca y:

Flecha-Derecha	Avanza al extremo derecho de la linea, la siguiente vez se pasa al final del siguiente renglón.
Flecha-Izq.	Se mueve al inicio del renglón, en una segunda ocasión, se coloca al inicio del renglón anterior.
Flecha-Arriba	Pasa a la primera linea de la página, la segunda vez se coloca en la primera linea de la página anterior.
Flecha-Abajo	Coloca al cursor en la última linea de la página, la segunda vez, se colocará en la última linea de la página siguiente.

Movimientos amplificados; usando manzana llena y:

Flecha-Derecha	Se mueve hasta la última linea de la pantalla, la segunda vez que se pida, se va a la última linea de la pantalla siguiente.
Flecha-Izq.	Se va a la primera linea de la pantalla, la segunda vez va a la primera linea de la pantalla anterior.
Flecha-Arriba	Va hasta el inicio del documento.
Flecha-Abajo	Fin del documento.

Movimientos Varios:

Tabulador	Se coloca en la siguiente marca de tabulador.

Con manzana llena y:

Tabulador	Se mueve a la siguiente columna si existe.
Espacio	Coloca al cursor al inicio de la palabra siguiente.
Punto	Coloca al cursor al inicio del enunciado posterior.
Coma	Coloca al cursor al inicio de la frase posterior, es decir que para en ";", ":" o ",".

Con manzana hueca y:

Tabulador	Mueve al cursor a la columna anterior o a la marca de tabulador anterior.
Espacio	Coloca al cursor al final de la palabra anterior.
Punto	Coloca al cursor al inicio del enunciado u oración anterior.
Coma	Coloca al cursor al final de la frase anterior.

4.2 Caracteres en Español

Para obtener los caracteres en español, se hace como en una máquina de escribir, para los acentos se escribe primero el acento y luego la vocal, para la ñ, interrogación y admiración abierta. El teclado de la Apple //e ya tiene teclas especiales con esos caracteres.

4.3 Menú principal del Procesador de Palabras

4.3.1 Descripción del menu "Manzana" de la barra de menús.

En este menu se agregan todas las funciones de presentación y de configuración del sistema que son:

Acerca de CCD
Muestra una ventana con la presentación del sistema. Incluyendo aquí el nombre del programa, algunas fechas y autores.

Guías Manzana-?
Este es el punto de entrada para información adicional acerca del procesador, explicaciones generales y algunas no tan generales. Es todo un conjunto de ayudas.

Calculadora Manzana-K
Muestra una Calculadora.

Escoger Impresora
Muestra una lista de impresoras de las que se pueden usar y permite escoger una. En otro lugar será posible definir algunos caracteres especiales para manejar alguna impresora.

4.3.2 Descripción del menú "Documento"

Maneja los documentos en disco, pone un catálogo en forma de una tabla de nombres de documentos dentro de una ventana. La selección se realiza por movimientos de cursor o tecleando el nombre. Tiene la opción de seleccionar subdirectorios.

Nuevo
Por medio de esta selección se crea un documento nuevo.

Abrir ...
Lee un documento, y lo pasa a la memoria de trabajo.

Cerrar
Quita el documento de la memoria sin actualizarlo a disco.

Guardar
Realiza una copia del documento en que estamos trabajando a disco.

Guardar como ...
Realiza una copia del documento en que estamos trabajando a disco, perimitiéndonos cambiarle de nombre.

Imprimir ...
Imprime un documento. Adicionalmente abre una ventana donde se pueden establecer los parámetros de impresión, que son: número de copias, numeración, dispositivo de salida.

Salir **Manz-Q**
Salir del Sistema.

4.3.3 Descripción del menu "Editar"

En este menú se reunieron todas las funciones que realizen operaciones sobre bloques (cadenas de caracteres), incluyendo la manera se seleccionar el bloque. Sus abreviaturas, aunque no coinciden con los nombres de las funciones, si coinciden con las funciones que hacen otros programas.

Deshacer **Manz-Z**
Con esta opción se inhibe la última función realizada.

Cortar **Manz-X**
Quita la región marcada guardándola temporalmente.

Copiar **Manz-C**
Guarda temporalmente la región marcada.

Pegar **Manz-V**
Mueve la región marcada a la posición del cursor.

Borrar
Borra la región marcada definitivamente

Marcar **Manz-M**
El usuario pasa a un modo diferente, en el que sus movimientos sirven para marcar.

4.3.4 Descripción del Menú Localiza

Aquí se reunen las funciones básicas para realizar búsquedas de palabras u frases y reemplazarlas por algo más

Localiza Manzana-L
Al seleccionar esta opción el programa pide las palabras a buscar y luego la localiza.

Otravez Manzana-O
Realiza la búsqueda, exactamente como la vez anterior.

Reemplaza Manzana-R
Igual que Localiza, añadiendo la opción de indicar la palabra con la que se reemplazará.

4.3.5 Descripción del menú "Tipos"

Con este menú se cambia el tipo de letra que se esté usaando, permitiéndose ver en pantalla el cambio del tipo. Este cambio puede afectar a una zona marcada o a lo que se seguirá tecleando.

Centra
Aunque la función centrado no es un tipo especial de letra se incluye aquí.

Subraya Manzana-S
Cambia al tipo de letra subrayada.

Negritas Manzana-N
Cambia al tipo de letra negrita.

Enfatiza Manzana-E
Cambia al tipo de letra enfatizada o doble golpe.

Itálicas Manzana-I
Cambia al tipo de letra itálica.

Gordas Manzana-G
Cambia al tipo de letre doble ancho.

Flacas Manzana-F
Cambia al tipo de letra comprimida.

Altas Manzana-A
Cambia al tipo de letra superíndice.

Bajas Manzana-B
Cambia al tipo de letra subíndice.

Habituales Manzana-H
Regresa al tipo de letra normal.

4.3.6 Descripción del menú "Ventanas"

El manejo de ventanas nos permite estar manejando dos partes diferentes del mismo texto a la vez o dos documentos diferentes. Esta es una de las opciones que nustro procesador no está manejando por el momento.

Segunda
Abre una nueva ventana

Cambiar
Nos coloca en la otra ventana o panel abierta.

Dividir
Parte la ventana para visualizar diferentes partes del mismo documento simultaneamente.

Mover
Mueve la ventana hacia donde las flechas lo indiquen.

Ampliar
Permite cambiar las dimensiones de la ventana.

Ocultar
Cierra la ventana en la que estemos trabjanadp.

4.3.7 Descripción del menú Formatos

La forma en que se imprime el documento se puede modificar por medio de este menú. El formato incluye márgenes, encabezados, numeración, pies de página, tabuladores, sangría.

Edita Regleta ...
Nos presenta una ventana donde podemos cambiar los parámetros: margen izquierrdo y derecho, tabuladores, sangría y en un futuro otra columna. También se indica si se alinea el texto al margen derecho.

Nueva Regleta
Crea una nueva regleta con los valores de la anterior.

Regleta Vertical ...
Aquí podemos modificar los sigueintes parámetros: inicio de página, margen superior e inferior, espaciamiento.

Cabecera
Establece los parámetros para los encabezados.

Pies
Establece los parámetro para los pies de página.

Mostrar Pies
Nos muestra en pantalla los encabezados y pies de página.

5 Conclusiones

5 CONCLUSION

Analizando retrospectivamente nuestro trabajo, llegamos a la conclusión de que más que un trabajo fue una buena experiencia.

Nos encontramos con un problema real, con restricciones, dificultades y necesidades reales. Algunas dificultades fueron la falta de información, nuestra habilidad al redactar y la falta de herramientas. Y las restricciones principales fueron el equipo y el tiempo del que disponíamos. Tuvimos grandes apoyos, como algunas herramientas de software ya existentes y algunas opciones de hardware. Y no podemos pasar por alto algunos errores como una mala organización y planeación del trabajo. A pesar de ello, el desarrollo dió como resultado un procesador de palabras de muy alta calidad (aunque no está terminado).

Nuestro trabajo es una base para aquellos de nuestros compañeros que se interesen por microcomputadoras. Y nos sirvió para darnos cuenta y evaluar mejor el trabajo que se requiere al resolver cualquier problema, y sobre todo si se hace por medio de algún sistema de programación.

También nos dimos cuenta de lo importante que fue nuestra formación académica (pues sin ella no hubiéramos resuelto algunos problemas), y de que la creatividad también forma una parte importante en el desarrollo de cualquier elemento. El conjunto de la creatividad con nuestros conocimientos, nos permite pasar de la etapa de aprendizaje a una de productividad, en la que generamos benficios a la gente que nos rodea.

6 Glosario

6 GLOSARIO

PALABRA	SIGNIFICADO
Amigable	es la característica de un sistema que debido a su interfase de usuario se hace facil de aprender y operar
Apple-writer	procesador de palabras para Apple II+ y IIe
Applesoft	versión de BASIC para la microcomputadora APPLE
ASCII	American Standard Code of Information Interchange
Backspace	Nombre utilizado para el caracter que regresa el cursor un caracter hacia atrás
BASIC	Beginner's All Purpose Symbolic Instruction Code lenguaje de programación más utilizado para micros
Best-seller	el mas vendido
Bit	digito binario
Block	bloque o conjunto de sentencias
Block copy	copiado de un bloque
Block move	movimiento de bloques
Buffer	area de memoria de uso temporal frecuentemente utilizado para realizar operaciones con bloques de texto
CAPS-LOCK	Nombre de la tecla utilizada para fijar mayúsculas
Case sensitive	propiedad de la función de búsqueda que permite hacer la distinción entre las palabras escritas en mayúsculas y las palabras escritas en minúsculas
Centrado	formateo en pantalla que coloca una o más palabras dejando el mismo espacio del margen izquierdo que del derecho
Character string	cadena de caracteres
Clicking keyboard	teclado con eco o sonido

Comandos	instrucciones
Debugger	depurador de programación
Delete	nombre de la tecla utilizada para borrar uno o más caracteres
Desk top	interfase de usuario que maneja el modelo familiar de escritorio de oficina
Diskettes	(vease floppys)
Double strike	cualidad de una impresora de matriz de puntos para dar dos pasadas sobre la misma línea para efectos de una mejor impresion
Draft copy	impresion sin formatear
Drives	unidades de disco flexible
Dump	vaciado de pantalla directamente a la impresora
Ensamblador	programa que traduce de un lenguaje de mnemónicos a código de máquina
Error mesage support	función de un sistema de cómputo que le indica al usuario que se ha cometido un error por medio de mensajes a la pantalla
Floppys	discos flexibles
Hardware	lo referente a la electrónica de las computadoras
Head & foot lines	encabezados y pies de página de un documento
Helps (ayudas)	función de un sistema ó paquete que se encarga de sacar de dudas al usuario de forma interactiva
Hint	guia o ayuda breve
Home	extremo superior izquierdo del area de trabajo de un procesador de palabras
Identación	sangrías
Jump features	facilidades que puede ofrecer un editor de textos para agilizar el recorrido de un documento
Justify	función de un procesador de palabras en el que agrega espacios a una línea para dar forma final a un documento

K	1024 caracteres o localidades de memoria
Kernel	núcleo
Lock & Unlock	protección de un archivo contra escritura o borrado de este
Línea de status	linea que indica la funcion que se esta realizando, y en que posición del documento se esta dentro de un editor
Línea guía	linea que orienta al usuario
Macros	reproducción de un solo teclazo de una serie de comandos o caracteres
Mail list	generador de correspondencia
Merge files	combinación de varios archivos o documentos
Modem	circuito utilizado para comunicar computadoras a distancia por medio de modulación y demodulación
Módulo	unidad logica y funcional de un sistema
Mouse	dispositivo externo cuyos movimientos representan desplazamientos del cursor en la pantalla
Multimate	uno de los procesadores de palabras más populares para IBM PC y compatibles
Overlays	manejo de memoria en el cual se utiliza un misma area de memoria para cargar rutinas o datos cuando se les requiera unicamente
Overstrike	modo de edicion en el cual se puede sobreecribir parte de un documento escrito con anterioridad.
Page-down	nombre de una tecla que es utilizada en los editores para mover el cursor una pantalla adelante
Page-up	nombre de una tecla que es utilizada en los editores para mover el cursor una pantalla atras
Password	clave de seguridad que permite el acceso a la información en un sitema de computo

pfs-writer	procesador de palabras de la familia PFS
phrase	frase o conjunto de palabras separadas por un punto
Printing queue	colas de impresion
Procesador dedicado	microprocesador o procesador dedicado totalmente al procesamiento de palabras
Procesamiento de ideas	es una combinación de un procesador de palabras y un aministrador de base de datos para organizar ideas
Prontuario	es un manual abreviado de un sistema, orientado a usuarios familiarizados con su uso
Pull-down menús	menú que al ser activado se despliega de arriba hacia abajo
Quick refernce	(ver prontuario)
RAM	memoria de acceso aleatorio (random access memory)
Rename	cambiar de nombre a un archivo
ROM	memoria de solo lectura (read only memory)
Scrolling	corrimiento horizontal o vertical de un documento a través de la pantalla
Search & replace	funcion del procesamiento de palabras que permite buscar o reemplazar una palabra dada dentro de un texto
Sistema Operativo	programa que permite a una computadora administrar el CPU y sus perifericos
Soft-switches	modalidad de la interfase de usuario que permite habilitar o deshabilitar una función
Software	lo referente a la programación de las computadoras
Software Acoplado	conjunto de módulos o programas que trabajan como sistemas independientes y que pueden compartir información
Software Integrado	conjunto de programas que trabajan de manera integrada y que contemplan varias aplicaciones en un solo paquete
Spelling checker	checador de ortografía

Spoller	programa que permite al CPU trabajar mientras se está imprimiendo
Swap	intercambio de información entre disco y memoria
System implementation languages	familia de lenguajes de nivel intermedio que ofrece facilidades para implementar sistemas operativos.
Tabs	tabuladores
Terminal Inteligente	terminal conectada a una una red de computadoras que puede tener proceso autónomo.
Thesaurus	diccionario de sinónimos
Think-tank	paquete procesador de ideas
Toggle switches	vease soft switches
Tutorial	parte de la documentación de un sistema que enseña a los usuarios manejar un sistema a traves de lecciones muy sencilla
Underline & boldface	tipos de letras subrayadas y negritas
Uppercase	letras mayúsculas
Vertical pitch	longitud vertical de impresión (pitch es una contraccion de characters per inch)
Wild card	caracter comodín
Wild card search	búsqueda por medio de wild-cards
Word	localidad de memoria compuesto de 2 bytes
Word-start	marca de un programa de procesamiento de palabras que se le puede considerar como un estandar
Word-wrap	función de un procesador de palabras que mantiene las palabras completas dentro de una linea en ves de partirlas en sílabas

7 Apéndices

7.1 DEFINICIONES DE FUNCIONES DE PROCESADOR DE PALABRAS

Algunas especificaciones funcionales se definen a continuación. Para estas definiciones se usan constantemente los conceptos de palabra, frase y párrafo. Una palabra es un conjunto de caracteres delimitados por espacios o signos de puntuación, una frase es un conjunto de palabras separadas de otro conjunto por puntos y un párrafo es un conjunto de frases separadas por un fin de linea seguido por un comienzo de frase en el renglón de abajo.

1) Documentación o facilidad de uso.

Es toda la información que existe junto con el procesador, esta información nos es útil al aprender a usar el sistema, o cuando se necesita información más especializada.

Es muy difícil encontrar procesadores fáciles de aprender.

- Ayudas. Son mensajes que despliegan algunos paquetes o sistemas, para que el usuario consulte sus dudas sobre algún tópico especifico, de manera interactiva.

  Casi siempre requieren de accesos a disco, una forma muy usada para solicitarlas es usando la tecla de interrogación en combinación con otra tecla (como Control).

  Otras teclas comunmente usadas son la "H(elp)" y la "F1" y "F10" en la IBM PC.

Tipos de despliegue:
+ mostrados con ventanas que se sobreponen al texto,
+ en pantallas diferentes a la de trabajo, pudiendo ser una sola pantalla o un conjunto jerárquico de ellas.
+ en una o varias líneas destinadas a ese fin.

Tipos de ayudas:
+ el nombre de la función y las teclas con que se realiza,
+ explicación de como se utiliza,
+ explicación y ejemplos de para qué fue hecha o en qué se puede usar.

- Tutorial. Es un programa o manual que sirve para enseñar a usar un sistema, llevando de la mano al usuario por medio de ejemplos sencillos.

Se presentan:
+ en forma impresa,
+ en un programa (generalmente en un disco separado),
+ en archivos de ejemplos, como un anexo.

Y son:
+ breves explicaciones de las funciones,
+ ejemplos en forma de lecciones.

Un sistema se presenta mejor y más completo mientras menos tutoriales necesite pero más incluya.

- Manual. Es un libro de consulta sobre un sistema, en el que está detallado el uso y funcionamiento de éste.

Entre más amplios y mejor organizados sean, un sistema es más sencillo en su aprendizaje y pede ser mejor aprovechado. La referencias técnicas extensas son necesarias para dominar ampliamente funciones especificas.

Los tipos son:
+ manual técnico (reference manual),
+ manual para principiantes (beginers' guide)
+ manual para el tutorial,
+ manual de aplicaciones avanzadas.

Nota: En cualquier procesador de palabras este elemento es muy importante.

- Prontuarios. Es un manual abreviado de un sistema, orientado a usuarios familiarizados con su uso.

  Se presentan:
  + en forma de tarjeta, conteniendo los comandos principales y su sintaxis,
  + en forma de plantilla para sobreponerse al teclado,
  + un conjunto de tarjetas bien organizado en forma de folleto.

  Nota: Están orientados a usuarios avanzados.

- Indices. Es una lista para localización rápida de información dentro de un manual. Este elemento facilita el uso y acceso rápido a las referencias.

  Existen:
  + alfabéticos,
  + por temas,
  + de ejemplos.

- Indicaciones de error.

  Hay de varios tipos:
  + Advertencias o avisos de problables errores por el mal uso de alguna función con necesidad de confirmación,
  + mensajes que no requieren de confirmación,
  + campanazos.

2) Edición del Documento.

Este es el término utilizado por la mayoría de los Procesadores de Palabras, y significa introducir, hacer cambios y correcciones a un texto. Es el conjunto de operaciones con las que se afecta al documento, como insercion de texto, movimiento de bloques o borrado de alguna parte del mismo.

- Funciones de Búsqueda y Reemplazo.

  El software se encarga de buscar palabras o series de caracteres y de poner en su lugar otras palabras o caracteres.

  Estas funciones permiten ahorran mucho tiempo y tecleado.

  En una búsqueda global, en cualquier lugar del texto se lo-

calizan cadenas de caracteres o palabras, y en caso de solicitar un reemplazo se verifica. Por lo general la búsqueda se hace de la posición del cursor hacia el fin de archivo, y se busca un conjunto de caracteres y no palabras.

La forma de pedir esta función es:
+ por teclas de control, usualmente se usa la tecla "F(ind)" o "S(earch)", "L(ocalizar)" o "B(uscar)".
+ Por menús.

La forma de especificar la cadena a buscar es:
+ simplemente tecleando los caracteres,
+ usando delimitadores en la cadena para relacionarla con la función de reemplazo.
+ También existen las búsquedas y reemplazos múltiples, separando las cadenas a buscar por delimitadores.

- Cadenas de caracteres. Para algunos Procesadores de Palabras es posible buscar cualquier conjunto de caracteres, aunque tengan caracteres especiales incluidos.

- Palabras completas exclusivamente. Se buscará un conjunto de caracteres que se encuentren entre delimitadores.

  Formas de hacerlo:
  + Por default,
  + usando dos entradas, una para la cadena de caracteres y otra para opciones, entre ellas ésta.
  + Por medio de una pregunta explícita.
  + Utilizando una función aparte.

- Global en Memoria y Disco. Cuando un procesador puede manejar un documento tan grande que parte de éste está en memoria y otra en disco, a veces puede realizar la búsqueda tanto en las áreas de memoria como en el disco, a lo largo de todo el documento.

  Puede pedirse como una opción en una entrada distinta a la de la cadena (G).

- Pausas para pedir verificación. Es una interrupción durante una cadena de reemplazos, que se produce después

de haber encontrado una palabra para preguntar si se debe de remeplazar o no.

Puede perdirse:
+ por default,
+ con una opción extra (para prender o apagarla según sea el default).
+ Por una pregunta específica.

- Toma en cuenta Mayúsculas y minúsculas.

Algunos programas dan la opción de que la búsqueda de palabras se haga sin tomar en cuenta si fueron escritas con mayúsculas o minúsculas.

Se puede pedir
+ como una opción ("U(pper-case)),
+ con una pregunta específica,
+ usando combinaciones de mayúsculas y minúsculas como por ejemplo: 'profesor' para buscar 'Profesor' y 'profesor'; pero si se usa alguna mayúscula entonces se buscan las palabras exactamente iguales como 'Profesor' para buscar exclusivamente 'Profesor'.

- Búsqueda aproximada. Es una función en la que se buscan cadenas similares a las especificadas por el usuario, esto se logra indicando la raiz común de la palabra y caracteres en lugar de los caracteres de poco interés para la búsqueda.

Tipos de caracteres de sustitución:
+ Caracter de sustitución individual (que sustituye un solo caracter).
+ Caracter de sustitución múltiple (que sustituye cualquier número de caracteres).
+ Caracteres que sustituyen a otros caracteres especiales (Return, Escape o caracteres de control).

Caracteres de Sustitución frecuentemente usados:
+ "*", ".." para cualquier número de caracteres,
+ "!", "?" para un solo caracter,
+ "^" para return.

- Busca N veces. Es una búsqueda global repetitiva.

Generalmente se realiza como una opción (n-número de veces a realizarse).

- Búsqueda en sentido contrario. Es poder buscar de la posición del cursor hacia el principio del documento.

  Formas de hacerlo.
  + Como una opción extra, ejem "B(ackwards)"
  + De acuerdo a una dirección establecida, manejada por el usuario (ejem. ">" hacia adelante o "<" hacia el inicio).

- Búsqueda en varios archivos. Se usa sólo en algunos casos y en el caso específico de MultiMate, esta búsqueda está restringida a las hojas de identificación de los archivos.

- Búsqueda por Marcas. Algunos procesadores de palabras son capaces de manejar varias marcas dentro de un texto, éstas no se imprimen y a veces ni se ven, sólo sirven para localizar rápidamente un punto importante, previamente especificado.

  La manera de realizarlo es por opciones específicas.

  Tipos de marcas:
  + fijas (principio, fin),
  + calculadas (décimas, octavas del documento),
  + fijadas por el usuario,
  + caracteres de control usados como marcas, ejemplo: "salto de página", "comienzo de subrayado".

  Identificación de las marcas:
  + números ("0" al "9"),
  + nombres formados por cualquier conjunto de caracteres,
  + nombres preestablecidos, como "inicio", "fin de bloque" "margen derecho", etc.

- Inserción. Esta es una de las funciones de un Procesador de Palabras más usadas; junto con la función de borrado, constituyen el núcleo de un Procesador de Palabras. Nos permite añadir caracteres en cualquier lugar del documento.

Esta función se puede pedir a través de un menú, como

cualquier otra función (usando "I(nsert)", "N" o "A(dd)" o puede ser automática, insertando cada que se teclea texto dentro de una edición.

- Inserción sencilla. Se lleva a cabo con sólo poner el cursor en el lugar donde queremos añadir el caracter y teclearlo, y en algunas ocaciones se necesita algo extra. Este tipo de inserción respeta lo anterior dejándolo como estaba.

  Push ahead, cada vez que se teclea un caracter empuja a los siguientes hacia delante, sin perder el contenido anterior.

  Paste and Glue, cuando se comienza a insertar, se parte la línea en el punto donde se va a insertar, quitando lo que resta del renglón, cuando la inserción termina se pega la parte que se movió a continuación

  Por buffer intermedio, consiste en guardar todos los caracteres a insertar en una área auxiliar de memoria (buffer), antes de ponerlos dentro del documento. Se realiza, generalmente, por teclas de control ("+" en MultiMate) o en forma continua (donde cada que se teclea un texto se recorre el resto del mismo).

- Insercion con escritura sobre lo anterior. Se realiza de manera similar a la anterior, pero borrando lo que se encontraba en el lugar donde se inserta texto nuevo.
- Inserción de otro archivo dentro del Documento. Es la capacidad que tiene un Procesador de Palabras para incluir el contenido de otros archivos al documento, algunos procesadores permiten incluir otro archivo parcial-

mente por medio de marcas, y otros solamente archivos completos.

Generalmente se inserta en el lugar donde está el cursor.

+ Total.
+ Parcial.

- Inserción de Bloques. Un bloque es un grupo de líneas de texto, y esta función nos permite insertar ese conjunto en la posición deseada.

- Borrado. Nos permite quitar partes o caracteres de un documento.

- Por caracter. Generalmente, con una sola tecla se puede borrar el caracter de la posición del cursor, eliminándolo así del texto.

Formas de hacerlo:
+ Hacia adelante,
+ hacia atrás.

Se realiza con las teclas de movimiento, "Delete", "Backspace" o Ctrl-"D"

- Por palabra. Borra desde la posición del cursor hasta un delimitador.

Se usa Ctrl-"W" como tecla de función.

- Por palabra a la izquierda. Igual que el anterior pero buscando el delimitador hacia la izquierda.

- Por línea. Una alternativa de solución es usando una sola tecla de control.

- Por bloques. Es cuando un bloque (conjunto de líneas) se puede borrar completo con una sola intrucción.

Se puede hacer:
+ por marcas fijadas independientemente a la función,

+ por una función específica que pide el principio y el fin del bloque.

- Por páginas. La página consiste en un número de líneas, de acuerdo al formateo de salida, y esta función checa el número de líneas que forman una página y las borra.
- Hasta el final de la línea
- Por oración
- Utilizando un buffer intermedio. Lo que se va a borrar se puede depositar en el buffer para después tomar de éste lo que se va a insertar en otro punto del texto.
- Continuamente. Significa que el borrado pueda hacerse desde cualquier modo de edición.

  Ya sea el caracter anterior al cursor o el del cursor.

  Se da el caso de:
  + Backspace,
  + Delete.

- Reproducción.

  Generalmente se usa por medio de una función especial o por opciones especiales.

  Se realiza:
  + por marcas, haciendo referencias a marcas ya definidas,
  + por buffers, donde se guardan los caracteres que fueron borrados,
  + por bloques, que se definen en el momento de pedir la función.

- Movimiento de Bloques. Significa cambiar un bloque de su posición original, a otra deseada. Esta función requiere que se le indique el inicio del bloque y el final de éste, y después el nuevo lugar a donde colocarlo.

- Copia de Bloques. En este caso, el conjunto de lineas, pueden ser copiado varias veces, dejando el conjunto original donde estaba, y depositando el texto a copiar en la posición del cursor o de otra marca definida.
- Escritura de Bloques a un archivo nuevo. El bloque se copia a un archivo nuevo, y aunque muy útil, no es una característica común.
- Movimiento de Bloques entre archivos. Es poder tomar conjuntos de líneas de un archivo en disco y pasarlo a otros archivos.

- Movimiento de columnas. Columna es una forma de formateo de la información que considera varios campos de escritura, como lo que se hace en los periódicos, es decir el texto que se encuentra entre dos márgenes. Y la función permite cambiar la información de una columna, y reformatear automáticamente el resto de la información.

  Formas de hacerlo:
  + Cambiando los márgenes, recalculando el contenido de cada columna.
  + Cambiando el orden de las columnas.

- Cambio de mayúsculas/minúsculas. Capacidad de cambiar una serie de letras de mayúsculas a minúsculas o viceversa.

  Formas de hacerlo:
  + Seleccionando una función y pasando el cursor por encima.

3) Manejo del Cursor y Pantalla.

Los movimientos sencillos y algunos más complicados se llevan a cabo con teclas especiales, tales como las flechas y las teclas <Backspace>, <Page-up>, <Page-down>, <Home>, etc. Otros movimientos requieren de una función o una combinación de teclas.

- Movimientos del cursor en pantalla. Significa que el programa llevará al usuario a otro lugar, otra página, otro renglón. Así se coloca sobre un error y rápidamante se corrige.
  - Movimiento por tabulador.
  - Movimiento por caracter. Por ejemplo: flechas y <Backspace>
  - Movimiento por palabras. Por ejemplo: <Ctrl-flechas> y teclas de <manzana-flechas>.
  - Movimiento por oración.
  - Movimiento por párrafo.
  - Movimiento por pantalla. Esta función sirve para ver la continuación de un texto, dando un brinco de una pantalla que tiene como tamaño estándard veinticuatro renglones. El movimiento puede ser hacia el fin o el principio del documento.
  - Movimiento por página. Es similar al anterior pero con la diferencia de que el tamaño de pantalla lo puede variar el usuario. Por ejemplo se realiza con: <Ctrl-P>, <Page-Up> o <Page-down>, dependiendo del procesador.
- Movimiento al comienzo y al final del documento. Este ti-

po de movimiento nos coloca al inicio o al final y es útil especialmente para corregir errores en las declaraciones iniciales. Se puede hacer por ejemplo con: <Ctrl-B>, <Ctrl-E>, <End>, <Home>.

  - Movimiento al fin y comienzo de la pantalla. Se puede hacer con un conjunto de caracteres o con <Home> y <End>.
  - Movimiento al comienzo y fin de la linea. También se usa con una combinación de caracteres.
  - Movimiento a una línea o página especificada. Se realiza por medio de marcas
  - Movimiento usando marcas.

- Manejo del Ratón, o dispositivos equivalentes. Es la capacidad de utilizar dispositivos localizadores para facilitar la edición.
- División de la pantalla. Significa que se pueda ver dos partes del texto a la vez, dividiendo la pantalla en varias partes.

  Se puede presentar:
  + Visualizando dos partes del mismo archivo.
  + Manejando dos archivos.

- Formateo. Si el texto se modifica de alguna manera, el procesador se encarga de hacer las operaciones necesarias para que la salida no se afecte en su formateo.

  Tipos de Formateo:
  + Automático. Se ejecuta cada que se teclea el texto.
  + Por bloque. Marcando un inicio y un final del bloque a formatear.
  + Por linea, a la derecha, a la izquierda o al centro.
  + Por medio de comandos escritos dentro del texto, que pueden ser invisibles y ejecutarse en tiempo de edición.

- Márgenes ajustables.

Se pueden ajustar:
+ Durante la edición.
+ En forma global.
+ Por líneas de formato en cualquier renglón.

- Manejo de diferentes tamaños de pantalla (40 a 80 columnas). Se usa para compatibilidad con máquinas que solo pueden manejar 40 columnas.

- Justificación a la derecha o izquierda. Es cargar todo el renglón hacia el margen derecho o hacia el izquierdo.

Se puede hacer:
+ Global.
+ Por bloque.
+ Por línea.

- Centrado. Es dejar el mismo número de espacios hacia el margen derecho que hacia el margen izquierdo.

Se puede hacer como el anterior o con una marca de centrado.

- Subrayado y negritas en pantalla. Para que ésto sea posible se necesita de un monitor y de un generador de caracteres que pueda hacer subrayado; característica no muy común.

- Paginación en Pantalla. Es ver en la pantalla el número de hoja impresa donde se encontrará lo que se ve en pantalla.

Se puede hacer:
+ Contando las líneas exclusivamente.
+ Tomando en cuenta el tamaño de hoja y márgenes superior e inferior se genera la hoja en la pantalla, incluyendo líneas en blanco, encabezados y pies de página.

Se pueden ajustar:
+ Durante la edición.
+ En forma global.
+ Por líneas de formato en cualquier renglón.

- Manejo de diferentes tamaños de pantalla (40 a 80 columnas). Se usa para compatibilidad con máquinas que solo pueden manejar 40 columnas.

- Justificación a la derecha o izquierda. Es cargar todo el renglón hacia el margen derecho o hacia el izquierdo.

  Se puede hacer:
  + Global.
  + Por bloque.
  + Por línea.

- Centrado. Es dejar el mismo número de espacios hacia el margen derecho que hacia el margen izquierdo.

  Se puede hacer como el anterior o con una marca de centrado.

- Subrayado y negritas en pantalla. Para que ésto sea posible se necesita de un monitor y de un generador de caracteres que pueda hacer subrayado; característica no muy común.

- Paginación en Pantalla. Es ver en la pantalla el número de hoja impresa donde se encontrará lo que se ve en pantalla.

  Se puede hacer:
  + Contando las líneas exclusivamente.
  + Tomando en cuenta el tamaño de hoja y márgenes superior e inferior se genera la hoja en la pantalla, incluyendo líneas en blanco, encabezados y pies de página.

- Tabulador en pantalla. Tabulador es una marca que nos facilita el movimiento rápido del cursor a lo largo de una línea. Con esta función se visualiza una serie de signos que representan las marcas del tabulador.

  Existen:
  + Globales.
  + Por página.
  + Múltiples e independientes de las páginas.

  Información que contienen:
  + Márgenes derecho e izquierdo.
  + Marcas de tabulador.
  + Columnas numeradas.
  + Sangría.
  + Espaciamiento entre líneas.
  + Columna en la cual se encuentra el cursor.

- Movimiento horizontal y vertical de la pantalla. Dado que generalmente los textos a modificar son mayores que lo que se puede ver en la pantalla, se necesita la habilidad de mover el texto en la pantalla, hacia arriba y hacia abajo o de un lado hacia otro, en forma continua para nuestra vista.

  Para movimiento vertical existe por:
  + Renglón.
  + Página.

  Para movimiento horizontal se hace por:
  + Carácter.
  + Número fijo de caracteres.

- Encabezados y pies de página. Son mensajes que aparecen al principio y al final de todas las páginas. Por ejemplo el nombre del documento.

- Manejo de varios conjuntos de caracteres. Capacidad de simular los diferentes tipos de letras que se usan en la impresión.

- Cambio automático de renglón. Evita la partición de palabras, para dar más claridad al texto.

4) Manejo de Memoria, de Disco y Archivos.

Características especiales de cada procesador de palabra que nos indican las capacidades de memoria y la forma en que se maneja el disco.

Muchas de las siguientes funciones se realizan a través de un paquete de utilerías, fuera del programa de edición.

- Posibilidad de crear archivos tipo ASCII, o de manejarlos, para poder comunicarse con otros programas, y poder usar también sus salidas como textos a formatear.

  Generalmente se pide desde el menú principal.

- Grabación continua a disco. Se usa cuando se manejan archivos muy grandes y cuando los programas hacen continuamente actualizaciones para prevenirse contra fallas eléctricas.

  Se realiza:
  + Por cada página (MultiMate).
  + Por cada edición de un párrafo (Think Tank).

- Borrado de archivos.

- Cambio del nombre de los archivos.

- Visualización del directorio de archivos en un disco.

  Existen diferentes tipos de directorios:
  + Los que manejan subdirectorios.
  + Los que incluyen funciones de borrado, selección y cambio de nombres.

- Protección y desprotección.

- Uso de contraseñas.

- Combinación de archivos.

Los tipos de combinación pueden ser:
+ Uniendo texto.
+ Uniendo archivos de comandos.
+ Unión con base de datos (Mail List), reportes (spreed-sheet), gráficas.

- Tamaño del archivo.
  - Limitado solamente por la memoria RAM.
  - Limitado solamente por la capacidad del almacenamiento en disco.
- Indicador de espacio restante en disco. Se puede dar un número de páginas, de sectores, de Bytes o de palabras que caben en el disco.
  Se puede realizar cuando se pide el directorio de un disco o por alguna función en especial.
- Recuperación del error: "disco lleno".

5) Formateo de la Impresión.

Son una serie de funciones que permiten que la salida tenga una mejor presentación. Muchas de éstas dependen de la impresora que se usa.

El formateo del documento se realiza generalmente un momento antes de la impresión, y además este proceso genera las secuencias que interpretará la impresora; por ello se nota claramente que depende de la impresora que se esté usando, por tanto, en esta parte se debe de analizar cómo se comunica el sistema con la impresora (por ejemplo: si puede manejar distintos tipos y desde diferentes puertos de comunicación de la computadora).

Formas de especificación del formateo:

+ Por medio de comandos en el texto (".page", ".lm", etc).
+ Por medio de una página o lista de parámetros con valores estándar que se pueden modificar o aceptar.

Comunicación con la impresora:
+ A través de una defininición de las caracteristicas de la impresora hecha por el usuario,
+ A través de una lista de impresoras que conozca el sistema y que el usuario escoja.
+ A través de una tabla de traducción de caracteres que el usuario pueda modificar (MultiMate).

- Altura del renglón. Generalmente se manejan 1/6" y 1/8".

Se refiere al número de caracteres por pulgada, siendo los más comunes 10 (pica) y 12 (elite), y los límites 4 y 30.

Algunos de los mejores programas, permiten hacer estos cambios en cualquier momento sin detener la impresión.

- Manejo de hojas sueltas. Si se están usando formas continuas, el programa al encontrarse cerca del fin de página, automáticamente avanza al principio de la siguiente. Cuando se usan hojas sueltas, el programa debe de provocar una pausa hasta que la hoja nueva, esté en la posición correcta.

La forma de especificarlo es a través de un comando general.

- Numeración de las hojas. Se imprimen automáticamente los números de páginas, ya sea en los encabezados o en los pies de página. Generalmente se puede escoger entre usar números de página o no.

Se realiza:
+ Interpretando mensajes en los encabezados o pies de página (PFS Writer).
+ Comenzando desde un número especificado.
+ Limitando a una palabra de mensaje ("page") o dejando abierto para cualquier otro mensaje.

- Manejo de Encabezados y pies de página. Al igual que en las funciones de pantalla, se maneja un conjunto de líneas que aparecerán en el principio o fin de cada página impresa. Los usos más comunes son para el número de página y para el nombre del documento.

  Pueden ser:
  + Una o varias líneas.
  + Locales o globales.

  Y pueden estar justificados o no.

- Control de pausa. Es un comando que obliga a la impresora a detenerse, de manera que se pueda ajustar para uso posterior.

  Tipos de Pausa:
  + Para insertar texto durante la impresión.
  + Para supervisar el funcionamiento y cambiar el papel.

- Impresión sin formatear. Esta es una impresión idéntica a lo que se muestra en pantalla. Sirve cuando solo se quiere checar o modificar algo sobre papel sin tardarse tanto como cuando se formatea.

  Tipos:
  + En los procesadores de palabra que formatean durante la impresión, consiste en una impresión sin formatear, tal como se ve en pantalla.
  + En los procesadores que formatean durante la edición y que hacen impresiones de calidad, esto es, imprimen todo dos veces, esta función consiste en imprimir una sola vez.

- Salida opcional a disco. Se usa en los procesadores en los que no se ve en la pantalla el texto formateado. Una vez que se manda a disco se puede ver tal como se verá en papel.

  Impresión secuencial de varios archivos. Con esta función se pueden imprimir varios archivos encadenados, lo que nos

es útil si el documento se necesitó dividir en partes, por limitaciones de memoria.

Formas de hacerlo:
+ Por comandos específicos (continue print)
+ Por un comando dentro del texto (include)
+ con una opción a la impresión.

- Justificación. Es agregar espacios en las líneas para acomodarlas en su renglón.

Se puede pedir:
+ por bloque
+ por línea
+ por párrafo
+ todo el documento

Formas de pedirlo:
+ por marcas
+ con caracteres
+ con comandos específicos

- Al centro. Automáticamente se coloca una palabra o un grupo de éstas entre el margen izquierdo y el derecho de la impresión
- A la derecha. Toda la línea se carga hacia el margen derecho, rellenando con espacios los lugares que queden vacíos.
- A la izquierda. Función similar a la anterior, pero corriendo toda la línea hacia la izquierda.
- A ambos lados. En este caso, todas las palabras de la línea se acomodan en el espacio que tiene, de manera que ocupa desde el margen derecho, hasta el izquierdo. Para lograr ésto, algunas palabras se separan entre sí por más de un espacio.
- Manejo de micro espacios. Significa que el espacio ocupado por un caracter es proporcional a su figura. Esto ca-

racterística contrasta con monoespaciamiento (monospacing), donde todos los caracteres ocupan espacios idénticos. Muchas impresoras pueden manejar espacios proporcionales, pero son pocos los procesadores que lo contemplan.

  - No justificar. El texto aparece sin formatear, como se encuentra realmente. Es útil cuando se maneja información tabular o dibujos.

- Control de líneas separadas de su parrafo por fin de página
- Márgenes. Limitan el espacio que el texto puede utilizar en cada hoja impresa.
  - Derecho.
  - Izquierdo.
  - Superior.
  - Inferiores.
  - Varios márgenes. Capacidad de poder manejar varios límites de espacio.
- Copias múltiples. El sistema, imprimirá el documento varias veces, pero el usuario deberá decirle cuántas veces.
- Manejo de colas de impresión.
- Impresión simultánea con edición. Sucede cuando el sistema puede realizar otras funciones o tareas mientras sigue imprimiendo. Aunque no todas las configuraciones lo pueden realizar, se les añade un área intermedia para lograrlo.
- Impresión por páginas. Algunos procesadores pueden seleccionar páginas para imprimir, característica útil cuando

por alguna razón una impresión previa se vió interrumpida o cuando la corrección solo se hará en una página.

- De cualquier página aislada. Cuando se puede imprimir una sola página.
- A partir de cualquier hoja. De esta forma la impresión no se tiene que efectuar desde el principio, sino que apartir de una página especificada.
- Salto condicional o incondicional de página. En algunos casos, el usuario puede tener la facilidad de comenzar una página nueva, si así lo desea. Por ejemplo el comienzo de un capítulo nuevo.

- Impresión de líneas o párrafos. Algunos sistemas manejan el tipo de impresión de párrafos o los formatean sin necesidad de comandos específicos para su impresión.
  - Sangrías.
  - Espaciamiento entre líneas.
  - Líneas entre párrafos.
- Manejo de Tabuladores.
  - Fijar una marca del tabulador.
  - Borrar una marca.
  - Tabuladores múltiples.
- Facilidad para meter caracteres especiales en el texto. Estos caracteres no se incluyen genralmente en el texto, pero son parte del conjunto ASCII de caracteres, y son necesarios ocasionalemente para utilizar completamente las características de algunas impresoras.
- Subíndices y superíndices. Al igual que los caracteres

especiales, necesita de algunas características especiales en la impresora a usar.

- Subrayado. Esta es una de las funciones más comunes en los documentos, por ello casi todos los procesadores son capaces de realizarla.
- Negritas, itálicas, doble ancho, comprimidas, enfatizadas, subrayado de negritas. Una característica en las impresoras actuales es poder manejar varios tipos de letras, por ello es importante manejarlos, y tomar en cuenta cualquier otra de sus facilidades.
- Cambio de tipos. El cambio de tipos en las impresoras, se puede hacer algunas veces por software, como en las impresoras de matriz de puntos.
- Sobreescritura. Nos permite la impresión de un caracter sobre otro. Es muy útil en lenguajes diferentes al inglés donde algunos caracteres se forman por dos signos, como acentuaciones.

6) Otras

- Intrucciones para anular el comando anterior. Con esta función podemos regresar al estado anterior de la edición, sin tener que realizar tantos pasos como los realizados en forma errónea.
- Teclado con echo. Para algunos procesadores, esta función les permite hacer un ruido adicional cada vez que se oprima cualquier tecla.
- Manejo de glosarios. Es la capacidad que tienen algunos

procesadores de palabras para reproducir una frase frecuentemente usada de la misma manera que se reproduciría un bloque, con sólo oprimir algunas teclas.

- Checador de ortografía. Es otro programa que verifica la ortografía de cualquier palabra, comparándola con una lista de 10,000 a 45,000 palabras escritas correctamente. Si alguna palabra del documento no aparece en la lista significa que está mal deletreada o que no se incluyó. El checador hace una lista de las palabras (que no aparecen en el diccionario) y su posición, para ser corregidas o incluídas al diccionario posteriormente.
- Checador de gramática. Es similar al checador de ortografía, y generalmente se encuentra separado del procesador de palabras. Nos indica posibles errores, de acuerdo a listas de palabras relacionadas con las funciones que puedan tener dentro de una gramática, la del idioma inglés.
- Manejo de Macros. Un macro, es la sustitución de una serie de funciones y actividades que se realizan en el texto, con solo unas cuantas teclas. Generalmente existe una etapa de definición, otra de corrección, otra de borrado y otra de ejecución o sustitución.
- Indice de sinónimos o ideas relacionadas. Que servirán como esqueleto vertebral, para la clasificación y recuperación de informacion.
- Generador de Correspondencia. Si existe esta función, que generalmente es un programa aparte, se unen dos archivos, uno que contiene el documento y otro los nombres y datos que

se insertarán al primero en areas especificas, generando
así una lista de cartas similares entre sí.

7.2 Tabla estadística de funciones (MUESTRA = 90 PROCESADORES)

Manejo de cursor y de pantalla

Característica	si	no
scrolling vertical	82	8
scrolling horizontal	50	40
brinco por pantalla	80	10
brinco por página	81	9
brinco por pantalla anterior	80	10
brinco a página anterior	80	10
brinco al inicio del documento	78	12
brinco al fin del documento	81	9
brinco por palabra	6	10
brinco por oración	5	11
inicio/fin del espacio de trabajo	15	1
inicio/fin de la linea	65	9
mitad de la pantalla	42	32
word-wrap	67	7

Funciones de edición características de inserción

Característica	si	no
por caracter	90	0
por bloque	78	12
frases por tecla	3	14
sobreescritura	10	6

por corrimiento	7	9
partir y pegar una linea	4	12
uso de buffer intermedio	3	13
por linea	66	8
por palabra	63	11

Manejo de bloques

Característica	si	no
movimiento de bloques	73	1

Características de borrado

Característica	si	no
por caracter	14	2
por palabras	73	17
por líneas	76	14
por bloques	83	7
por pantalla o página	49	41
por oración	4	12
borrar continuamente	2	14
lo restante de una linea	62	12
buffers de borrado	33	41
recuperación después del borrado	43	47

Características de Busqueda y Reemplazo

Característica	si	no
encontrar una frase en cualquier lugar	6	10
encontrar y reemplazar con autorización	10	14
encontrar y reemplazar en todo el texto	4	10
encontrar y reemplazar en todo el texto	9	7
busquedas y reemplazo por marcas	7	9
sin considerar si son may. o min.	8	8
búsqueda un número dado de veces	66	24
unicamente	65	9
cadenas de caracteres	72	2
cadenas aproximadas	38	36

Características de Formateo en pantalla

Característica	si	no
partición de pantalla	18	72

Características de Formateo de Impresión

Característica	si	no
márgenes	74	16
tabuladores	80	10
subrayado	50	24
centrado	60	14
ancho/largo de página	61	13
formateo del texto completo	9	7
formateo de partes del texto	8	8

Características de manejo de memoria y disco

Característica	si	no
copiado	72	2
mezclado	70	4

7.3 Listado de la estructura de programa en Pascal

```
{$setc apple:=2}
{$V-}
{ Version de la interfase de usuarios que no emplea las ventanas
 que maneja directamente el mousegraphics tool kit, para ahorrar
 4K de memoria de datos y ensamblador                              }

PROGRAM m;                  {interfase sin ventanas, ARCHIVO: TESIS6:M1}
                            {Creado: Lunes 27 de Octubre del 86}
                            {Ultima modificacion: Martes 18 de Noviembre del 86}


  USES
    {$u tesis1.lib}  mgrkit, pepo, cargaf, declara, inicia,
    {$u libsilabeo.code} silabeo,
    {$u edita.code} edita,
    {$u movimiento.code} movimiento;

  CONST
    Pagina_2 = 16384;
    Nomb_Inicial = 'Tariacuri';
    Doc_Guardar = 4;
  TYPE
    T_Caracter = SET OF char;

    Adr_Cheat = RECORD
      CASE boolean OF
        true : (int : integer);
        false : (ptr : ^integer);
       END;


  VAR
    cheat : Adr_Cheat;
    Con_Lin_Fis : integer;
    oprimio_esc : boolean;



        {borrar caracteres}
PROCEDURE maneja_delete;
  CONST
    TamLin = 90;
  VAR
    Home_Aux, Lim, PosAux : integer;
    Seg : Tipo_2_Lineas;
  BEGIN
                {calcula pincipio de la linea}
    IF pos (C_Sangria, LinT) <> 0 THEN
      Lim := Regleta.Sang
    ELSE
      Lim := Regleta.MarIzq;
                {si no esta al principio}
    IF CursorC.Log > Lim THEN
      BEGIN
        procesado := false;
                {corrige apuntadores}
        CursorC.Fis := pred (CursorC.Fis);
        CursorC.Log := pred (CursorC.Log);
        PosCur.X := PosCur.X - Ancho;
```

```
            {borra un caracter}
      delete (LinT, CursorC.Fis, 1);
            {refrescar la linea}
      refresca_LinT;
    END
  ELSE IF (NOT Dialogo) AND (CursorL.Log > 1) THEN
    {o si esta al principio pero no es dialogo y
     no esta al principio del archivo}
    BEGIN
      procesado := false;
      Home_Aux := Home.Fis;
      insert (Marca_Cursor, LinT, CursorC.Fis);
      IF NOT Procesado THEN
        IF length (LinT) > TamLin THEN
        {si no ha sido procesado y la linea de trab. es mayor que TamLin}
          BEGIN
            {partirla en dos y guardar segundo pedazo}
            separa_lineas (LinT, Seg, TamLin DIV 2);
            guarda_linea (Seg, CursorL.Fis);
          END;
            {si tiene sangria, quitarsela}
      PosAux := pos (C_Sangria, LinT);
      IF PosAux <> 0 THEN
        delete (LinT, PosAux, 1);
            {guardar linea de trabajo}
      guarda_linea (LinT, CursorL.Fis);
      busca_cursor;
            { y tomar anterior}
      toma_linea (LinT, pred (CursorL.Fis));
            {si tiene CR, quitarselo}
      PosAux := pos (CR, LinT);
      IF PosAux <> 0 THEN
        delete (LinT, PosAux, 1)
      ELSE
        BEGIN
            {si no tiene CR, quitarle el ultimo caracter si es diferente
             de Sangria y de espacio suave (porque ya esta al principio)}
          PosAux := longitud (LinT);
          IF (LinT[PosAux] <> C_Sangria[1]) AND
             (LinT[PosAux] <> Esp_Suave[1]) THEN
            delete (LinT, PosAux, 1)
          ELSE
            PosAux := succ (PosAux);
        END;
            {formatear}
      forma_anterior (LinT, pred (CursorL.Fis), PosAux);
      busca_cursor;
            {y pintar}
      IF (Home_Aux <> Home.Fis) OR (CursorL.Fis = Home.Fis) THEN
        pinta_vent (Home.Linea, Y_Ini_Vent + TamReng)
      ELSE
        pinta_vent (CursorL.Linea^.Ant, PosCur.Y - TamReng);
            {tomar linea de trabajo}
      toma_linea (LinT, CursorL.Fis);
    END
  ELSE
    write (chr(7));
END; {maneja_delete}
```

```
  {separa los distintos tipos de teclasos que pueden llegar al procesador}
PROCEDURE maneja_teclas;
  BEGIN
              {quita el acento de la pantalla}
    IF acentuado THEN
      delete (LinT, CursorC.Fis, 1);
              {seleccion de teclazos}
    IF chr (Event.char1) IN Car_Movimientos THEN
      maneja_movimientos
    ELSE IF Event.char2 <> No_Manzana THEN
      maneja_manzana
    ELSE
      CASE Event.Char1 OF
        esc : IF Dialogo THEN oprimio_esc := true;    {nada}
        Del : maneja_delete;
        Tab : inserta_Tab;
        Return : inserta_CR;
        acento : inserta_acento;
        OtherWise
          IF Dialogo THEN
            IF (CursorC.Log < Regleta.MarDer) AND
               (length (LinT) < Tam2Lin - 10) THEN
              inserta
            ELSE
              write (chr(7))
          ELSE
            inserta;
      END; {CASE}
            {apaga la bandera de acentuado}
    IF (Event.Char1 <> acento) OR (Event.Char2 <> No_Manzana) THEN
      acentuado := false;
  END; {maneja_teclas}


PROCEDURE guar_est_cur;
  VAR
    partir : boolean;
    Event_Aux : Type_Event;
  BEGIN
      {quitar el acento si lo hay}
    IF acentuado THEN
      BEGIN
        delete (LinT, CursorC.Fis, 1);
        acentuado := false;
      END;

      {formatear}
    IF NOT Procesado THEN
      BEGIN
        forma_anterior (LinT, CursorL.Fis, CursorC.Fis);
        busca_cursor;
        toma_linea (LinT, CursorL.Fis);
      END;
    partir := true;
    IF pos (C_Sangria, LinT) <> 0 THEN
      IF CursorC.Fis = Regleta.Sang THEN
        partir := false;
    IF CursorC.Fis > longitud (LinT) THEN
```

```
      BEGIN
        IF (CursorL.Log <= Cont_Lineas.Log) AND (pos (CR, LinT) <> 0) THEN
          BEGIN
            Event_Aux := Event;
            Event.Char1 := Flech_Adelante;
            Event.Char2 := 0;
            maneja_movimientos;
            Event := Event_Aux;
            partir := false;
          END;
      END;
    IF partir THEN
      inserta_CR;
    guarda_linea (LinT, CursorL.Fis);
        {guarda el estado actual de la pantalla}
    Primera_estado := Primera;
    Ultima_estado := Ultima;
    Home_estado := Home;
    CursorL_estado := CursorL;
    CursorC_estado := CursorC;
    PosCur_estado := PosCur;
    Reg_estado := Regleta;
    Con_Lin_Fis := Cont_Lineas.Fis;
    {guarda las lines que estan en mem. principal}
    WHILE Primera.Fis <> CursorL.Fis DO
      BEGIN
        IF NOT swapeos (Primera.Linea^.En_Tex, 1, 0, Guarda_Abajo) THEN
          salir_error ('--- guar_est_cur, guardando primeras lineas ---');
        dispo (Primera.Linea);
      END;
    WHILE Primera.Linea <> Ultima.Linea DO
      BEGIN
        IF NOT swapeos (Ultima.Linea^.En_Tex, 1, 0, Guarda_Arriba) THEN
          salir_error ('----- guarda_estado, guardando lineas ----');
        dispo (Ultima.Linea);
      END;
    IF NOT swapeos (Ultima.Linea^.En_Tex, 1, 0, Guarda_Arriba) THEN
      salir_error ('----- guar_est_cur, guardando ultima linea ----');
  END; {guar_est_cur}


PROCEDURE guarda_estado;
  BEGIN
        {quitar el acento si lo hay}
    IF acentuado THEN
      BEGIN
        delete (LinT, CursorC.Fis, 1);
        acentuado := false;
      END;
        {formatear}
    forma_anterior (LinT, CursorL.Fis, CursorC.Fis);
    busca_cursor;
    pinta_vent (Home.Linea, Y_Ini_Vent + TamReng);
        {guarda el estado actual de la pantalla}
    Primera_estado := Primera;
    Ultima_estado := Ultima;
    Home_estado := Home;
    CursorL_estado := CursorL;
    CursorC_estado := CursorC;
```

```
    PosCur_estado := PosCur;
    Reg_estado := Regleta;
        {guarda las lines que estan en mem. principal}
    WHILE Primera.Linea <> Ultima.Linea DO
      BEGIN
        IF NOT swapeos (Ultima.Linea^.En_Tex, 1, 0, Guarda_Arriba) THEN
          salir_error ('----- guarda_estado, guardando lineas ----');
        dispo (Ultima.Linea);
      END;
    IF NOT swapeos (Ultima.Linea^.En_Tex, 1, 0, Guarda_Arriba) THEN
      salir_error ('----- guarda_estado, guardando ultima linea ----');
  END; {guarda_estado}


PROCEDURE ir_principio;
  VAR
    i, Cont, Cont_Bl, Cont_Li : integer;
    Ban : boolean;
  BEGIN
    Cont := pred (Primera.Fis);
    Cont_Bl := Cont DIV NumLin;
    Cont_Li := ConT MOD NumLin;
    FOR i := 1 TO Cont_Bl DO
      BEGIN
        IF NOT swapeos (i, NumLin, 0, trae_de_abajo) THEN
          salir_error ('-----ir_principio, error al traer bloque-----');
        IF NOT swapeos (i, NumLin, 0, guarda_arriba) THEN
          salir_error ('-----ir_principio, error al guardar bloque-----');
      END;
    IF Cont_Li > 0 THEN
      BEGIN
        IF NOT swapeos (i, Cont_Li, 0, trae_de_abajo) THEN
          salir_error ('-----ir_principio, error al traer lineas-----');
        IF NOT swapeos (i, Cont_Li, 0, guarda_arriba) THEN
          salir_error ('-----ir_principio, error al guardar lineas-----');
      END;
  END; {ir_principio}


PROCEDURE main_loop;
  FORWARD;


PROCEDURE esp;
  VAR
    r : rect;
  BEGIN
    ln;
    drawtext ('Oprima <Return> para continuar');
    doreadln;
    haz_rect (r, 7, 56, 552, 183);
    setpattern (Negro^);
    paintrect (r);
    setpattern (Blanco^);
  END;
```

```
PROCEDURE BG_parpadeo;
  FORWARD;


FUNCTION pide_dialogo (mensaje : string; Resp_Posi : T_Caracter) : char;
  VAR
    Carac : Type_Event;
    x1, y1 : integer;
    PosCur_Aux : Point;
    Salir_Dialogo, Cur_Pren_Aux : boolean;
    s : string[1];
  BEGIN
    WITH MainPort.PenLoc DO
      BEGIN
        x1 := X;
        y1 := Y;
      END;
    moveto (habit, Seg_Dialogo);
    drawtext (mensaje);
    PosCur_Aux := PosCur;
    PosCur := MainPort.PenLoc;
    Cur_Pren_Aux := Cursor_Prendido;
    Resp_Posi := Resp_Posi + [chr (esc)];          {siempre se acepta el esc}

    Salir_Dialogo := false;
    s := '?';
    FlushEvents;
    REPEAT
      GetEvent (Carac);
      IF Carac.evt_kind = keydown THEN
        BEGIN
          IF Cursor_Prendido THEN
            parpadea_cursor;
          IF chr (Carac.Char1) IN Resp_Posi THEN
            Salir_Dialogo := true
          ELSE
            BEGIN
              s[1] := chr (Carac.Char1);
              write (chr (7));
              drawtext (s);
            END;
          parpadea_cursor;
        END
      ELSE
        BG_parpadeo;
    UNTIL Salir_Dialogo;

    pide_dialogo := chr (Carac.Char1);

    moveto (habit, Seg_Dialogo);
    drawtext (
    PosCur := PosCur_Aux;
    Cursor_Prendido := Cur_Pren_Aux;
    moveto (x1, y1);
  END; {pide_dialogo}


PROCEDURE doApple;
```

```
  FORWARD;


        {dialogo para pedir nombre de archivo}
PROCEDURE pide_nombre (Palabra : string);
  VAR
    x, y, x1, y1, PosAux : integer;
    Cur_Pren_Aux : boolean;
  BEGIN
    Dialogo := true;
    PosCur.Y := Prim_Dialogo;
    x1 := MainPort.PenLoc.X;
    y1 := MainPort.PenLoc.Y;
    moveto (Habit, Prim_Dialogo);
    drawtext (concat ('Nombre del archivo a ', Palabra, ': '));
    x := MainPort.PenLoc.X;
    y := MainPort.PenLoc.Y;
    PosAux := x DIV Habit;
    WITH Regleta DO
      BEGIN
        MarIzq := PosAux;
        Sang := PosAux;
        MarDer := PosAux + 25;
      END;
    LinT := concat (copy ('
                           1, pred (PosAux)), NombreArch);
    CursorC.Fis := succ (length (LinT));
    CursorC.Log := CursorC.Fis;
    PosCur.X := x + succ (length (Nombre_Arch)) * Habit;

    SetTextBG (TextBGBlanco);
    escribe ('                                        ');
    moveto (x, y);
    escribe (concat (' ', NombreArch));
    Cur_Pren_Aux := Cursor_Prendido;
    oprimio_esc := false;

    REPEAT
      main_loop;
      IF oprimio_esc THEN Dialogo := false;
    UNTIL NOT Dialogo;

    IF Cursor_Prendido THEN
      parpadea_cursor;
    LinT := copy (LinT, PosAux, length (LinT) - pred (PosAux));

    SetTextBG (TextBGNegro);
    Cursor_Prendido := Cur_Pren_Aux;
    moveto (x1, y1);
  END; {pide_nombre}



        {pide el nombre de un archivo a leer, si se sale, regresa false}
FUNCTION pide_nom_lee : boolean;
  CONST
    Tipo_Arch = '.EDDC';     {editor CDC}
    File_Not_Found = 10;
```

```
    No_Error = 0;
  VAR
    x, y, PosAux, respuesta : integer;
    c : char;
    p_n : boolean;
    s : string;
  BEGIN
    pide_nombre ('abrir');
    p_n := false;

    IF NOT oprimio_esc THEN
      BEGIN
        {$i-} {averiguar si ya existe ese archivo}
        reset (TempFile, concat (LinT, Tipo_Arch));
        respuesta := io_result;
        IF respuesta = No_Error THEN
          p_n := true
        ELSE IF respuesta = File_Not_Found THEN
          c := pide_dialogo (concat ('no se encontro ', LinT,
                                     Oprima <Return> '), [CR[1]])
        ELSE
          BEGIN
            intstr (respuesta, s);
            c := pide_dialogo (concat ('Error de Entrada/Salida numero  ', s,
                                     oprima <Return> '), [CR[1]]);
          END;
        close (TempFile);
        {$i+}
      END;

    pide_nom_lee := p_n;
  END; {pide_nom_lee}



        {pide el nombre de un archivo a salvar, si se sale, regresa false}
FUNCTION pide_nom_esc : boolean;
  CONST
    Tipo_Arch = '.ECDC';      {editor CDC}
    File_Not_Found = 10;
    No_Error = 0;
  VAR
    x, y, PosAux, respuesta : integer;
    c : char;
    p_n : boolean;
    s : string;
  BEGIN
    pide_nombre ('guardar');
    p_n := false;

    IF NOT oprimio_esc THEN
      BEGIN
        {$i-} {averiguar si ya existe ese archivo}
        reset (TempFile, concat (LinT, Tipo_Arch));
        respuesta := io_result;
        IF respuesta = File_Not_Found THEN
          p_n := true
        ELSE IF respuesta = No_Error THEN
          BEGIN
            c := pide_dialogo (concat (LinT, ' ya exite, ¿lo remplazo? (S/N) '),
```

```
                                    ['S', 's', 'N', 'n', CR[1]]);
            IF c IN ['S', 's'] THEN
              p_n := true;
          END
        ELSE
          BEGIN
            intstr (respuesta, s);
            c := pide_dialogo (concat ('Error de Entrada/Salida numero ', s,
                                       oprima <Return> '), [CR[1]]);
          END;
        close (TempFile);
        {$i+}
      END;

    IF p_n THEN
      Nombre_Arch := LinT;
    pide_nom_esc := p_n;
  END; {pide_nom_esc}



PROCEDURE leer_texto;
  CONST
    Tipo_Arch = '.ECDC';     {editor CDC}
    Offset = 2000;


  PROCEDURE lee;
    CONST
      CR_disco = 10;
      CR_texto = 13;
      Fin_Arch = 255;
      TamLin = 80;
    VAR
      i, j, Ap_Car : integer;
      Fin_Lectura, Segunda : boolean;
    BEGIN
      i := 1;
      j := 1;
      {$r-}
      Texto[1][0] := chr (TamLin);
      {$r+}
      Ap_Car := 1 + Offset;
      Fin_Lectura := false;
      Segunda := false;

      REPEAT
        IF BufFuentes[Ap_Car] = Fin_Arch THEN
          IF Segunda THEN
            Fin_Lectura := true
          ELSE
            Segunda := true
        ELSE
          BEGIN
            Segunda := false;
            CASE BufFuentes[Ap_Car] OF
              CR_disco:
                BEGIN
```

```
                Texto[i][j] := chr (CR_texto);
                j := succ (j);
              END;
            CR_texto:
              BEGIN
                {$r-}
                Texto[i][0] := chr (pred (j));
                {$r+}
                IF tipo (Texto[i]) = 0 THEN
                  ConT_Lineas.Log := succ (Cont_Lineas.Log)
                ELSE
                  i := pred (i);
                j := 1;
                i := succ (i);
                IF i > NumLin THEN
                  BEGIN
                    IF NOT swapeos (1, NumLin, 0, guarda_abajo) THEN
                      salir_error ('-----lee, error al guard bloq aba-----');
                    Cont_Lineas.Fis := Cont_Lineas.Fis + NumLin;
                    i := 1;
                  END;
                {$r-}
                Texto[i][0] := chr (TamLin);
                {$r+}
              END;
            OTHERWISE
              BEGIN
                Texto[i][j] := chr (BufFuentes[Ap_Car]);
                j := succ (j);
              END;
          END; {CASE}
      END;
    IF NOT Fin_Lectura THEN
      BEGIN
        Ap_Car := succ (Ap_Car);
        IF Ap_Car > Tam_Bloque + Offset THEN
          BEGIN
            IF blockread (TempFile, BufFuentes[succ (Offset)], 1) <> 1 THEN
              salir_error ('----lee, error al leer bloque-----');
            Ap_Car := 1 + Offset;
          END;
      END;
  UNTIL Fin_Lectura;

  IF i > 1 THEN
    BEGIN
      IF NOT swapeos (i, pred (i), 0, guarda_abajo) THEN
        salir_error ('-----lee, error al guard Line aba-----');
      Cont_Lineas.Fis := Cont_Lineas.Fis + pred (i);
    END;
END; {lee}



BEGIN
  reset (TempFile, concat (LinT, Tipo_Arch));

  IF blockread (TempFile, BufFuentes[succ (Offset)], 1) <> 1 THEN
    salir_error ('----leer_texto, error al leer bloque----');
```

```
  lee;

  close (TempFile, lock);
  traefuente (Fuente_Actual);
END; {leer_texto}



PROCEDURE imprim_texto;
  VAR
    ConT, Cont_Bl, Cont_Li, i, Ap_Car : integer;
    Imp : INTERACTIVE;

  PROCEDURE impri (Lineas : integer);
    VAR i, j : integer;
    BEGIN
      FOR i := 1 TO Lineas DO
        BEGIN
          FOR j := 1 TO length (Texto[i]) DO
            CASE ord (Texto[i][j]) OF
              Return, M_Cursor :;  {nada}
              Sangria, Esp_Suave : write (Imp, ' ');
              Guion_Suave :  write (Imp, '-');
              a_acento : write (Imp, 'a', chr(8), chr (96));
              e_acento : write (Imp, 'e', chr(8), chr (96));
              i_acento : write (Imp, 'i', chr(8), chr (96));
              o_acento : write (Imp, 'o', chr(8), chr (96));
              u_acento : write (Imp, 'u', chr(8), chr (96));
              OTHERWISE
                write (Imp, Texto[i][j]);
            END; {CASE}
          writeln (Imp);
        END;
    END; {impri}



  BEGIN
    rewrite (Imp, 'printer:');
    write (Imp, chr(27), chr(68), chr(7), chr(0));   {espanol}
    Ap_Car := 1;

    Cont := Cont_Lineas.Fis;
    Cont_Bl := Cont DIV NumLin;
    Cont_Li := ConT MOD NumLin;
    FOR i := 1 TO Cont_Bl DO
      BEGIN
        IF NOT svapeos (1, NumLin, 0, trae_de_arriba) THEN
          salir_error ('------imprim_texto, error al traer bloque------');
        impri (NumLin);
        IF NOT svapeos (1, NumLin, 0, guarda_abajo) THEN
          salir_error ('------imprim_texto, error al guardar bloque------');
      END;
    IF Cont_Li > 0 THEN
      BEGIN
        IF NOT svapeos (1, Cont_Li, 0, trae_de_arriba) THEN
          salir_error ('------imprim_texto, error al traer lineas------');
        impri (Cont_Li);
        IF NOT svapeos (1, Cont_Li, 0, guarda_abajo) THEN
```

```
        salir_error ('-----imprim_textp, error al guardar lineas-----');
    END;

  write (Imp, chr(27), chr(90), chr(7), chr(0));   {ingles}
  close (Imp);
END; {imprim_texto}


PROCEDURE salvar_texto;
  CONST
    Tipo_Arch = '.ECDC';     {editor CDC}
  VAR
    ConT, Cont_Bl, Cont_Li, i, Ap_Car : integer;

  PROCEDURE salva (Lineas : integer);
    CONST
      CR_disco = 10;
      CR_texto = 13;
    VAR i, j : integer;
    BEGIN
      FOR i := 1 TO Lineas DO
        BEGIN
          FOR j := 1 TO length (Texto[i]) DO
            BEGIN
              IF Texto[i][j] = chr (CR_texto) THEN
                BufFuentes[Ap_Car] := CR_disco
              ELSE
                BufFuentes[Ap_Car] := ord (Texto[i][j]);
              Ap_Car := succ (Ap_Car);
              IF Ap_Car > Tam_Bloque THEN
                BEGIN
                  IF blockwrite (TempFile, BufFuentes, 1) <> 1 THEN
                    salir_error ('-----salva, error al escribir bloque-----');
                  Ap_Car := 1;
                END;
            END;
          BufFuentes[Ap_Car] := CR_texto;
          Ap_Car := succ (Ap_Car);
          IF Ap_Car > Tam_Bloque THEN
            BEGIN
              IF blockwrite (TempFile, BufFuentes, 1) <> 1 THEN
                salir_error ('-----salva, error al escribir bloque-----');
              Ap_Car := 1;
            END;
        END;
    END; {salva}


  BEGIN
    rewrite (TempFile, concat (Nombre_Arch, Tipo_Arch));
    Ap_Car := 1;

    Cont := Cont_Lineas.Fis;
    Cont_Bl := Cont DIV NumLin;
    Cont_Li := ConT MOD NumLin;
    FOR i := 1 TO Cont_Bl DO
      BEGIN
```

```
      IF NOT swapeos (1, NumLin, 0, trae_de_arriba) THEN
        salir_error ('-----salvar_texto, error al traer bloque-----');
      salva (NumLin);
      IF NOT swapeos (1, NumLin, 0, guarda_abajo) THEN
        salir_error ('-----salvar_texto, error al guardar bloque-----');
    END;
  IF Cont_Li > 0 THEN
    BEGIN
      IF NOT swapeos (1, Cont_Li, 0, trae_de_arriba) THEN
        salir_error ('-----salvar_texto, error al traer lineas-----');
      salva (Cont_Li);
      IF NOT swapeos (1, Cont_Li, 0, guarda_abajo) THEN
        salir_error ('-----salvar_textp, error al guardar lineas-----');
    END;

  Texto[1] := '??';
  Texto[1][1] := chr (255);
  Texto[1][2] := chr (255);
  salva (1);
  IF Ap_Car > 1 THEN
    IF blockwrite (TempFile, BufFuentes, 1) <> 1 THEN
      salir_error ('----salvar_texto, error al escribir bloque-----');
  close (TempFile, lock);
  traefuente (Fuente_Actual);
END; {salvar_texto}


PROCEDURE ir_posicion (del_final : boolean);
  VAR
    Cont, Cont_Bl, Cont_Li, i : integer;
    t, g : Casos;

  BEGIN
    IF del_final THEN
      BEGIN
        Cont := Cont_Lineas.Fis - pred (Primera.Fis);
        t := trae_de_abajo;
        g := guarda_arriba;
      END
    ELSE
      BEGIN
        Cont := pred (Primera.Fis);
        t := trae_de_arriba;
        g := guarda_abajo;
      END;

    Cont_Bl := Cont DIV NumLin;
    Cont_Li := ConT MOD NumLin;
    FOR i := 1 TO Cont_Bl DO
      BEGIN
        IF NOT swapeos (1, NumLin, 0, t) THEN
          salir_error ('-----ir_posicion, error al traer bloque-----');
        IF NOT swapeos (1, NumLin, 0, g) THEN
          salir_error ('-----ir_posicion, error al guardar bloque-----');
      END;
    IF Cont_Li > 0 THEN
      BEGIN
        IF NOT swapeos (1, Cont_Li, 0, t) THEN
          salir_error ('-----ir_posicion, error al traer lineas-----');
```

```
        IF NOT swapeos (1, Cont_Li, 0, g) THEN
          salir_error ('-----ir_posicion, error al guardar lineas-----');
      END;
  END; {ir_posicion}


PROCEDURE limpia_dialogo;
  CONST
    Top = 13; Bottom = 31; Izq = 0; Der = 559;
  VAR
    r : rect;
  BEGIN
    haz_rect (r, Izq, Top, Der, Bottom);
    setpattern (Negro);
    paintrect (r);
    setpattern (Blanco);
  END; {limpia_dialogo}



PROCEDURE regre_cursor (pinto : boolean);
  VAR Aux, Cont, Cont_Bl, Cont_Li, i : integer;
  BEGIN
    Cont := Cont_Lineas.Fis - Con_Lin_Fis;
    Cont_Bl := Cont DIV NumLin;
    Cont_Li := Cont MOD NumLin;
    FOR i := 1 TO Cont_Bl DO
      BEGIN
        IF NOT swapeos (1, NumLin, 0, trae_de_abajo) THEN
          salir_error ('-----regre_cursor, error al traer bloque-----');
        IF NOT swapeos (1, NumLin, 0, guarda_arriba) THEN
          salir_error ('-----regre_cursor, error al guardar bloque-----');
      END;
    IF Cont_Li > 0 THEN
      BEGIN
        IF NOT swapeos (1, Cont_Li, 0, trae_de_abajo) THEN
          salir_error ('-----regre_cursor, error al traer lineas-----');
        IF NOT swapeos (1, Cont_Li, 0, guarda_arriba) THEN
          salir_error ('-----regre_cursor, error al guardar lineas-----');
      END;
        {recupera las lines que estaban en mem. principal}
    IF NOT swapeos (Primera_Linea^.En_Tex, 1, 0, trae_de_arriba) THEN
      salir_error ('--- - regresar_estado, la primera linea ---- ');
    FOR Aux := succ (CursorL_estado.Fis) TO Ultima_estado.Fis DO
      BEGIN
        toma_linea (LinT, succ (Ultima.Fis));
        guarda_linea (LinT, succ (Ultima.Fis));
      END;
    FOR Aux := pred (CursorL_estado.Fis) DOWNTO Primera_estado.Fis DO
      BEGIN
        toma_linea (LinT, Aux);
        guarda_linea (LinT, Aux);
      END;

        {regresar el estado actual de la pantalla}
    Home := Home_estado;
    CursorL := CursorL_estado;
    CursorC := CursorC_estado;
```

```
    PosCur := PosCur_estado;
    Regleta := Reg_estado;
    limpia_dialogo;
    IF pinto THEN
      BEGIN
        toma_linea (LinT, CursorL.Fis);
        insert (Marca_Cursor, LinT, CursorC.Fis);
        guarda_linea (LinT, CursorL.Fis);
        busca_cursor;
        pinta_vent (Home.Linea, Y_Ini_Vent + TamReng);
      END;
    toma_linea (LinT, CursorL.Fis);
  END; {regre_cursor}


PROCEDURE regresar_estado;
  VAR Aux : integer;
  BEGIN
      {recupera las lines que estaban en mem. principal}
    IF NOT swapeos (Primera.Linea^.En_Tex, 1, 0, trae_de_arriba) THEN
      salir_error ('----- regresar_estado, la primera linea ----');
    FOR Aux := succ (Primera_estado.Fis) TO Ultima_estado.Fis DO
      BEGIN
        toma_linea (LinT, succ (Ultima.Fis));
        guarda_linea (LinT, succ (Ultima.Fis));
      END;

      {regresar el estado actual de la pantalla}
    Home := Home_estado;
    CursorL := CursorL_estado;
    CursorC := CursorC_estado;
    PosCur := PosCur_estado;
    Regleta := Reg_estado;
    toma_linea (LinT, CursorL.Fis);
    limpia_dialogo;
  END; {regresar_estado}


PROCEDURE doApple;
  TYPE
    Adr_Cheat = RECORD
      CASE boolean OF
        true : (int : integer);
        false : (ptr : ^integer);
      END;
  VAR
    x, y, mem : integer;
    cheat : Adr_Cheat;
  BEGIN
    x := MainPort.PenLoc.X;
    y := MainPort.PenLoc.Y;
    moveto (10, 170);
    drawtext ('  memoria ->');
    mark (cheat.ptr);
    drawint (cheat.int);
    drawtext ('                    ');
    ln;
    drawtext ('   memavail ->');
```

```
    drawint (memavail);
    drawtext (',    ');
    mem := memavail;
    drawint (mem);
    drawtext ('          ');
    doreadln;
    moveto (x, y);
  END; {doApple}


PROCEDURE doDoc;
  CONST
    Doc_Nuevo = 1;        {Elementos del menu Documento}
    Doc_Abrir = 2;
    Doc_Cerrar = 3;
    Doc_Guardar = 4;
    Doc_Guar_Como = 5;
    Doc_Imprimir = 6;
    Doc_Salir = 7;

  BEGIN { dodoc }
    CASE Menu_choice OF
      Doc_Abrir :
        BEGIN
          guar_est_cur;
          IF pide_nom_lee THEN
            BEGIN
              leer_texto;
              IF Nombre_Arch = Nomb_Inicial THEN
                BEGIN
                  DisableItem (Doc_Menu, Doc_Guardar, false);
                  Nombre_Arch := LinT;
                END;
              regre_cursor (true);
            END
          ELSE
            regre_cursor (false);
        END;
      Doc_Guardar :
        BEGIN
          guarda_estado;
          ir_principio;
          salvar_texto;
          ir_posicion (true);
          regresar_estado;
        END;
      Doc_Guar_Como :
        BEGIN
          guarda_estado;
          ir_principio;
          IF pide_nom_esc THEN
            BEGIN
              salvar_texto;
              ir_posicion (true);
              DisableItem (Doc_Menu, Doc_Guardar, false);
            END
          ELSE
            ir_posicion (false);
          regresar_estado;
```

```
      END;
    Doc_Imprimir :
      BEGIN
        guarda_estado;
        ir_principio;
        imprim_texto;
        ir_posicion (true);
        regresar_estado;
      END;
    Doc_Salir : Salir := true;     {SALIR}
    OTHERWISE;
   END; {CASE}
END; { dodoc }


PROCEDURE doEdit;
  BEGIN
  END;


PROCEDURE doLoc;
  BEGIN
  END; {doLoc}


PROCEDURE doTipos;
  VAR Ban : boolean;
  BEGIN
    Ban := MData^[Tipos_Menu].Items[Menu_Choice].item_is_checked;
    Ban := no (Ban);      {la funcion NOT no es confiable en ENSAMBLADOR}

    IF Menu_Choice > 1 THEN
      CheckItem (Tipos_Menu, Menu_Choice, Ban);
  END;


PROCEDURE doVent;
  VAR
    Ap : Ap_Linea;
    x, y : integer;
  BEGIN
    Ap := Primera.Linea;
    x := MainPort.PenLoc.x;
    y := MainPort.PenLoc.y;
    WHILE Ap <> nil DO
      BEGIN
        moveto (10, 160);drawint (Ap^.En_Tex);
        IF tipo (Texto[Ap^.En_Tex]) = 0 THEN
          escribe (Texto[Ap^.En_Tex])
        ELSE
          escribe ('Regleta');
        doreadln;
        Ap := Ap^.Sig;
      END;
    moveto (10, 160);
    escribe ('Home.Fis = ');
```

```
    drawint (Home.Fis);
    escribe ('                    ');
    doreadln;
    moveto (10, 160);
    escribe ('-------------- fin ------------------');
    esp;
    moveto (x, y);
  END; {doVent}


PROCEDURE doForma;
  BEGIN
  END;


PROCEDURE doMenu;
  BEGIN { doMenu }
    IF Menu_ID <> 0 THEN
      CASE Menu_id OF
        Apple_Menu: doApple;
        Doc_Menu: doDoc;
        Edit_Menu: doEdit;
        Loc_Menu: doLoc;
        Tipos_Menu : doTipos;
        Vent_Menu : doVent;
        Forma_Menu : doForma ;
      END
    ELSE exit(doMenu);
    hilitemenu(0);
  END; { doMenu }


PROCEDURE handlebutton;
  VAR
    ScreenX, ScreenY : integer;
BEGIN { handlebutton }
  WITH event DO
   BEGIN
    ScreenX := char1 + 256 * char2;
    ScreenY := char3 + 256 * char4;
    IF ScreenY < 10 THEN
     BEGIN
      MenuSelect(Menu_id, Menu_choice);
      domenu;
     END; { CASE inMenubar }
  END; { WITH }
END; { handlebutton }


{
* tecla
*  FUNCION: manejar la tecla que recibio el loop principal.
}
PROCEDURE tecla;
  BEGIN
    ObscureCursor;
    IF NOT Dialogo THEN
      MenuKey(Menu_id, Menu_choice, event);
    IF (Menu_id <> 0) AND (NOT Dialogo)  THEN doMenu
```

```
    ELSE maneja_teclas;
  END;


{
* parpadeo en BackGrown
*  FUNCION:  Realiza el parpadeo del cursor, usando esta tecnica se
*  pueden hacer otras funciones sin detener el programa, tales como:
*  impresion, pintado de la pantalla, calculo de palabras escritas,
*  correccion de ortografia, etc.
}
PROCEDURE BG_parpadeo;
  BEGIN
    Cont_No_Event := succ (Cont_No_Event);
    IF Cont_No_Event >= Cont_Event THEN
      parpadea_cursor;
  END;



PROCEDURE main_loop;
  BEGIN
    GetEvent(event);
    IF event.evt_kind in [buttondown,keydown] THEN
      BEGIN
        IF Cursor_Prendido THEN
          parpadea_cursor;
        CASE event.evt_kind OF
           button_down : IF NOT Dialogo THEN HandleButton;
           key_down : tecla;
         END; {CASE}
        parpadea_cursor;
      END
    ELSE IF event.evt_kind = No_Event THEN
      BG_parpadeo;
  END; {main_loop}



BEGIN                                   { program }
comienza;
Cont_Event := 0;
procesado := true;
acentuado := false;
Nombre_Arch := Nomb_Inicial;
DisableItem (Doc_Menu, Doc_Guardar, true);
Dialogo := false;
cheat.int := Pagina_2;
Release (cheat.ptr);                    {proteje el buffer de graficas}
pinta_pantalla;

REPEAT                          {loop principal}
  main_loop;
UNTIL Salir;

Version(VMajor,VMedium,VMinor,VStatus,VRelease);
writeln('Version ',VMajor,'.',VMedium,'.',VMinor,VStatus,VRelease,
                  MemAvail:  ',Memavail);
StopDeskTop;
END. { M }
```

```
LIBRARY MAP FOR tesis1.lib

                    Apple Computers Inc.

Segment #39:
System version = A2/1.3, code type is P-Code (least sig. 1st)
CARGAF    library unit (LINKED INTRINSIC)

  USES
    {$u tesis1.lib}  mgrkit.              {Mouse Graphics Kit, Menus only}
    pepo;                                 {acc. direc. de mem. aux., swapeos}

  CONST
    Tam_Bloque = 512;                     {tamano de un bloque de disco, Pascal}
    Max_Buff = 2560;                      {tam. del maximo buffer para
                                           contener una fuente}
    Banco_Fuentes = 7;

  TYPE

    Fuentes = (Negritas, Italicas, Gordas, Flacas, Habitual);

    Direcciones = RECORD
                    Adr : integer;
                    Tam : integer;
                  END;

    B_Fuente =  PACKED ARRAY [1..Max_Buff] OF Byte;

    CharBlock = PACKED ARRAY [1..Tam_Bloque] OF char;

  VAR
    TempFile : file;                      {para leer fuentes}
    PtrToFont : integer;                  {apuntador a la fuente del menu}
    BufFuentes : B_Fuente;                {buffer para las otras fuentes}
    Fuente_Actual : Fuentes;              {la fuente actual en mem. prin.}


    {trae a memoria principal la fuente "f"}
  PROCEDURE traefuente (f : fuentes);

-------------------------------------------------------------------------------

Segment #40:
System version = A2/1.3, code type is P-Code (least sig. 1st)
CARGAF    data segment

-------------------------------------------------------------------------------

Segment #37:
System version = A2/1.3, code type is 6502
```

```
PEPO      library unit (LINKED INTRINSIC)


USES
   {$u menus.lib} mgrkit;

CONST

  minima_loc = 512;              {minima localidad de memoria }
  Tam_Fis_Lin = 92;              {tamano fisico de nuestras lineas en bytes}
  Max_Lin_Banco = 528;           {maximo numero de lineas por banco}
  Banco_hf_Ini = 6;              {limites del doble stack}
  Banco_lf_Ini = 1;
  NumLin = 30;                   {numero de lineas en mem. prin.}

TYPE

     {los tipos de manejos de memoria auxiliar}
  Movi = (Guarda, Recupera);

     {los tipos de swapeos}
  Casos = (trae_de_arriba, trae_de_abajo, guarda_arriba, guarda_abajo,
           retrocede trae_arriba, avanza trae_abajo);


   Varrec = PACKED ARRAY[0..0] OF Byte;       {para "peek" y "poke"}

   Trixi = RECORD                            {para "peek" y "poke"}
     CASE boolean OF
       false : (Address : integer);
       true : (Pointer : ^varrec);
     END;

   Tipo_Linea = string[90];                  {unidad fundamental de texto}
   Matriz = ARRAY[1..NumLin] OF Tipo_Linea;  {el texto en mem. prin.}


VAR
  fin, Alerta_Mem : boolean;   {indican si se termino el espacio para texto}
  Texto : Matriz;              {area de trabajo en mem. prin.}


     {lee la localidad de memoria "Addr"}
FUNCTION peek (Addr : integer) : Byte;

     {escribe "Val" en la localidad de memoria "Addr"}
PROCEDURE  poke (Addr : integer; Val : Byte);

     {realiza un paso de datos entre memoria principal y
      auxiliar, los parametros son: descriptores del bloque
      a mover, con cual banco auxiliar se va a trabajar y el
      sentido de la transferencia}
PROCEDURE auxmove (IniBloque, FinBloque, Destino, Banco : integer;
                   Mov : Movi);

              {funcion general de manejo de memoria auxiliar}
FUNCTION swapeos (Ini, Lineas, Final : integer;
```

```
                    Caso : Casos              {la funcion a realizar}
                      ) : boolean;

      {trae lineas de algun banco}
PROCEDURE traebloc (Ini,              {inicio del bloque de lineas}
                    Lineas,           {# de lin. a traer}
                    Destino,          {lin. de tex. a donde se traera}
                    Banco : integer);

      {trae lineas enteras de Hi_File, si falla, regresa "false"}
FUNCTION traearri (Ini,                {la linea inicial a llenar}
                   Lineas : integer)   {el numero de lineas a traer}
                     : boolean;

      {como "traearri" pero de Low_File}
FUNCTION traeaba (Ini,                 {la linea inicial a llenar}
                  Lineas : integer)    {el numero de lineas a traer}
                    : boolean;

      {guarda lineas en algun banco}
PROCEDURE guardabloc (Ini,             {inicio del bloque de lineas}
                      Lineas,          {# de lin. a guardar}
                      Destino,         {loc. de mem. en donde se guardara}
                      Banco : integer);

      {guarda lineas enteras en Hi_File, regresa "false" si se agoto la mem.}
FUNCTION guardarri (Ini,               {la linea inicial del texto a guardar}
                    Lineas : integer) {el numero de lineas a guardar}
                      : boolean;

      {como "guardarri" pero en Low_File}
FUNCTION guardaba (Ini,                {la linea inicial del texto a guardar}
                   Lineas : integer) {el numero de lineas a guardar}
                     : boolean;

      {recorre lineas de "Texto" a Hi_File y de Low_File a "Texto"}
FUNCTION recarri (Ini,              {linea inicial a recorrer}
                  Lineas,           {# de lineas a recorrer}
                  Final             {ultima linea a recorrer}
                    : integer) : boolean;

      {recorre lineas de "Texto" a Low_File y de Hi_File a "Texto"}
FUNCTION recaba (Ini,               {linea inicial a recorrer}
                 Lineas,            {# de lineas a recorrer}
                 Final              {ultima linea a recorrer}
                   : integer) : boolean;
```

```
Segment #38:
System version = A2/1.3, code type is P-Code (least sig. 1st)
PEPO      data segment
```

```
Segment #20:
System version = A2/1.1, code type is 6502
MORKIT     library unit (LINKED INTRINSIC)


const
     max_menus = 10;
     max_title_str = 20;
     max_item_str = 30;
     max_num_items = 10;

     no_event = 0;
     button_down = 1;
     button_up = 2;
     key_down = 3;
     drag = 4;
     apple_key = 5;
     update_event = 6;

type
     byte = 0..255;

     bits = packed array [0..7] of boolean;

     pmode = (penCopy, penOr, penXor, penBic,
              notpenCopy, notpenOr, notpenXor, notpenBic);

     point = record
                x: integer;
                y: integer;
             end;

     rect = record
               case integer of
                  0: (left: integer;
                      top: integer;
                      right: integer;
                      bottom: integer);
                  1: (topleft: point;
                      botright: point);
               end;

     mapinfo = packed record
                  viewloc: point;
                  mapbits: integer;
                  mapwidth: byte;
                  reserved: byte;
                  maprect: rect;
               end;

     maskinfo = packed record
                   andmask : byte;
                   ormask : byte;
                end;

     pattern = packed array [1..8] of byte;

     grafport= packed record
```

```
            portmap: mapinfo;
            penpattern: pattern;
            colormasks : maskinfo;
            penloc: point;
            penwidth: byte;
            penheight: byte;
            penmode: pmode;
            reserved: 0..31;
            textback: byte;
            textfont: integer;
          end;

type_event = packed record
              evt_kind : byte;
              char1 : byte;
              char2 : byte;
              char3 : byte;
              char4 : byte;
             end;

title_str = string[max_title_str];
item_str = string[max_item_str];

menu_item = packed record

            open_apple : boolean;              {bit 0}
            solid_apple : boolean;
            item_has_mark : boolean;
            reserve2 : boolean;
            reserve3 : boolean;
            item_is_checked : boolean;
            item_is_filler : boolean;
            disable_flag : boolean;            {bit 7}

            mark_char : byte;

            char1 : byte;
            char2 : byte;

            item_str_ptr : ^item_str;

             end;

menu_data = packed record
             num_items : byte;
             reserve1 : byte;
             reserve2 : packed array [1..4] of byte;
             items : packed array [1..max_num_items] of menu_item;
             end;

menu_title = packed record
               menu_id : byte;
               disabled : byte;
               title_ptr : ^title_str;
               m_data_ptr : ^menu_data;
               reserved : packed array [1..6] of byte;
             end;

menu_bar = packed record
            num_menus : byte;
```

```
                reserved : byte;
                menus : array [1..max_menus] of menu_title;
                end;


    cursor_image = packed array [0..23] of byte;

    cursor_rec = packed record
                    cursor_bits: cursor_image;
                    cursor_mask: cursor_image;
                    hotspotx: byte;
                    hotspoty: byte;
                 end;

var
    pointer,
    TKError : integer;

function PointerTo(var Variable)
        : integer;

procedure InitGraf;
procedure SetSwitches(flags: bits);
procedure InitPort(var a_grafport: grafport);
procedure SetPort(var a_grafport: grafport);
procedure GetPort(var portptr: integer);
procedure SetPortBits(a_mapinfo: mapinfo);
procedure SetPenMode(a_penmode: pmode);
procedure SetPattern(a_pattern: pattern);
procedure SetColorMask(a_maskinfo : maskinfo);
procedure SetPenSize(width, height: byte);
procedure SetFont(FontPtr: integer);
procedure SetTextBG(TextBG: byte);
procedure PaintRect(a_rect: rect);
procedure FrameRect(a_rect: rect);
function InRect(a_rect: rect)
        : boolean;
procedure PaintBits(a_mapinfo: mapinfo);
procedure Move(dx, dy: integer);
procedure MoveTo(x, y: integer);
procedure Line(dx, dy: integer);
procedure LineTo(x, y: integer);
procedure PaintPoly(polyptr: integer);
procedure FramePoly(polyptr: integer);
function InPoly(polyptr: integer)
        : boolean;
function TextWidth(a_string: string)
        : integer;
procedure DrawText(a_string: string);
procedure Version (var VMajor,VMedium,VMinor : integer;
                   var VStatus : char;
                   var VRelease : integer);

procedure StartDeskTop ( mach_id : integer;
                         sub_id: integer;
                         var slot_num : integer;
                         var use_interrupts : boolean;
                         FontPtr: integer;
                         savearea: integer;
                         savesize: integer);
```

```
procedure StopDeskTop;
procedure GetIntAdr    (var IntAdr : integer);

procedure SetCursor ( var a_cursor: cursor_rec );
procedure ShowCursor;
procedure HideCursor;
procedure ObscureCursor;
procedure GetCursorAdr ( var CursorPtr : integer);


procedure CheckEvents;
procedure GetEvent ( var event : type_event );
procedure PeekEvent ( var event : type_event );
procedure FlushEvents;
procedure PostEvent( event: type_event );
procedure SetKeyEvent ( chk_keyboard : boolean );


procedure InitMenu ( solid_char, open_char, control_char, check_char,
                     inactive: byte);
procedure SetMenu ( var a_menu_bar : menu_bar );
procedure MenuSelect ( var menu_id, item_num : integer );
procedure MenuKey ( var menu_id, item_num : integer;
                    var key_event : type_event );
procedure HiliteMenu ( menu_id : integer );
procedure DisableMenu ( menu_id : integer; disable : boolean );
procedure DisableItem ( menu_id, item_num : integer; disable : boolean );
procedure CheckItem ( menu_id, item_num : integer; check : boolean );
procedure SetMark ( menu_id, item_num: integer; mark_on: boolean;
                    mark_char: char);
```

```
procedure ScaleMouse(x_exponent, y_exponent: integer);

procedure FakeMouse;

procedure ToolkitAddress(var address:integer);
```

```
Segment #21:
System version = A2.1.1, code type is P-Code (least sig. 1st)
MGRKIT    data segment
```

```
Segment #45:
System version = A2/1.3, code type is P-Code (least sig. 1st)
DECLARA   library unit (LINKED INTRINSIC)


  USES
    {$u menus.lib}  mgrkit,
    {$u peelpole.lib} pepo;

  CONST
    Mis_Menus = 7;                {  constantes de los menus  }

    Apple_Menu = 1;    Doc_Menu = 2;    Edit_Menu = 3;    Loc_Menu = 4;
    Tipos_Menu = 5;    Vent_Menu = 6;   Forma_Menu = 7;

                              {  numero de elementos por menu  }
    Apple_Elems = 4;    Doc_Elems = 7;    Edit_Elems = 6;    Loc_Elems = 3;
    Tipos_Elems = 10;   Vent_Elems = 7;   Forma_Elems = 6;

                {marcas de tipos de letras}
    Inic_Negritas = 129;       Fin_Negritas = 130;       Inic_Altas = 131;
    Fin_Altas = 132;           Inic_Bajas = 133;         Fin_Bajas = 134;
    Inic_Subrayado = 135;      Fin_Subrayado = 136;      Inic_Enfatizadas = 137;
    Fin_Enfatizadas = 138;     Inic_Italicas = 139;      Fin_Italicas = 140;
    Inic_Gordas = 141;         Fin_Gordas = 142;         Inic_Flacas = 143;
    Fin_Flacas = 144;

                {caracteres de movimiento}
    Flech_Adelante = 21;       Flech_Atras = 8;
    Flech_Arriba = 11;         Flech_Abajo = 10;

                {algunos caracteres especiales de edicion}
    Margen = 1;        Margen_y_Sangria = 2;   Sangria = 3;       Tab_Decimal = 4;
    Esp1 = 5;          Esp2 = 6;               Esp3 = 7;          Tab = 9;
    Return = 13;       No_Alineado = 14;       Esp_Suave = 19;    Guion_Suave = 20;
    Esp_Duro = 24;     Del = 127;              M_Borra = 324;     M_Cursor = 325;
    a_acento = 23;     e_acento = 24;          i_acento = 25;     o_acento = 26;
    u_acento = 27;     u_dieresis = 28;        esc = 27;          acento = 39;


    TipoRegHor = 250;                {tipos de reglstas}
```

```
  TipoRegVar = 251;
  TipoRegCab = 252;

  TamRegHor = 18;                 {tamanos de regletas}
  TamRegVar = 6;
  TamRegCab = 20;

  Tam2Lin = 182;                  {tamano de las lineas dobles}

  MaxTab = 10;                    {numero maximo de tabuladores}
  MaxTabMasl = 11;

  MaxMarcas = 40;

  Habit = 7;                      {anchos de las fuentes}
  Flac = 4;
  Gord = 14;

  Y_Ini_Vent = 39;      {16 del menu + 16 de 2 renglones + 8 para comenzar
                         en la parte inferior del renglon}
  TamReng = 8;          {en pixels}
  Ancho_Pant = 559;
  Alto_Pant = 191;
  Prim_Dialogo = 23;
  Seg_Dialogo = 31;
  Fin_Vent = 183;       {Alto_Pant - TamReng}
  Tam_Vent = 18;        {renglones en los que puede ir texto}

  No_Manzana = 0;       {no se apreto ninguno ninguna manzana}

  TextBGNegro = 0;
  TextBGBlanco = 127;             {7F}


TYPE
  Menu_Buffer = PACKED ARRAY [0..1999] OF Byte;

  { cada arreglo es declarado solo para el espacio que
    necesita, para ahorrar memoria                      }

  TAppleItem = ARRAY [1..Apple_Elems] OF string[17];
  TDocItem   = ARRAY [1..Doc_Elems]   OF string[15];
  TEditItem  = ARRAY [1..Edit_Elems]  OF string[8];
  TLocItem   = ARRAY [1..Loc_Elems]   OF string[9];
  TTiposItem = ARRAY [1..Tipos_Elems] OF string[8];
  TVentItem  = ARRAY [1..Vent_Elems]  OF string[7];
  TFormaItem = ARRAY [1..Forma_Elems] OF string[19];

  Datos_Menu = ARRAY [1..Mis_Menus] OF Menu_data;
  Titulos = ARRAY [1..Mis_Menus] OF title_str;

  {*TamLin = 91;                  {tamano de las lineas}
  Tipo_Linea = string[TamLin];          {tipo fundamental de texto}*)
  Tipo_2_Lineas = string[Tam2Lin];    {para que quepan 2 lineas.}
                                      {esto es necesario al editar}

            {regletas}
  Reg = PACKED RECORD
    Tam : Byte;                                 {tamano}
    Tipo : Byte;                                {tipo de regleta}
```

```
  Tabs : PACKED ARRAY [1..MaxTab] OF Byte;  {marcas de tabulador}
  MarIzq : Byte;
  MarDer : Byte;
  Sang : Byte;                {sangria, booleano, 1 = true, 0 = false}
  NuevPag : Byte;             {booleano}
  Alinea : Byte;              {booleano}
  Silabeo : Byte;             {booleano}
  Espaciamiento : Byte;       {1, 2 o 3}
  LinCab : Byte;
  LinPies : Byte;
 END;    { Reg }

 Desc_Pos = RECORD
  Fis,                  {posicion fisica dentro de una linea}
  Log                   {posicion logica dentro de una linea}
    : integer;
 END; {Desc_Pos}

 Ap_Linea = ^Desc_Linea;

 Desc_Linea = RECORD
  Sig : Ap_Linea;           {apuntador a la siguiente linea}
  Ant : Ap_Linea;           {apuntador a la linea anterior}
  En_Tex : integer;          {apuntador al texto}
 END;

 Ap_1_Linea = RECORD
  linea : Ap_Linea;         {apunta a una linea}
  Fis : integer;             {contador de lineas fisico}
  Log : integer;             {contador de lineas logico}
 END;

 Rango_Marcas = 1..MaxMarcas;

 TAp_Marcas = RECORD                    {tipo apuntador a marcas}
  Marc : Tipo_Linea;                                {Guarda las marcas}
  Apunt : PACKED ARRAY [Rango_Marcas] OF Byte;      {Apuntan a las marca}
 END; {TAp_Marcas}


VAR
 Salir,                             {para terminar el loop principal}
 Use_Interrupts,                    {por si se usan interupciones}
 Cursor_Prendido,                   {para saber si hay que apagarlo}
 acentuado,                         {si el ultimo caracter fue un acento}
 editando            {si esta prendida, LinT debe ser guardado en Texto}
   : boolean;

 MainPort : grafport;

 Menu_choice,
 Menu_id,
 MouseSlot,          {obtenido por StartDesk.Top}
 Cont_No_Event,      {usado por el parpadeo}
 Ancho                 {el ancho de la fuente en uso}
   : integer;

 Event
   : type_event;
```

```
  a_menubar                    {la barra del menu}
    : ^Menu_bar;

  {DATOS DEL MENU}
  MData
    : ^Datos_Menu;

  {Menubar NOMBRES}
  MTitle
    : ^Titulos;

  No_usadas : Ap_Linea;              {para la lista de renglones no usados}
  Primera,                           {primera linea de texto usada en mem}
  Ultima,                            {ultima =}
  CursorL,                           {posicion del cursor}
  Cont_Lineas,                       {tamano actual de archivo}
  Home : Ap_1_Linea;                 {primera linea a desplegar}

  VMajor,VMedium,VMinor,VRelease : integer;
  VStatus : char;

  AppleItem : ^TAppleItem;
  DocItem   : ^TDocItem;
  EditItem  : ^TEditItem;
  LocItem   : ^TLocItem;
  TiposItem : ^TTiposItem;
  VentItem  : ^TVentItem;
  FormaItem : ^TFormaItem;

  Blanco,
  Negro
     : ^Pattern;

  Menu_Area : ^Menu_Buffer;

  CursorC : Desc_Pos;          {columna del cursor}
      {apunta al caracter siguiente al punto de insercion}

  C_Sangria,             {caracter para la sangria}
  C_Tab,                 {caracter de tabulador}
{  Los tabuladores se
```

```
Segment #46:
System version = A2/1.3, code type is P-Code (least sig. 1st)
DECLARA   data segment
```

```
Segment #47:
System version = A2/1.3, code type is P-Code (least sig. 1st)
INICIA    library unit (LINKED INTRINSIC)


  USES
```

```
{$u tesis1.tib}  mgrkit, pepo, cargaf,
{$U declara.code} declara;



{la funcion NOT no es confiable en ENSAMBLADOR}

FUNCTION no (Ban : boolean) : boolean;


{convierte un entero en string, para desplegarlo}

PROCEDURE intstr (i : integer; VAR s : string);


{toma una linea de la lista de no usadas y la pone en Ap}

PROCEDURE usa_linea (VAR Ap : Ap_Linea);


{
*  pon_offset
*     agrega a todos los caracteres de una regleta en texto un offset (128)
*     para que no ocacionen un problema al escribir a disco caracteres 0
}

PROCEDURE pon_offset (VAR Lin : Tipo_Linea);



{
* quita_offset
*    quita a todos los caracteres de una regleta en texto un offset (128)
*    para poder usarla en  edicion.
}

PROCEDURE quita_offset (VAR Lin : Tipo_Linea);



{
* tipo
*  FUNCION : Regreza el tipo de linea del parametro pasado en Ap_Linea.
}

FUNCTION tipo (Ap_Linea : Tipo_2_Lineas) : Byte;
```

```
{
* marca_contraria
*  FUNCION : Regresa un caracter que es la marca opuesta a la marca pasada
*            como parametro, por ejemplo, si se pasa una marca Ini_Flacas,
*            se regresara una marca Fin_Flacas.
}

FUNCTION marca_contraria (Car : char) : char;



{
* agrega_sangria
*  FUNCION : Poner los espacios suaves a la izquierda de una linea dependiendo
*            de si tiene sangria o no.
}

PROCEDURE agrega_sangria (VAR Ap_Lin : Tipo_2_Lineas);



{
* inicializa la mayoria de las estructuras de datos, puede ser segmento o
* estar en una libreria
}

PROCEDURE comienza;
```

```
LIBRARY MAP FOR movimiento.code

Segment #50:
System version = A2/1.3, code type is P-Code (least sig. 1st)
MOVIMIEN  library unit (LINKED INTRINSIC)


  USES
    i$u tesis1.libi  mgrkit, pepo, cargaf, declara, inicia,
    i$u libsilabso.codei silabso,
    i$u edita.codei edita;


        {movimientos del cursor}
PROCEDURE maneja_movimientos;
```

```
  LIBRARY MAP FOR libsilabeo.code

Segment #33:
System version = A2/1.3, code type is P-Code (least sig. 1st)
SILABEO    library unit (LINKED INTRINSIC)

  CONST
   MaxMD = 50;
  TYPE
   con = SET OF char;
   T_2_L = string[182];
  VAR
   vocales, juego_r, juego_l, juego_h, letras : con;

  PROCEDURE parte (VAR palabra, palabra1 : T_2_L;
           md : integer);

  ------------------------------------------------------------------------

Segment #34:
System version = A2/1.3, code type is P-Code (least sig. 1st)
SILABEO    data segment

  ------------------------------------------------------------------------
```

```
LIBRARY MAP FOR edita.code

Segment #48:
System version = A2/1.3, code type is P-Code (least sig. 1st)
EDITA      library unit (LINKED INTRINSIC)

  USES
    {$u tesis1.lib} mgrkit, pepo, cargaf, declara, inicia,
    {$u libsilabeo.code} silabeo;

  VAR
    Cont_Event : integer;
    procesado, Dialogo : boolean;
    Primera_estado, Ultima_estado, Home_estado, CursorL_estado : Ap_1_Linea;
    CursorC_estado : Desc_Pos;
    PosCur_estado : Point;
    Reg_estado : Reg;
    Nombre_Arch : string;


      {regresa el ultimo caracter de un string}
FUNCTION ultimo (s : Tipo_2_Lineas) : char;

        {quita los espacios suaves que pueda tener por formateo}
PROCEDURE quita_esp_suaves (VAR Lin : Tipo_2_Lineas);

        {regresa el tamano logico de una linea}
FUNCTION longitud (s : Tipo_2_Lineas) : integer;

                {despliega un entero}
PROCEDURE drawint (x : integer);

{
* ln
*  FUNCION :  Mover el lapiz de dibujo a la linea de abajo, como
*  al mandar un CR en la pantalla de texto, por eso se llama ln
*  (de writeLN).
}
PROCEDURE ln;

{
* esc_regleta
*  FUNCION :  Escribir una regleta como una linea en la pantalla
*  simulando una regleta de maquina de escribir.
*    La regleta se dibuja en la posicion Y en la que esta el lapiz,
*  por lo tanto hay que posicionar antes de llamar esta rutina.
}
PROCEDURE esc_regleta (r : Reg);

{
*  escribe
*   FUNCION :   Escribir una linea de texto en la pantalla, cuidando de no
*               pasar el marco derecho.
}
PROCEDURE escribe (L : Tipo_2_Lineas);
```

```
{
*  escribe_ln
*    FUNCION :    Escribir una linea y pasar al siguiente renglon,
*                 llama a escribe.
}
PROCEDURE escribe_ln (L : Tipo_2_Lineas);

{
* pinta_pantalla
*  FUNCION : pintar la pantalla inicial, el marco, la regleta estandar.
}
PROCEDURE pinta_pantalla;

{
* parpadea_cursor
*  FUNCION :  Prender y apagar el cursor, fija el valor de Cont_Event
*             dependiendo de si el cursor se apaga o se prende, porque la
*             duracion debe ser distinta, 2/3 predida y 1/3 apagada.
*  VARIABLES GLOBALES MODIFICADAS: Cursor_Prendido, Cont_Event.
}
PROCEDURE parpadea_cursor;

      {separa una linea en 2 lineas}
PROCEDURE separa_lineas (VAR Lin, Seg : Tipo_2_Lineas;
                         Carac : integer);      {apunta al fin de la pri. lin.}

{parte una linea en el margen derecho}
PROCEDURE parte_G (VAR Lin, Seg : Tipo_2_Lineas;
                   MD : integer);

        {imprime un mensaje de error y luego sale}
PROCEDURE salir_error (s : string);

    {mueve un descriptor de lineas en base a el texto}
PROCEDURE en_tex_avanza (VAR Ap_l : Ap_l_Linea;
                             Direccion : integer);

   {quita una linea de la lista de lineas en memoria principal}
PROCEDURE dispo (Ap : Ap_Linea);

 {guarda uno de los extremos del texto en el stack correspondiente}
PROCEDURE guarda_stack (Ap : Ap_Linea);

      {mete una linea en "Texto"}
PROCEDURE guarda_linea (L : Tipo_2_Lineas; PosFis : integer);

   {toma una linea de la Posicion fisica especificada y la deja en "L"}
PROCEDURE toma_linea (VAR L : Tipo_2_Lineas; PosFis : integer);

   {busca la marca del cursor, corrije los apuntadores a la pantalla}
PROCEDURE busca_cursor;

  {refresca la pantalla}
PROCEDURE pinta_vent (Ap : Ap_Linea; Ap_Y : integer);

        {formatea el texto}
PROCEDURE forma_anterior (VAR L : Tipo_2_Lineas;
                              PosLin, PosCur : integer);

 {refresca la linea de ttrabajo en la pantalla}
```

```
PROCEDURE refresca_LinT;

{
* FUNCION: Pone un acento temporal en LinT y prende la bandera "acentuado".
* VARIABLES GLOBALES MODIFICADAS: acentuado.
}
PROCEDURE inserta_acento;

      ino

    ------------------------------------------------------------------------

Segment #49:
System version = A2/1.3. code type is P-Code (least sig. 1st)
EDITA     data segment

    ------------------------------------------------------------------------
```

```
PAGE -  0
Current memory available:   9294
0000:                       ;      {LLAMA AUTOMOVE ( 9C311 EN ROM)  {Miercoles 30 de Julio del 86}
0000:                       ;                                       {Lunes 4 de Agosto del 86}
0000:                       :       PROCEDURE automove (IniBloque, FinBloque, Destino, Banco : integer;
0000:                       ;                           Mov : Movi);
2 blocks for procedure code  8787 words left
```

```
0000!                                   .PROC   auxmove, 5
Current memory available:   8769
0000!                   ;
0000!                   ;               DEFINICION DE PARAMETROS
0000!                   ;
0000! 0000              Mov             .EQU    0               ; EL TIPO DE MOVIMIENTO,
0000!                                                           ; Guarda = 0, Recupera = 1
0000!                   ;
0000!                   ;               DEFINICION DE VARIABLES
0000!                   ;
0000! 0002              Return          .EQU    Mov+2           ; LA DIRECCION DE RETORNO
0000! 0004              A1L_aux         .EQU    Return+2        ; ALMACENAMIENTOS TEMPORALES
0000! 0006              A2L_aux         .EQU    A1L_aux+2
0000! 0008              A4L_aux         .EQU    A2L_aux+2
0000! 000A              STORE80_aux     .EQU    A4L_aux+2       ; PARA GUARDAR EL ESTADO DE 80STORE
0000!                   ;
0000!                   ;               DEFINICION DE CONSTANTES
0000!                   ;
0000! C311              AUXMOVE_ROM     .EQU    0C311           ; ENTRY POINT DE LA RUTINA EN ROM
0000! 003C              A1L             .EQU    3C              ; VARIABLES USADAS POR AUXMOVE_ROM
0000! 003E              A2L             .EQU    3E
0000! 0042              A4L             .EQU    42
0000! C075              SwBank          .EQU    0C075           ; PUERTO DE LA TARJETA SuperRam 512
0000! C001              STORE80_ON      .EQU    0C001           ; PRENDE 80STORE
0000! C000              STORE80_OFF     .EQU    0C000           ; APAGA 80STORE
0000! C018              RD80STORE       .EQU    0C018           ; LEE EL ESTADO DE 80STORE
0000!                   ;
0000!                   ;               DEFINICION DE MACROS
0000!                   ;
0000!                                   .NOMACROLIST            ; NO EXPANSION DE MACROS
0000!
0000!                                   .MACRO  pop
0000!                                   PLA                     ; TOMA DEL STACK EL LSB
0000!                                   STA     %1
0000!                                   PLA                     ; Y EL MSB
0000!                                   STA     %1+1
0000!                                   .ENDM
0000!
0000!                                   .MACRO  push
0000!                                   LDA     %1+1            ; METE EN EL STACK EL MSB
0000!                                   PHA
0000!                                   LDA     %1              ; Y EL LSB
0000!                                   PHA
0000!                                   .ENDM
0000!
0000!                                   .MACRO  mueve
0000!                                   LDA     %1              ; MUEVE LSB DEL PARAMETRO 1
0000!                                   STA     %2              ; AL PARAMETRO 2
0000!                                   LDA     %1+1            ; Y EL MSB
0000!                                   STA     %2+1
0000!                                   .ENDM
0000!
0000!                   ;       BEGIN [auxmove]
0000!
0000!                   ;       GUARDA LA DIRECCION DE RETORNO
```

```
0000:                                   pop     Return
0006:
0006:                           ;       GUARDA EL PARAMETRO Mov
0006:                                   pop     Mov
000C:
000C:                           ;       HABILITA EL BANCO AUXILIAR "Banco"
000C: 68                                pla                     ; LSB DE Banco
000D: 8D 7500                           sta     SwBank
0010: 68                                pla                     ; DESECHAMOS MSB
0011:
0011:                           ;       GUARDAMOS LOS VALORES ACTUALES DE LAS VARTABLES DE AUXMOVE_ROM (A)
0011:                                   mueve   A1L,A1L_aux
0019:                                   mueve   A2L,A2L_aux
0021:                                   mueve   A4L,A4L_aux
0029:
0029:                           ;       CARGAMOS LOS NUEVOS VALORES PASADOS COMO PARAMETROS
0029:                                   pop     A4L             ; Destino
002F:                                   pop     A2L             ; FinBloque
0035:                                   pop     A1L             ; IniBloque
003B:
003B:                           ;       GUARDAMOS EL ESTADO DE 80STORE Y LO APAGAMOS
003B: AD 1900                           lda     RD80STORE       ; GUARDAMOS EL ESTADO ACTUAL
003E: 85 0A                             sta     STORE80_aux
0040: 8D 0000                           sta     STOR80_OFF      ; LO APAGAMOS
0043:
0043:                           ;       ESCOJEMOS MOVIMIENTO DE ACUERDO CON Mov,
0043:                           ;        Guarda => Carry Set,   Recupera => Carry Clear
0043: 38                                sec                     ; ESTANDAR Guarda
0044: A9 01                             lda     #1              ; MASCARA PARA PROBAR EL PRIMER BIT
0046: 24 06                             bit     Mov             ; PRUEBA LSB DE Mov
0048: F0**                              beq     Guar            ; SI ES CERO Guarda
004A: 18                                clc                     ; SI NO, Recupera
0048* 01
004B: 004B                      Guar    .EQU    *
004B:
004D:                           ;       SALTA A AUXMOVE_ROM
004D: 20 11C0                           jsr     AUXMOVE_ROM
004E:
004E:                           ;       RECUPERAMOS LOS VALORES DE LAS VARIABLES DE AUXMOVE_ROM (A)
004E:                                   mueve   A1L_aux,A1L
0056:                                   mueve   A2L_aux,A2L
005E:                                   mueve   A4L_aux,A4L
0066:
0066:                           ;       SELECCIONAMOS EL BANCO AUXILIAR CERO
0066: A9 00                             lda     #0
0068: 8D 7500                           sta     SwBank
006B:
006B:                           ;       RECUPERAMOS EL ESTADO ANTERIOR DE 80STORE
006B: 8D 0100                           sta     STORE80_ON      ; ESTANDAR PRENDIDO
006E: 24 0A                             bit     STORE80_aux     ; PRUEBA EL ESTADO ANTERIOR
0070: 30**                              bmi     Prend           ; SI BIT 7 = 1, PRENDIDO
0072: 8D 0000                           sta     STOR80_OFF      ; SI NO, APAGADO
0070* 03
0075: 0075                      Prend   .EQU    *
0075:
```

```
0075!                       ;       RECUPERAMOS LA DIRECCION DE RETORNO
0075!                               push Return
007B!
007B!                       ;       REGRESAMOS A Pascal
007B! 60                            RTS
007C!
007C!                               .END                    ; [au.move]
```

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def FR - Proc FC - Func
PB - Public PV - Private CS - Consts

AIL	AB	003C	AILAUX	AB	0004	AZL	AB	003E	AZLAUX	AB	0006	AAL	AB	0042	AALAUX	AB	0008	AUXMOVE	PR
AUXMOVER	AB	C311	GUAR	LB	004B	MOV	AB	0000	MUEVE	MC	----	POP	MC	----	PREND	LB	0075	PUSH	MC
RB8KSTOR	AB	C018	RETURN	AB	0002	STORE0F	AB	C000	STORE80A	AB	000A	STORE800	AB	C001	SWBANK	AB	C075		

Current minimum space is 8470 words.

Assembly complete: 166 lines
0 Errors flagged on this Assembly

7.4 BIBLIOGRAFIA

Revistas

- Popular Computing
  Febrero 1982
  Febrero 1983
- Scientific American
  Diciembre 1982
- Personal Computing
  Agosto 1982
  Abril 1983
  Marzo 1984
  Mayo 1984
- Creative Computing
  Junio 1983
- Fortune
  Octubre 1983
  Enero 1984
- Byte
  Noviembre 1983
  Febrero 1984
- Get More Out of your PC
  Edición 1985
- A +
  Enero 1985
- PC World
  Octubre 1984
  Enero 1985
- Compute!
  Marzo 1985
  Artículo: Speed Script 3.0 un procesador de palabras en código de máquina
  Junio 1985
  Artículo: Speed Script
- Nibble
  Agosto 1984
  Artículo: Disassembly Lines, DOS 3.3 File Manager

Manuales

-"Apple Programmer's Handbook", Paul Irwin
Sams & Co., 1984
-"Beneath Apple Dos", D.Worth & P.Lechner
Omality Software, 1982
-"Beneath ProDOS", D.Worth & P.Lechner
Omality Software, 1985
-"Apple Writer //e"
Apple Computer Inc., 1983
-"Jack 2", "Incredible Jack"
Bussiness Solutions Inc., 1983 y 1982

-"Compute!'s First Book of Apple"
Compute! Publications, 1984
-"Assembly lines, The Book", Roger Wagner
Softalk Publishing, 1982
-"ProDOS & Profile support tools, Technical notes"
Apple Computer Inc,, 1984
-"Device & Interrupt Support Tools Manual"
Apple //, Pascal 1.2 Work Bench
Apple Computer Inc., 1984
-"ProDOS Users Manual"
Apple Computer Inc., 1983
-"Apple ProDOS Advanced features for programmers", Gary B.Little
Prentice Hall-Brady, 1985
-"ProDOS Technical Reference Manual"
Apple Computer Inc., 1985
-"Basic programming with ProDOS"
Apple Computer Inc., 1984
-"Super Ram, 512 K"
Computación y Servicios Auxiliares, 1983
-"Mouse Toolkit","Desktop Toolkit (Apple //e)",
"Apple //e Graphics Toolkit"
Apple Computer Inc., 1985
-"Manual del Usuario de Apple Mouse //"
Apple Computer Inc., 1983
-"80 Column Text Card Manual"
Apple Computer Inc., 1982
-"Apple Pascal. Languaje Reference Manual"
Apple Computer 1985
-"Apple Pascal. Operating System Reference Manual"
Apple Computer 1985
-"Apple Pascal. Reference Manual"
Apple Computer 1985
-"Apple//, The DOS Manual"
Apple Computer 1981
-"Apple // Instant Pascal, Language Reference Manual"
Addison-Wesley Pub. Co., 1985
-"Pocket Guide to Instant Pascal"
Apple Computer, Inc., 1985
-"How to write a computer manual", Jonathan Price
The Benjamin/Cummings Publishing Company, 1984
-"Software Engineering for micros", T.G. Lewis
Hayden Book Company, 1979
-"Principles of Interactive Computer Graphics", William Newman
McGraw-Hill, Inc., 1979