Universidad Nacional Autónoma de México

Facultad de Ingeniería

PROCESADOR DE TEXTO PARA USO INFANTIL

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

Que para obtener el título de:

I N G E N I E R O E N
C O M P U T A C I O N

P r e s e n t a :

Santiago Dario Ortega Aceves
José Manuel Hernández Guzmán

Director de Tesis :
Marco Antonio Murray Lasso

México, D. F. 1993.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Indice

Introducción

En nuestros días resulta incuestionable el impacto que han tenido los avances en computación sobre áreas cada vez más extensas de la actividad humana. Estas áreas van de la medicina a la música y de la administración al diseño.

La enorme importancia de las computadoras en cada una de las actividades del hombre moderno ha provocado la necesidad de desarrollar un software adecuado para ellas: ingeniería, ciencias, administración, economía, medicina, etc.

La computadora es una herramienta que ayuda en las tareas rutinarias y pesadas, que ha agilizado los trabajos que un hombre por sí sólo no podría realizar. La computadora ha llegado ha ser indispensable en muchos casos, como: cálculos matemáticos, procesamiento y consulta de información, desarrollo de documentos, etc. El desarrollo de documentos es un caso particular en donde este fenómeno es patente, y ello hace necesario la utilización de un procesador de texto para facilitar la elaboración e impresión de ellos.

Los procesadores de texto pretenden dar cada vez más posibilidades y facilidades en su uso. Algunos de estos sistemas son elaborados y dirigidos a diversas áreas y personas, olvidándose de alguna manera de un núcleo muy importante, como son los niños.

El propósito de esta tesis es crear un procesador de texto en donde el niño no encuentre vedada una herramienta de trabajo que puede ser fundamental para su futuro cercano.

La mayoría de los procesadores de texto fueron enfocados para su uso en su país de origen. México, por ser un país en vías de desarrollo generalmente importa tecnología de países desarrollados, en particular de Estados Unidos. En esas importaciones se incluye también al software. Es por eso que los procesadores que importamos están hechos en el idioma inglés, y muchos de ellos no incluyen los caracteres especiales del español (ñ, Ñ, ¿, á, é, í, ó, ú, ü, Ü, etc.). Estos procesadores sólo manejan el teclado en inglés, olvidando dichos caracteres, y en caso de incluirlos dificulta su utilización e impresión.

Surge por ello, la necesidad de elaborar un procesador de texto en español para que el usuario no tenga una doble dificultad en la traducción y razonamiento de los términos, al igual que la utilización e impresión de los caracteres especiales mencionados. Conviene agregar que no existen procesadores de texto enfocados hacia la niñez, elaborados a partir de las características intrínsecas del niño. Es por ello de singular importancia crear un procesador que llene todas estas carencias.

El desarrollo del procesador de texto para uso infantil, busca, en primera instancia, eliminar una infinidad de funciones que no son de uso frecuente, que confunden al niño o no son de utilidad a su nivel. Se pretende de igual manera, erradicar la utilización de una serie de combinaciones simultáneas de teclas para accesar una función, que complicarían el manejo y uso para los niños.

El procesador comprende las siguientes funciones básicas:

Escribir Texto	.-	Crear un documento.
Grabar Texto	.-	Salvar o almacenar el texto escrito en disco (A, B, C,...) y directorio.
Recuperar Texto	.-	Traer de Disco un archivo de trabajo.
Insertar Texto	.-	Adicionar texto en cualquier parte del documento.
Borrar Texto	.-	Borrar texto no deseado del documento.
Movimiento del	.-	Movimiento del cursor entre el límite curso de líneas y columnas del texto.
Búsqueda	.-	Busca a través del texto alguna cadena de caracteres.

Para la impresión, se cuenta con los manejadores de impresora Epson e IBM compatibles, ya que la mayoría de las impresoras pueden emularlas. Algunas de las características en la impresión, son el centrado de texto a lo largo y ancho de la página, salto automático de hoja, colocación del número de página y la opción de letra de calidad. Para ello se necesitó estudiar y manejar las secuencias de escape de las impresoras Epson e IBM, para así inicializarlas, checar si existe impresora en el puerto, hacer el salto de página, letra de calidad, etc., sabiendo claramente la necesidad de imprimir los caracteres especiales del español.

Como una herramienta necesaria para el niño, se pensó colocar un diccionario que permita al infante consultar el significado de alguna palabra y adicionar palabras si así se desea. Ello implicó investigar el conjunto de palabras más usuales que un niño puede manejar y/o necesitar. Debiéndose al mismo tiempo seleccionar un buen método de búsqueda y almacenamiento de información (palabra-significado).

Con lo anterior parecía ya solucionado el problema de elaborar un procesador de texto para uso infantil. Pero la forma de presentación del mismo en el monitor resultaba de capital importancia. Por esta razón, el tipo de letra utilizado dentro del display es de un tamaño mayor; la que nos proporciona el monitor de 40x25. El manejo es a través de un menú principal, con figuras alusivas a cada función del procesador, estableciendo de esta manera una estructura razonable para el infante. Dichas características en su conjunto, no son encontradas en ningún otro procesador.

Por otra parte, el software seleccionado para la elaboración del procesador es el Lenguaje C, debido a que C es un lenguaje de programación de empleo general, caracterizado por su concisión y por poseer un moderno flujo de control y estructuras de datos, así como un rico conjunto de operadores. C no es un lenguaje de "muy alto nivel" ni "grande", no estando especializado en ningún área particular de aplicación. En cambio, su carencia de restricciones y su generalidad lo hacen más eficaz y conveniente para muchas tareas que otros lenguajes, supuestamente más potentes. Por ejemplo C permite tener acceso a interrupciones de BIOS (Basic Input/Output System). Y su cercanía a un lenguaje de bajo nivel lo hace más rápido.

Finalmente, al realizar este sistema se ha pretendido colaborar en la solución de los problemas de derechos de autor, al proporcionar a nuestra casa de estudios este procesador de texto y evitar la piratería. Por lo tanto el desarrollo del presente, busca crear un ambiente donde el niño tenga un procesador amigable y las instituciones pertenecientes a la universidad puedan usarlo sin el problema de licencias de software.

Capítulo I **Planteamiento y análisis del problema**

I.1 ¿Qué es un procesador de texto?

Es un sistema que nos da la facilidad de crear, corregir, mejorar, guardar o imprimir un documento. Permitiendo incluir páginas enteras de información, borrar parte del texto, centrar líneas, subrayar caracteres, resaltar palabras, seleccionar tipos de letras, hacer notas de pie de página, etc. De acuerdo a la sofisticación del procesador, algunos permitirán incluir gráficas e inclusive corregir ortografía. Es decir, al utilizar un procesador de texto se aumentará la creatividad, productividad y claridad en los trabajos, el tiempo y desperdicio de papel serán mucho menores.

Al hacer un poco de historia alrededor de los procesadores de texto, es posible mencionar parte de su evolución a través de los siguientes acontecimientos:

a) El primer acercamiento hacia los procesadores de texto es por medio de las máquinas de escribir. Pero si recordamos un poco más, podemos considerar la invención de la imprenta, como uno de los acontecimientos más importantes de la historia; hecho con el que empezó la difusión masiva de la 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 se inventaron las máquinas eléctricas de escribir, mejorando en mucho a las antiguas máquinas mecánicas. Pero la esencia de las máquinas de escribir no cambió; si se cometían errores, las correcciones seguían realizándose de igual manera que antes.

En 1964, la empresa Internacional Business Machines (IBM), dio 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.

En la década de los setenta, 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 firma Vydec presentó una máquina similar, usando un disco magnético, lo que incrementó su 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 área de estudio son las microcomputadoras, podemos agregar que las micros se fueron desarrollando paralelamente a ese tipo de máquinas, solamente que ahora son más comunes las primeras que los procesadores dedicados.

Finalmente las microcomputadoras dieron origen a editores sencillos de texto, que al evolucionar fueron muy similares a los procesadores dedicados.

Ya en nuestros días, un conjunto de programas para el procesamiento de palabras, son un módulo de los paquetes del tipo software integrado.

Según se van haciendo más comunes los programas de procesamiento de texto, mucha gente se va dando cuenta de la enorme diferencia entre usar un procesador de texto y una máquina de escribir. Con un procesador de texto, el revisar un texto es extremadamente fácil, reestructurar una carta o una memoria es muy rápido y conservar texto en un disco significa que nunca tendrá que volver a escribirlo.

I.1.1 ¿Cómo trabaja un procesador de texto?

Probablemente nos resulta fácil imaginar un programa que visualiza los caracteres en la pantalla cuando los introduce por el teclado. Exceptuando la pantalla, el programa actúa igual que una máquina de escribir normal. También uno puede imaginar que cuando se presionan las teclas, el programa los guarda en un archivo; posteriormente se puede dar una orden para imprimir el archivo en la impresora. Hasta ahora, esta descripción coincide con la de una máquina de escribir con memoria: una vez que se escribe algo, siempre se puede imprimir.

Sin embargo, la similaridad termina cuando se quiere modificar lo que se ha tecleado. Con un procesador de texto se pueden añadir o borrar palabras, frases o párrafos, fácilmente. También se pueden trasladar frases y párrafos por todo el archivo texto que se ha creado. Al hacer estos cambios, el programa está permitiendo editar el texto. Otra característica de un procesador de texto es que se puede diseñar un

documento para que tenga un formato particular (numeración de página automática, sangría, letra cursiva, etc.) cuando se imprime en papel.

En la mayoría de los programas se utilizan una combinación de pulsaciones de teclas para dar estas órdenes (como por ejemplo, para moverse hacia adelante o hacia atrás en el texto, para borrar, para insertar, etc.). La mayoría de los teclados de las microcomputadoras tienen teclas especiales que, controladas por el sistema (procesador), hacen más fácil el manejo de textos.

Los documentos que se escriben con un procesador de texto se guardan en un archivo. Si se desea introducir un nuevo archivo, se le indica al programa que se quiere introducir texto y a continuación se comienza a teclear. Si se edita el texto que se escribió, se indica al programa el nombre del archivo que se quiera editar, y dónde está dentro del texto lo que se quiere editar (como por ejemplo, la primera línea o la segunda frase del párrafo cuarto). Cuando el programa lo lleva ahí, podrá hacer los cambios. En realidad, se mueve de esta forma a lo largo de todo el archivo, cambiando el texto antiguo o añadiendo nuevo texto. Una de las características agradables de la edición es que se puede mover hacia atrás y hacia adelante por el archivo a voluntad; no hay la limitación de ir sólo del principio a fin.

Se debe guardar el trabajo periódicamente en el dispositivo de almacenamiento, mientras se está escribiendo y guardarlo nuevamente cuando haya terminado de editar o de intoducir el texto en la máquina. Si desea, puede imprimir este nuevo archivo. Después de haber usado el programa de procesamiento de texto para editar un documento, difícilmente se querrá volver a utilizar una máquina de escribir.

La mayoría de los procesadores de texto muestran cómo aparecerá el texto en la página. Otros, sólo el contenido del archivo en pantalla. Con ellos, no se verá el formato del texto sino hasta que se mande el archivo a la impresora. Se utilizan órdenes de formato de texto para decir a la impresora cómo imprimir el texto.

Generalmente, cuanto más completo es el procesador de texto, más cosas se pueden hacer con sus órdenes de formato. Hay muchos tipos de órdenes de formato. Algunas permiten cumplir requisitos sencillos (como poner los márgenes izquierdo y derecho, numerar páginas o hacer una sangría en un párrafo), mientras que otras son más sofisticadas. Por ejemplo, tales órdenes pueden hacer funciones como un sangrado automático en una lista, centrar los títulos de las selecciones , poner un encabezado específico en la cabecera de cada página, o poner en negritas o subrayar automáticamente un grupo de palabras.

Debe asegurarse de que el paquete que se usa es un procesador de texto y no simplemente un editor de texto (a no ser que se quiera esto último). Hay una gran diferencia entre los dos: los editores de texto no dan formato al texto y lo usan los programadores (puesto que los programas no necesitan el formato) o por gente que tiene un paquete separado de formato de textos.

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.

I.2 Procesadores convencionales: Características y usos

Esta sección proporciona una visión general del mercado, describiendo algunos de los programas más comunes.

Volkswriter

Uno de los paquetes más fáciles de usar es *Volkswriter*, de Lifetree Software. En el cual, en vez de un largo manual describiendo cómo usar el editor, la mayoría de las instrucciones se dan en unos cuantos archivos. Para aprender el sistema, se leen los archivos y prueba cada orden de edición cuando se le presenta. El formateo de texto es, desgraciadamente, inadecuado cuando se trata de algo más que cartas. Sin embargo, el paquete es muy popular entre los principiantes porque no es caro, tiene muchas características de edición y proporciona una buena introducción a la utilización de los procesadores de texto. *Volkswriter de luxe* tiene más características para usuarios avanzados, pero cuesta más.

Superwriter

Otro paquete fácil de aprender es *Superwriter*, de Sorcim. El programa trae instrucciones completas en un manual, además de un manual suplementario denominado *10 Minutes to Superwriter* que enseña a crear, editar e imprimir una carta en 10 minutos. Aunque algunas de las peticiones de órdenes no son muy claras, está disponible una pantalla de ayuda para cada petición de orden. Aquellos que han usado el popular programa de hoja de cálculo SuperCalc de Sorcim, reconocerán la mayoría de las peticiones de órdenes. Otra característica muy útil de *Superwrite* es la historia del documento, que mantiene al tanto de quien creó un documento y cuándo fue actualizado

por última vez. Esto puede ser muy importante en una oficina donde mucha gente trabaja en un documento.

WordVision

WordVision, de Bruce & James Program Publishers, es un paquete de procesamiento de texto mucho más completo que Volkswriter o Superwriter. El manual es muy fácil de seguir, aunque lleva bastante tiempo para empezar algún procesamiento de texto útil. El paquete incluye pequeñas etiquetas autoadhesivas que se pegan en las teclas; desafortunadamente, estas etiquetas hacen que el teclado parezca atestado. También puede pedir capuchones para las teclas, que son semipermanentes y más fáciles de usar. Aunque muchos programas pretenden mostrarle en la pantalla el aspecto que tendrá exactamente la salida, Wordvision lo hace mejor que ningún otro. Su uso del color también hace que ciertas características sean más fáciles de ver, por ejemplo, el texto en negritas se destaca del texto normal usando un color diferente en la pantalla (en monitores monocromáticos, esto se visualiza como un nivel diferente de luminosidad).

The Final Word

Si decide que no necesita ver el documento formateado inmediatamente, *The Final Word*, de Mark of the Unicorn, es un paquete excelente. Las órdenes de edición de texto son más fáciles de recordar que las de cualquier otro paquete y las posibilidades de la memoria ampliada del MS-DOS se utilizan en su más completa extensión. Es bastante fácil de usar las órdenes de formateado para hacer informes avanzados y manuales, pero la documentación de las órdenes de formateado no es muy explícita. *Perfect Writer*, de

Perfect Software, tiene muchas de las características de The Final Word, pero son más fáciles de usar en The Final Word.

PFS:Write

PFS:Write, de Software Publishing Corp., tiene una ventaja interesante sobre otros muchos paquetes: permite mezclar con el texto gráfico de negocios del programa PFS:Graph. Los gráficos pueden hacer que la salida sea mucho más interesante y pueden facilitar la producción de informes. PFS:Write tiene sin embargo una desafortunada limitación: cada archivo que se edita tiene que ser pequeño. Esto obliga a unir varios archivos si se está escribiendo un documento largo.

PeachText

PeachText 5000, de Peachtree Software, es un procesador de texto que permite formatear documentos para impresión, pero no muestra formato de la página por pantalla. Muchos usuarios creen ciegamente en PeachText, mientras que otros piensan que el programa hace frecuentemente suposiciones incorrectas de lo que ellos quieren. Por ejemplo, la mayor parte de las medidas de seguridad del paquete requieren que se den dos grupos de pulsaciones de teclas, uno para dar la órden de edición y otro para confirmar que se quiere ejecutar la órden. Además, el manual no proporciona indicaciones completas de cómo usar las características de formateado. Esto las hace muy difíciles de descifrar.

Multimate

Multimate, de Softword System, hace que su computadora parezca un procesador de texto avanzado. En realidad, si está familiarizado con los procesadores de texto de

Wang, se sentirá muy cómodo con Multimate. Características como las líneas de formato global (áreas de texto que describen el formato del documento) son idénticas al procesador de texto Wang.

WordPerfect

WordPerfect, de Satellite Software International, tiene una multitud de características de edición y formateado. Es capaz de editar dos documentos a la vez. El formateador permite poner múltiples columnas de texto en la página y tener varias líneas en los encabezados y pies de página. Este paquete es muy completo y, aunque el manual no está escrito para el principiante, se pueden usar muchas de las órdenes inmediatamente.

Benchmark

Benchmark, de Metasoft, es uno de los paquetes más difíciles de aprender. El usuario de este paquete se ve obligado a pasar por una gran cantidad de preparativos antes de poder siquiera ejecutar el programa. El gran número de opciones técnicas que se le presentan durante la instalación, puede intimidar a los principiantes. El programa tiene algunas características interesantes, como una calculadora interna, que no se encuentra en la mayoría de los procesadores de texto. Sin embargo, las deficientes instrucciones pueden hacer difícil el aprovechamiento de estas características.

VisiWord

Algunos sistemas nos bombardean con una serie de mensajes de ayuda y pantallas informativas. *VisiWord*, de VisiCorp, es uno de ellos. Los mensajes por pantalla y el manual bien escritos son importantes cuando se está aprendiendo y proporcionan una

asistencia suficiente para los usuarios poco frecuentes. Pero una vez que se aprende el paquete, es factible pensar que los mensajes estorban. Una gran ventaja de VisiWord sobre otros procesadores de texto es la facilidad con que puede usarse para formatear tablas con números.

Word

Microsoft Word, o simplemente Word, es probablemente el sistema de procesamiento de texto más potente disponible para computadoras. Es fácil de aprender, pero puede que se sienta abrumado por el número de características avanzadas. Word le permite usar un ratón, que es un dispositivo controlado con la mano que se usa para mover una flecha por la pantalla. El ratón apunta a las distintas opciones en el menú de órdenes, a una palabra (o grupo de palabras) para trasladar o borrar, o simplemente a la posición donde quiere comenzar a introducir texto. Algunos usuarios se quejan de que usar un ratón es molesto, puesto que la mano tiene que dejar el teclado para moverlo. Sin embargo, se puede usar Word sin el ratón.

Microsoft Word utiliza "glosarios", que son colecciones de reglas relativas al texto que se guarda en disco. Por ejemplo, puede indicar que cada vez que haya un título, debe centrar la línea e imprimirla con caracteres en negrita. A continuación, lo único que se necesita hacer es marcar una línea como título y Word la formateará. En realidad, se pueden tener muchos glosarios con reglas diferentes. Hay otras muchas características avanzadas que hacen de Microsoft Word una magnífica opción para un procesador de texto con todo tipo de funciones.

Textra

Textra, de Ann Arbor Software, es uno de los mejores paquetes de procesamiento de texto a bajo precio. Tiene muchas de las características de los paquetes más grandes (como por ejemplo, ver anticipadamente el texto formateado) y un programa tutorial muy bueno que enseña a manejar el producto, de una forma muy simple.

WordStar

WordStar, de Micropro, es uno de los programas de más venta. Fue uno de los primeros editores de pantalla para microcomputadoras. Muchos distribuidores de software piensan que es fácil de vender debido al fácil reconocimiento de su nombre. Sin embargo, no es tan fácil de usar como muchos de los otros programas mencionados aquí. Sus ordenes son fáciles de olvidar y algunas personas que lo usan a diario piensan que todavía necesitan tener a mano una relación de las órdenes.

Desde una perspectiva optimista, hay más de una docena de libros disponibles para aprender WordStar y hay más cursos de formación para el WordStar que para cualquier otro programa de procesamiento de texto. Con WordStar todas las opciones de edición se visualizan en la pantalla a cada momento. Algunos encuentran que esta característica no es conveniente porque el menu quita casi un tercio del área de texto utilizable.

Otros procesadores de texto populares para computadoras personales son *Palantir*, de Designer Software, *WordPlus*, de Professional Software, *Easy Writer*, de Information Unlimited Software, *XyWrite II*, de Xyquest, *ReadiWriter*, de ReadiWare Systems y *Blue*, de Symmetric Software.

I.3 El niño, su ambiente y educación

En los últimos años, la educación preescolar-primaria ha registrado, en forma paulatina, la introducción de la computación en las actividades escolares de los mismos, de tal manera que ha surgido la necesidad de crear programas específicos para este nivel; este trabajo no ha sido fácil debido a que la mayor parte de los estudios en computación en un principio estuvo enfocada hacia estudios superiores.

Debido a esto se busca la manera de cómo hacer llegar la computación a los niños en forma atractiva y amena para apoyar la enseñanza de esta manera.

Uno de los objetivos de los programas elaborados para la educación tiene como finalidad la utilización del poder de razonamiento, del juicio crítico y de la creatividad del niño; cuidando de no caer en una situación equívoca, esto es, el manejo mecánico de la máquina.

El niño, al encontrarse en la etapa inicial de su vida, lleva a cuestas un conjunto de incógnitas que giran a su alrededor. Busca a cada instante, respuesta a las inquietudes más significativas.

En este momento es cuando el niño tiene una gran energía que es necesario canalizar. Y que mejor si se canaliza hacia una educación que permita su buen desarrollo, utilizando nuevas herramientas, como la computadora, dispositivo que estimulará nuevas inquietudes en su formación.

Actualmente, las escuelas requieren dar una educación que proporcione una preparación, con enfoque en las futuras profesiones. Por ello, son importantes todas las

innovaciones que puedan beneficiar a la educación, como sucede ahora con la computación.

Así como los libros han sido, por siglos, importantes en la educación, las computadoras ocuparán un lugar decisivo en la escuelas. Por lo tanto, en el presente se debe preparar a los niños para la vida en el siglo XXI. En consecuencia, en las escuelas se necesitará crear la infraestructura computacional para explotarlas completamente y dar al niño un equivalente de confianza como con los libros.

Las actividades que puedan desarrollarse con la computadora, fortalecerán diferentes aspectos de la enseñanza y aprendizaje en la escuela.

La enseñanza apoyada en la computadora:

- Permitirá a los maestros extender e intensificar la educación de los niños.
- Permitirá a los niños adquirir la destreza, al usar la computadora como una herramienta, para sus propósitos.
- Dará al niño, un mejor entendimiento de la forma en que la tecnología afecta a sus propias vidas.
- Motivará al niño en su educación.
- Generará una interacción y retroalimentación inmediata, etc.

I.3.1 Aptitudes desarrolladas

El trabajar con computadoras, contribuye al desarrollo de ciertas habilidades y aptitudes:

Comunicación. Las computadoras pueden representar un punto de reunión que propicia la comunicación entre los niños. Ello también ofrece caminos de representación y manipulación de palabras, números, símbolos, formas o sonidos en el formato de texto, en una figura o diagrama, en un modelo, en una tabla o gráfica.

Estudio y observación. Una computadora puede ofrecer a los niños, oportunidades para seleccionar y extraer información, para arreglar y reacomodar datos, de modo que ellos puedan más facilmente reconocer similitudes y diferencias, o modelos y relaciones, y obtener conclusiones.

Solución de problemas. En muchas de las actividades que se le presentan al niño, él puede resolver sus problemas con la computadora. Esto permitirá, al mismo tiempo cierta habilidad para identificar problemas, planear, elegir y comparar estrategias.

Creatividad e imaginación. Existe software que permite al niño crear sus propios programas, para hacer dibujos, música o elaborar documentos.

Consecuencias personales y sociales. Al existir un punto de reunión (la computadora), particularmente donde hay responsabilidad para la toma de decisiones, los niños consideran los puntos de vista de otros, para contribuir y cooperar, o tomar la idea más apropiada en la solución de los problemas.

Ayudar a los niños a apreciar el potencial de los paquetes de software, y desarrollar aptitudes en el uso de ellos, no es una tarea fácil. Los maestros necesitan tiempo para aprender, planear y aplicar. Los niños absorben las nuevas ideas en diferentes proporciones y formas.

I.3.2 El niño y el procesador

El procesador de texto es una clase de software que ayuda al manejo de información. Y como tal, permite conjuntar datos e ideas que constituirán un texto.

Los niños de escuela primaria pasan una gran cantidad de tiempo escribiendo historias cortas o reportes, y tratan de lograrlo al primer intento con buena presentación y sin errores. Pero en general ello no es posible, pues en esta edad es más común tener una gama de errores que se concentra en el perfeccionamiento de palabras, gramática y caligrafía. Además de ello se consideran las frases y palabras alternativas, siguiendo un orden lógico. Son partes difíciles de lograr en primera instancia, llegando a ser tedioso y cansado, a la vez de consumir tiempo para reescribir cada borrador a mano.

Es así como el niño encuentra un conjunto de problemas en la elaboración de documentos por no contar con herramientas adecuadas para trabajar.

Al utilizar procesadores de texto, se ayuda en la solución de estos problemas y al mismo tiempo se propician ciertas cualidades en el alumno. En primarias, secundarias y preparatorias que incluyen esta educación computacional, se ha encontrado que los alumnos pueden desarrollar:

1) Su creatividad.

2) La habilidad para resolver problemas.

3) Su redacción y ortografía.

4) Su expresión escrita.

5) El trabajo en equipo

Y conforme los alumnos empiezan a dominar el procesador, escribir se convierte en una actividad divertida porque la calidad de sus trabajos y el interés por hacerlos se incrementa notablemente.

La actitud del niño hacia la tarea de escribir así como su interés por expresarse en forma escrita mejoran considerablemente.

La importancia de utilizar su trabajo escrito, aunque solo sea para leerlos en voz alta, permite una mejor comunicación con sus compañeros, maestros y familiares.

I.4 Análisis de características deseables en un procesador para uso infantil

Después de conocer los procesadores convencionales y entender las cualidades de un niño es posible determinar las características que debe tener un procesador de texto para uso infantil.

Uno de los primeros requerimientos para el procesador es el idioma. Este factor permite tener el primer contacto entre el programa y el usuario, que de ser diferente a su lengua materna, se dificultaría el entendimiento inicial del procesador con su conjunto de comandos que se le están mostrando. En nuestro caso la lengua materna es el español, que facilita la comunicación escrita y ayuda a la comprensión del procesador, siendo la

intención de su elaboración. Se evitan así traducciones inecesarias por importar software que generalmente se encuentra en otro idioma.

Respecto a los teclados, se sabe existen diferentes tipos de plantillas que identifican a cada una de las teclas, y por ende al carácter. Es decir, la plantilla permite localizar las posiciones de los caracteres en el teclado. Pueden, por ello, aparecer caracteres distintos en posiciones distintas, según el lenguaje del país seleccionado con los mandatos del sistema operativo de la computadora. Por lo tanto, si no coincide el tipo de plantilla del teclado con el seleccionado a través del sistema operativo, causará errores entre el carácter del teclado y lo visualizado en el monitor, en el momento de teclear. Otro problema que existe con los teclados existentes en el mercado, es que no todos tienen físicamente la plantilla del teclado latinoamericano, que permita escribir directamente los caracteres especiales del español (ñ, Ñ, ¿, á, é, í, ó, ú, ü, Ü, etc.). Para este caso, el procesador a elaborar contará con la alternativa de poder teclear directamente los caracteres especiales, si la plantilla de teclado pertenece al latinoamericano, o utilizar una combinación de dos teclas para obtener los mismos.

La edición contemplarará las siguientes características: movimiento del cursor, grabación, recuperación, inserción, borrado y búsqueda de texto. En un despliegue que visualizará los caracteres, con un tamaño mayor al monitor normal de 80 X 25 caracteres.

En la impresión, se manejarán las impresoras Epson e IBM compatibles, por ser las más comunes dentro del mercado y las que generalmente son emuladas por otras impresoras. Por consiguiente es necesario estudiar los códigos de control y las secuencias

de escape que corresponden a estos dispositivos, para adicionar posibilidades al procesador.

Los códigos de control y las secuencias de escape son mensajes especiales que el procesador tendrá que enviar a la impresora, los cuales modificarán la forma en que la computadora imprime hasta ese momento y le permitirá controlar el aspecto de la salida impresa.

La impresora recibe cada código de control como un carácter único de información, estos códigos pueden desplazar la cabeza de impresión en distintas direcciones y modificar la calidad que la impresora está produciendo.

Las secuencias de escape consisten en dos o más caracteres de información. Al igual que los códigos de control, permitiendo modificar la forma en que la impresora ha estado funcionando hasta el momento. Por lo tanto, el procesador debe contar con estas opciones, para poder tener el control sobre el aspecto de la salida impresa.

El primer carácter de una secuencia de escape es siempre <Esc>. El carácter de escape le indica a la impresora que trate la información que sigue a continuación como datos de control y no como datos a imprimir. Los códigos de control y las secuencias de escape pueden colocarse en cualquier lugar dentro de la secuencia de datos que se envía a la impresora.

Por lo tanto, el procesador incluirá archivos[1] que convertirán los mensajes del texto hacia la impresora.

---

1 Estos suelen denominarse archivos de descripción para la impresora o direccionadores de impresión.

Otro aspecto que se encontró necesario, es la adición de un diccionario, para consulta de significado de palabras. Coadyuvando a corregir los errores y dudas ortográficas que pudiera tener el niño, al mismo tiempo de proporcionarle el significado de la palabra. Otra opción consiste en poder aumentar "n" palabras al diccionario.

El número de palabras que contendrá el diccionario, estará basado en una investigación exhaustiva de la cantidad de palabras que puede utilizar un niño.

Finalmente, un punto que también tiene importancia, es la presentación del procesador en pantalla. Lugar donde se desplegará todo el conjunto de comandos, con sus respectivas respuestas a cada llamado. Se debe tener una estructuración que haga sentir seguro al niño respecto de los comandos a invocar y de la respuesta de los mismos. Se establece para ello un desarrollo a través de menús con figuras alusivas al comando. Se le agregará sonido al escribir el texto por cada tecla oprimida.

Capítulo II Selección del lenguaje y equipo

II.1 Clasificación de lenguajes

La gran mayoría de los lenguajes de programación se han transferido a microcomputadoras, y ya no es necesario tener acceso a grandes equipos para poder hacer uso de alguno de ellos, ya que se pueden tener al alcance fácilmente en una computadora personal. No obstante, muy pocos sabrán la gran variedad de características que hacen diferir a unos de otros, para lo cual se clasifican los lenguajes en cinco grandes grupos :

- Ensambladores.
- De Desarrollo de Sistemas.
- De Alto Nivel.
- De Alto Nivel Estructurados.
- De Alto Nivel Dinámicos.

II.1.1 Lenguajes ensambladores

Los ensambladores permiten escribir programas a nivel de lenguaje de máquina, las personas que programan así, deben conocer profundamente la arquitectura de la computadora y su lenguaje de máquina. El compilador del lenguaje ensamblador traduce cada instrucción , escrita por el programador, a la instrucción o instrucciones de ensamble equivalentes en lenguaje de máquina de la computadora (código binario).

Es difícil que un programa ensamblador realizado por un programador se pueda modificar por otro programador, a menos que el programa se encuentre bien documentado. Generalmente, los programas componentes de un sistema (como el Sistema Operativo y los sistemas de manejo de Base de Datos), se apoyan en este tipo de lenguajes.

Realmente el uso de lenguajes ensambladores se limita para ocasiones en que se tengan necesidades especiales de velocidad o manejo de dispositivos, como en aquellos procesos que durante su ejecución tengan partes críticas y demanden rapidez en el manejo de algunos elementos de la computadora.

II.1.2 Lenguajes de desarrollo de sistemas

Se derivaron de los lenguajes ensambladores cuando se notaron las grandes dificultades que presentaban estos. Los lenguajes de Desarrollo de Sistemas, proveen facilidades de estructura, control de flujo en los programas, y hasta funciones de alto nivel. Entre estas se anotan 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 complejo. Un ejemplo muy conocido de estos lenguajes es C.

II.1.3 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. Los primeros lenguajes de este tipo son: BASIC, FORTRAN Y COBOL.

II.1.4 Lenguajes de alto nivel estructurados

Se derivan de los lenguajes de alto nivel, pero se distingen por tener construcciones de control variadas, tales como IF-THEN-ELSE, WHILE, REPEAT, y por poder clasificar objetos (valores) como de cierto tipo[2]. 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 modulariadad y facilitan la programación estructurada. Ejemplo bien conocidos son : PASCAL Y ALGOL.

II.1.5 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. Ejemplo de este tipo de lenguajes son : APL Y LISP.

---

2 tipo : character -- caracter, integer -- entero, real -- real, etc.

II.2 Comparación de lenguajes

A continuación hablaremos de algunos de los lenguajes de uso más común en el desarrollo de sistemas y programas en microcomputadoras.

II.2.1 BASIC

Desarrollado por John Kemeny y Thomas Kurtz en 1965, BASIC no es un lenguaje estructurado ni poderoso, se puede clasificar como un lenguaje de alto nivel estático. Aparace como un lenguaje para microcomputadoras, pequeño (en cuanto al espacio que el sistema ocupa en memoria), fácil de aprender, usar y tal vez el lenguaje actual más popular.

La principal razón del éxito de BASIC es el hecho de ser interprete y venir como lenguaje estándar en muchas computadoras. El ser interprete le da la ventaja de desarrollo de programas en forma interactiva, en el sentido de poder 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, además de 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 obtener el valor de cualquier variable en especial; enseguida realizar la operación que se quiera sobre los datos o sobre el programa, o continuar la ejecución del mismo. Lograr esto en otros lenguajes requiere el uso de herramientas muy sofisticadas de programación que son difíciles de usar.

Una de las desventajas de los lenguajes intérpretes es su lentitud, que se puede resolver actualmente en muchos casos gracias a la aparición de microcomputadoras, conteniendo procesadores con una arquitectura más veloz. Otra desventaja es el hecho de existir muchísimas versiones de BASIC. Muchas empresas al notar las deficiencias de este lenguaje, le hicieron ampliaciones hasta convertirlo en el lenguaje que cuenta con más utilerías, a tal grado que se pueden conseguir muchas versiones de BASIC muy poderosas, tanto como los demás lenguajes. No obstante, toda esta cantidad de versiones son en muchas ocasiones incompatibles, teniendo que realizar siempre una pequeña adaptación, y a veces casi volver a escribir el programa cada que se quiera transportar de una equipo a otro o inclusive a otra versión de BASIC del la misma máquina.

Otra desventaja, es la forma en sí del lenguaje: BASIC es no estructurado, y aunque actualmente se consiguen versiones que supuestamente pretenden serlo, ésto sólo es cierto en algunos aspectos. Aún en las más capaces y nuevas implementaciones se sigue careciendo de muchas facilidades y algunas limitaciones como en lenguajes estructurados. BASIC es muy propenso a generar errores en la lógica de los programas, algunos muy difíciles de corregir.

Una ventaja de BASIC es el dominio total que ofrece sobre la máquina y el sistema en que habita, sobre todo en máquinas pequeñas, puesto que es el lenguaje residente en muchas de ellas. En general, los programas en BASIC no son legibles, aún cuando se puede dividir en módulos y usar 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.

II.2.2 LOGO

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

Este lenguaje es intérprete y por lo tanto, lento. El sistema ocupa una parte importante de la memoria, por lo que no es recomendable para desarrollo de sistemas complicados con problemas de velocidad y espacio en memoria.

II.2.3 FORTRAN (Formula Translator)

Es un lenguaje de programación de alto nivel, creado principalmente para resolver problemas científicos y matemáticos. Desarrollado por IBM, en 1954, para expresar con facilidad las fórmulas matemáticas destinadas a procesos por computadora. Es el más utilizado para problemas científicos, de ingeniería y matemáticos. Este lenguaje es fácil de aprender.

II.2.4 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 pudiera instalarse en casi cualquier máquina y al mismo tiempo convertirse en un estandar de programación.

El lenguaje diseñado por Wirth realmente resultó compacto, fácil de dominar, totalmente estructurado y con formato libre de escritura. Ello hizo que se usara y se siga utilizando ampliamente para escribir algoritmos.

Actualmente es muy usado, tiene varias ventajas para sistemas grandes, como el hecho de evitar muchas de las malas prácticas en programación. Otra ventaja es su facilidad de expresar ideas de una manera natural y similar a una expresión en lenguaje natural. Los programas son muy legibles y no se tiene que hacer un manejo de datos complicado, gracias a su capacidad de definición de tipos de variables, asignación de valores que toman esos tipos o la forma de la variable en sí (por ejemplo, los registros). Con PASCAL se puede desarrollar una programación clara, confiable, mantenible y transportable.

Sus desventajas se encuentran en el desarrollo de sistemas que tengan mucha relación con la computadora y el sistema operativo en general. PASCAL aleja mucho al programador del nivel de máquina, facilitando el trabajo común. En PASCAL es difícil hacer programas que se salgan 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 y memoria (excepto por el uso de funciones escritas en ensamblador, ya que muchas implementaciones de Pascal tienen todas las herramientas para comunicarse con ellas).

II.2.5 ADA

Desarrollado por el Departamento de Defensa de los Estados Unidos, deriva practicamente del Pascal, con la inclusión de conceptos nuevos de programación. Un ejemplo es su capacidad de sustituir eficientemente programas que antes sólo se hacían en ensamblador.

ADA permite hacer una amplia gama de estructuras de datos de programación, restringidas bajo un marco rígido que evita muchos errores, pero que quita la libertad en algunos casos o aumenta el tiempo de ejecución y el tamaño del programa por el código de supervisión que genera (igual que PASCAL). No obstante ello, muchos de estos códigos de supervisión se pueden omitir con las opciones al compilar. No se recomienda mucho su uso, por hacer los programas propensos a errores y difíciles de transportar.

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

La forma de un programa es una serie de rutinas 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), después el cuerpo de las rutinas. También se pueden anidar rutinas dentro de otra global, donde todas las rutinas están anidadas en el programa principal. Esta forma le resta poco la modularidad al programa y facilidad para depurarlo, al tiempo de dar acceso a muchos datos desde distintas rutinas. Esto puede generar muchos efectos colaterales, si no se tiene cuidado al programar.

Actualmente no se consigen muchas utilerías para trabajar con ADA en computadoras personales, ni siquiera existen muchos compiladores ADA. Por esta razón, su uso no es frecuente y la documentación es muy reducida.

II.2.6 Lenguaje C

C es un lenguaje que ha tenido una gran aceptación y que continúa haciéndose más popular. Fue creado por Kernighan y Ritchie, para escribir originalmente UNIX[3], un sistema operativo ampliamente utilizado que ha marcado pautas para varios sistemas operativos. Actualmente, C se ha usado para proyectos de mucha importancia, como sistemas operativos, animaciones para películas de ficción y programas muy sofisticados, tanto en microcomputadoras como en computadoras grandes. Esto debido a que C proporciona un código extremadamente rápido y compacto. Este lenguaje es capaz de aprovechar las ventajas de la computadora en la que se encuentra, por manejar instrucciones de bajo nivel, ser claro, eficiente y productivo, ya que maneja varias estructuras de control y de datos. Entre otras funciones, permite una gran portabilidad y tiene un pequeño kernel de 30 palabras reservadas. Además, sus rutinas de entrada y salida están en una biblioteca que es fácil de hacer. En C es sencillo realizar módulos independientes de cualquier programa.

C cuenta con una variedad de operadores que se ajustan a las ventajas de cada máquina. Aunque en un principio la gran cantidad de operadores que maneja, lo hacen

---

3 Desarrollado por Laboratorios Bell.

difícil de aprender,como por ejemplo, "++" y "--" operadores de incremento, decremento respectivamente. Permiten aumentar la versatilidad de operaciones, pues muchos microprocesadores no pueden realizar esas operaciones con una sola instrucción de máquina.

Otra ventaja de C 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 referenciar una variable a través de su nombre o través de un apuntador. La validación de tipo, sí se realiza, porque un apuntador está definido para un tipo específico de objeto. Además, se pueden realizar operaciones 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. A pesar de ello, C da muchas ventajas en cuanto a facilidad de programación y libertad para usos muy especiales, totalmente prohibidos en muchos lenguajes, tales como acceder a la memoria directamente y no através de variables.

El lenguaje C soporta variables estructuradas y permite convertir variables de cualquier tipo[4] 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. Tiene la facilidad de realizar programas muy eficientes aprovechando las características de cada máquina (en

---

[4] tipo : character – caracter, integer – entero, real – real, etc.

especial al estilo de los lenguajes ensambladores). Los programas escritos en C frecuentemente son muy fáciles de usar en otras computadoras e inclusive bajo otro sistema operativo.

C permite una gran modularidad, pues todo se maneja a través de funciones o subrutinas, pudiéndose 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. Tiene además libertad para manejos de variables, direcciones de memoria, acceso a interrupciones, etc. Por todo ello, se debe tener mucho cuidado al programar y no caer en errores imprevistos.

II.3 Elección de lenguaje

Después de haber efectuado la anterior revisión, se determinó que el lenguaje más adecuado para el desarrollo del sistema es el lenguaje C; porque mezcla poderío, estructura y generalidad en cada una de sus partes.

Dichos puntos son fáciles de observar durante la programación en las declaraciones, tipos, módulos, controles de flujo de programas, compilación, manejos de errores, portabilidad, librerías, funciones, etc. Además de conjugar la cercanía a un lenguaje de bajo nivel sin serlo, junto a su capacidad para aprovechar las ventajas de la computadora donde se encuentra.

II.4 Clasificación de equipos

Al hablar de un equipo de cómputo, la referencia es en general a las computadoras digitales. Ahora bien, una computadora digital es un sistema de procesadores, memorias y dispositivos de entrada/salida interconectados. Representa una máquina que puede resolver problemas al ejecutar un conjunto de instrucciones dadas (programa).

Sus características las han clasificado principalmente en tres rubros: Maxi, mini y microcomputadoras.

II.4.1 Maxicomputadoras

Aproximadamente hasta 1960 las computadoras eran grandes. Ocupaban cientos de metros cuadrados, disipaban suficiente energía como para calentar un edificio completo, costaban una pequeña fortuna y tenían un equipo de docenas de programadores, operadores y técnicos para atenderlas. Con pequeñas modificaciones, este modelo continúa en la mayoría de los centros de cálculo. Estos monstruos se han hecho llamar maxicomputadoras, macrocomputadoras, maxis o *mainframes*.

II.4.2 Minicomputadoras

Hacia 1961, Digital Equipment Corporation (DEC) introdujo la PDP-1, que ocupaba menos de 10 metros cuadrados, podía calentar a lo sumo su propia habitación, costaba solamente 120,000 dólares y podía ser controlada por dos o tres personas. Era la primera minicomputadora o mini. En 1965 se podían tener minicomputadoras por 20,000 dólares

y tenían enorme demanda. A ese precio, un departamento de una compañía o de una universidad podía comprar su propia computadora y lo hicieron por miles.

II.4.3 Microcomputadora

En 1972 Intel introdujo la primera Unidad Central de Procesamiento (UCP[5]) contenida en una pastilla de circuito integrado, la 4004. Para lo que se acostumbra actualmente, la microcomputadora era una máquina muy primitiva, pero tuvo éxito y en pocos años dio vida a una nueva industria. Debido a que el término "minicomputadora" se utilizaba ya y designaba algo mucho más grande, a esta UCP contenida en una pastilla se le denominó microprocesador, por lo tanto una computadora cuya UCP fuera un microprocesador se le llamó en consecuencia microcomputadora (o micro).

II.5 Comparación de equipo por uso típico y técnico

Al principio, las maxis, minis y micros tenían diferencias técnicas muy claras. Por ejemplo, las maxis tenían tamaños de palabra[6] de al menos 32 bits, las minis de 16 y las micros de 8. Además las maxis podían manejar memorias muy grandes, las minis de tamaño medio, y las micros, unas muy pequeñas.

A raíz de las grandes mejoras en la tecnología de los circuitos integrados, estas diferencias ya no existen hoy en día. Algunas micros tienen longitudes de palabra de 32

---

5 Comunmente llamado "Cerebro" de la computadora.

6 Para almacenar datos en la memoria, realizar operaciones lógicas y aritméticas, y efectuar transferencia de datos a dispositivos de Entrada/Salida se utilizan canales que transmiten cierta cantidad de dígitos binarios a la vez. La longitud de palabra en una computadora es un parámetro que permite determinar su potencial, entre mayor sea esta longitud, se procesará más rápido y habrá posibilidad de manejar mayor memoria.

bits y pueden direccionar tanta memoria como una maxi. Por tanto, no tiene sentido diferenciarlos con base en sus características técnicas.

Lo que es más o menos significativo reside en la distinción hecha por su utilización típica. Una maxicomputadora es una máquina grande que funciona en un centro de cálculo para el beneficio de una gran comunidad de usuarios, que a menudo no tienen relación entre ellos. Cuando se usan en tiempo compartido, están dotadas de muchos millones de bytes de memoria central y miles de millones de bytes de memoria de disco, y pueden atender a unos 100 usuarios en forma interactiva.

Una minicomputadora es una máquina más pequeña operada por un departamento o destinada a un proyecto. Como máquina de tiempo compartido debería tener, a lo sumo, unos cuantos millones de bytes de memoria central, unos cientos de millones de bytes de disco y la capacidad de atender a una docena de terminales. Con la llegada de las "superminis" a finales de los setenta, la distinción entre mini y maxi se ha hecho más imprecisa debido a que estas superminis pueden configurarse para que tengan la potencia de una pequeña maxi.

Por el contrario, una microcomputadora no se concibe como una computadora en sí, sino que suele estar incrustada en una parte de un equipo, como un coche o un televisor. En esta configuración tiene poca memoria y ningún disco. La confusión comienza cuando es posible equipar las microcomputadoras con un millón de bytes de memoria y varios discos, usándolos como minicomputadoras o incluso como pequeñas maxis. Aunque en su forma más básica es utilizada como computadora personal.

La conclusión de esta explicación es que no existe ninguna diferencia conceptual entre micros, minis y maxis. Son simplemente nombres burdos con los que se designan varias partes solapadas de un espectro continuo de potencias de los procesadores. Además, los futuros avances de la tecnología probablemente harán estos nombres algo menos significativo de lo que son hoy.

Sólo cabe mencionar a la demanda en equipos por su propósito general, facilidad en instalación, facilidad en uso, existencia de software suficiente para trabajar, existencia de dispositivos para expansión, sin olvidar el costo del equipo original y el de los incorporados.

II.6 Elección de equipo

Tan importante es escoger el lenguaje que se va a emplear para programar, como la elección del equipo donde se va a desarrollar el programa o sistema. La clasificación y comparación marcó a las microcomputadoras como el equipo viable para trabajar.

Entre los factores importantes están el precio, la disponibilidad en el mercado, la paquetería o software existente para ellas y el propósito general de las mismas.

La confirmación en el uso de las microcomputadoras compatibles con el estándar IBM, llamadas PC[7] (Personal Computer) compatibles, es el incremento en el número de personas que las utilizan en el nivel educativo. Permitiendo la mejor difusión de paquetes o sistemas computacionales educativos.

---

7 Las llamadas computadoras personales compatibles con el estándar IBM, deben su nombre y diseño a la mencionada compañía que las publicitó en agosto de 1981.

A partir de 1989 se han instalado PCs en diversas secundarias del país, se han capacitado maestros y actualmente se pone en marcha un programa experimental de computación en primarias y preprimarias mexicanas. Igualmente, se instalan PCs en Normales Federales para formar maestros de primaria.

En vista de la inusitada difusión de las computadoras personales y de los requerimientos tecnológicos actuales, el gobierno de la República ha destinado recursos para el plan de introducción de las computadoras al sistema educativo nacional.

En relación con la enseñanza básica, la Secretaría de Educación Publica (SEP) ha encargado al Instituto Latinoamericano de la Comunicación Educativa (ILCE) el diseño y puesta en marcha del proyecto que introduce la computación en las escuelas de ese nivel.

Dicho plan, denominado Programa de "Introducción de la Computación Electrónica en la Educación Básica" (COEEBA-SEP), se inició en 1985 y actualmente cuenta con una experiencia importante en todos sus aspectos.

La fase experimental del Programa comenzó en 1986, al introducir la computación como auxiliar didáctico en las áreas de Español, Ciencias Sociales, Ciencias Naturales y Matemáticas del tercer grado de secundaria.

En el primer semestre de 1989 se inició la fase de generalización, al extender los Programas Educativos de Computadora (PEC) al 2o. y 1er. grados de ese nivel educativo, y en carácter de prueba a la educación primaria, preescolar, especial y de adultos.

Justamente, el Programa COEEBA-SEP pretende establecer y perfeccionar el uso de la microcomputadora como auxiliar didáctico en los niveles de primaria y secundaria -principalmente-, además de establecer programas de estudio y asignaturas para la enseñanza sistemática de la computación desde este último nivel de educación básica.

De esta forma, las múltiples circunstancias nos alentaron de sobremanera para desarrollar nuestro procesador de texto infantil en este tipo de equipo personal.

Capítulo III **Diseño**

Después de seleccionar el lenguaje de programación y equipo, previo análisis y planteamiento del problema, se preparó el camino hacia el diseño; fase donde se analizó detalladamente el conjunto de elementos que conformaría la estructura del procesador.

Los elementos se dividieron en los siguientes puntos:

a) Interfase de usuario.

b) Editor.

c) Diccionario.

d) Impresión y salida.

e) Caracteres especiales del español.

III.1 Interfase de usuario

Este punto hace referencia a la forma de interacción entre el programa y usuario, al manejo particular de cada una de sus partes, así como la presentación y documentación que le acompañan.

En su clasificación se agrupó a la interfaz en tres partes:

a) Pantalla de presentación.

b) Menú general.

c) Edición.

III.1.1 Pantalla de presentación

La primera impresión que recibe el niño al momento de entrar al programa es la pantalla de presentación, razón por la cual debió darse la importancia necesaria y no soslayarse.

En su diseño, la presentación contempla el conjunto de datos que describen al software en cuestión.

Esta primera pantalla muestra en la parte central el nombre del procesador de texto y su versión; la parte inferior cita los nombres de los autores junto al año de terminación del mismo.

En su manipulación, los datos o componentes hacen un movimiento animado. El recuadro central se divide en dos partes simétricas, una zona izquierda y otra derecha; ellos empiezan un movimiento horizontal desde cada uno de sus extremos laterales, hasta llegar a la posición original (centro) para conformar el recuadro inicial. A la par, se realiza un movimiento sincronizado de los nombres de autores. Finalmente, termina la animación con el enmarcado del nombre del procesador y el despliegue del año de terminación del sistema. Su elaboración técnica tiene como base al monitor en modo gráfico.

III.1.2 Menú general

Al invocar el programa lo primero que se muestra es la pantalla de presentación, enseguida toma lugar el Menú general que, de igual manera, es elaborado en modo gráfico. Para llevar a cabo esta tarea fue preciso detectar el tipo de adaptador gráfico de la computadora y seleccionar de entre la gama existente el modo adecuado para la pantalla de trabajo, tomando en consideración la resolución y paleta de colores (véase tabla III.1).

La siguiente tabla muestra dicha selección:

Tabla III.1 Selección de adaptador gráfico

Driver gráfico	Modo gráfico	Columnas x renglones	Paleta de colores	Selección
CGA	CGAC0 CGAC1 CGAC2 CGAC3 CGAHI	320 X 200 320 X 200 320 X 200 320 X 200 640 X 200	C0 C1 C2 C3 2 colores	selección
MCGA	MCGAC0 MCGAC1 MCGAC2 MCGAC3 MCGAMED MCGAHI	320 X 200 320 X 200 320 X 200 320 X 200 640 X 200 640 X 480	C0 C1 C2 C3 2 colores 2 colores	selección
EGA	EGALO EGAHI	640 X 200 640 X 350	16 colores 16 colores	selección
EGA64	EGA64LO EGA64HI	640 X 200 640 X 350	16 colores 4 colores	selección
ATT400	ATT400C0 ATT400C1 ATT400C2 ATT400C3 ATT400MED ATT400HI	320 X 200 320 X 200 320 X 200 320 X 200 640 X 200 640 X 200	C0 C1 C2 C3 2 colores 2 colores	selección
VGA	VGALO VGAMED VGAHI	640 X 200 640 X 350 640 X 480	16 colores 16 colores 16 colores	selección

Al momento de hacer la selección se pretendió estandarizar auna sola resolución para evitar conflictos con el número de pixeles (puntos) por renglón y columna, cuyas variedades repercuten en el tamaño de figuras, letras y recuadros que se muestren en pantalla. A este problema se adicionó la paleta de colores, donde la cantidad de colores

a disposición en cada resolución no coincide en todos los tipos de monitores, por ello fué necesario elegir más de una opción.

Al observar estas desavenencias se seleccionaron dos estandares:

	RESOLUCION	PALETA DE COLORES
Estándar 1	320 x 200	16 colores
Estándar 2	640 x 200	4 colores

Con las características de esta selección se cubren los modos gráficos más comunes en monitores.

Ahora bien, contar con dos resoluciones provoca la creación de un factor de ajuste en el número de pixeles por columna, y la selección de colores cuando la paleta contenga 4 colores o 16 colores. Y de esta manera, poder cumplir en ambos estandares con el mismo color, tamaño y distribución de los componentes a desplegar.

Cuando el procesador de texto detecta el adaptador gráfico, siempre busca usar el estándar que le dé la más alta resolución. El modo gráfico representa la base de trabajo del Menú general, por eso la importancia en su establecimiento. El paso inmediato consistió en conformar su estructura con base en figuras. Se optó por la colocación de figuras que hicieran alusión a cada una de las funciones del procesador, y así el niño tuviera (aparte del nombre) otro punto de referencia en la elección de las funciones. Al mismo tiempo, se asignaron a esas opciones las teclas <F1>, <F2>, etc., del teclado para enseñar al niño a asociar cada una de las teclas con la función

respectiva del procesador y familiarizarlo con normas que encontrará más adelante en otras sistemas. El orden de presentación es el siguiente:

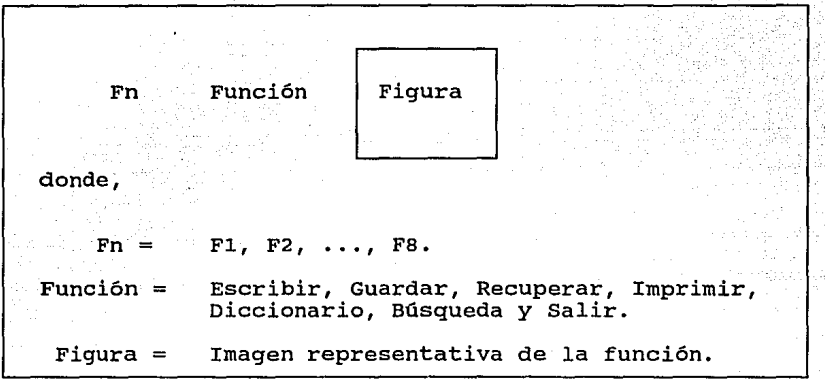


Fig. III.1 Descripción de la función

La distribución de las figuras en la pantalla es simétrica, reservando la parte baja para presentar mensajes relacionados con las funciones. Al mismo tiempo, se ha provisto al Menú con un movimiento para selección de comandos; con la ayuda de las flechas de teclado se realiza el desplazamiento a través de las teclas de funciones, que cambian de color al instante de pasar por cada una de ellas, ofreciendo un indicativo al niño de la Fn que selecciona. Este indicativo se complementa con el mensaje breve que aparece en el recuadro inferior de la pantalla.

El movimiento puede ser hacia la derecha, izquierda, arriba, o abajo según se desee. Una vez en la posición del comando, con sólo presionar la tecla de <Enter> se activa la función. Para el caso, se ha considerado otra alternativa para evitar moverse

hasta la función deseada: con presionar únicamente la Fn del teclado (aún sin estar en ella) se obtiene el mismo resultado. Esta doble alternativa se ha hecho pensando en facilitar y dar opciones a la selección dentro del Menú general.

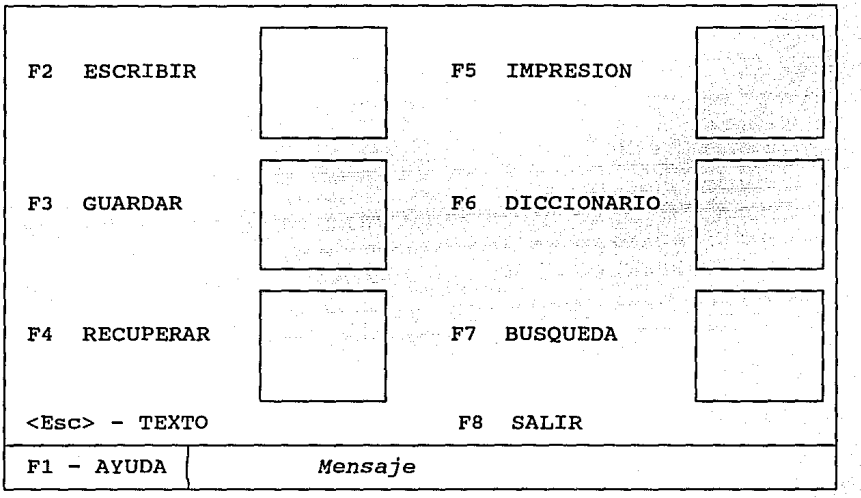


Fig. III.2 Menú general

Como parte adicional pero necesaria, en el menú se colocó una ayuda que explicará más ampliamente la función de los comandos, sin dejar de ser una explicación breve. Esta información que estará contenida en un archivo podrá consultarse cuando se requiera, al presionar la tecla <F1>[8]. Al invocar la ayuda se accesará al archivo, considerando la posición dentro del Menú para presentar de entrada, la explicación del

[8] En varias paqueterías se tiene la convención de utilizar F1 como opción de ayuda.

comando que corresponde a la función donde estamos localizados. No obstante, es posible recorrer todo el archivo de ayuda si así se desea.

La otra característica del Menú general es permitir salir de esta pantalla menú y entrar a la pantalla de edición cuando no se quiera invocar alguna función, sino sólo consultar la ayuda o verificar algunos detalles del menú. El cambio o regreso al texto para continuar o empezar con la edición del documento de trabajo, se lleva a cabo con la tecla de <Esc>[9].

Después de la estructuración y diseño de las funcionalidades del menú, resta ahora el análisis de figuras. En primer orden, hay que determinar el tamaño de las figuras, en segundo, saber cuáles son las figuras adecuadas y, tercero, indagar cómo elaborarlas y guardarlas para su despliegue en pantalla.

Para determinar el tamaño de las figuras, se realizaron pruebas en los dos estándares gráficos, tomando en cuenta el número de pixeles por pantalla. Se programaron tamaños hasta observar en ambas pantallas una acertada distribución y apariencia, con el objeto de evitar repetir o ajustar las figuras en los dos estándares. Después de algunas pruebas se decidió por un tamaño final de 50 x 50 pixeles.

En la creación y elección de figuras, se debe idear una representación gráfica que describa de la mejor manera al commando. En el caso de un niño debe cuidarse además de esta descripción certera, la familiaridad que pueda tener con la imagen o en su defecto la relación real con la función. De esta manera, a partir del análisis que entrañó se llegó a la siguiente elección de figuras:

---

9 Tecla utilizada por varias paqueterías para salir de un menu.

FUNCION		FIGURA
Escribir un documento	.-	Un lápiz.
Guardar un documento	.-	Un archivero con uno de sus cajones abierto, mostrando sus documentos guardados y ordenados.
Recuperar un documento	.-	Un archivero con uno de sus cajones abierto, mostrando sus documentos a punto de salir.
Imprimir un documento	.-	Una impresora con papel.
Consultar diccionario	.-	Un libro abierto.
Buscar palabras en el texto.	.-	Una lupa.

Para elaborar las figuras anteriores fue preciso empezar con un boceto en papel que se ajustara al tamaño de 50x50 puntos. Al observar la dificultad para ajustarlo en papel, se auxilió del paquete Story Board para crear una plantilla de 50x50 cuadros e iniciar la minuciosa labor de dibujo, cuadro por cuadro. Al cabo de ello, se contaba con el conjunto de pixeles a encender en cada recuadro del Menú General y así conformar las imagenes respectivas. Sólo restaba, como último paso, buscar un formato de código para guardar las imágenes, tomando en cuenta los pixeles encendidos, trazos, repeticiones, bytes necesitados para almacenar la figura y la cantidad de programación requerida para desplegar las imágenes a través del código. Las convenciones desprendidas de los puntos anteriores para guardar y leer el contenido de las imagenes, son:

a) Guardar

- Sólo pixeles encendidos.
- Información por renglones, indicando el número de la columna con pixel encendido.

b) Claves de código

[0]	-	Separador de renglones.
[1]	-	Separador de figuras
[2,x1,x2]	-	Una línea entre dos puntos (columnas x1-x2).
[3,n,...]	-	Repetir n veces el renglón.
[49]	-	Repetir código en la figura siguiente.
[50]	-	Fin de figuras.
5,6,...48	-	Número de pixel(columna) a encender.

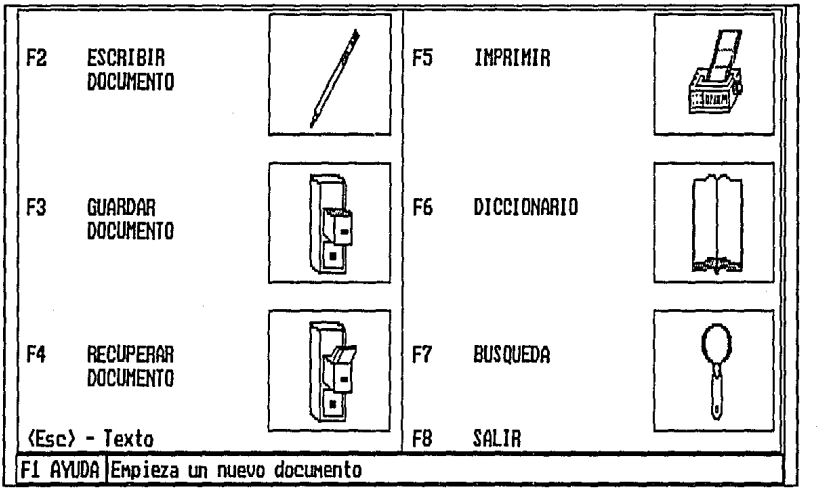


Fig. III.3 Menú Principal

III.1.3 Edición

Algunos de los elementos que juegan un papel muy importante y muchas veces determinan la aceptación de un producto, son la presentación y comunicación con el usuario. Motivo por el cual se han contemplado las siguientes cualidades dentro de la edición:

Responsividad	.-	Las acciones del usuario producen resultados directos y se podrán ver reflejados de inmediato.
Flexibilidad	.-	Las aplicaciones permiten al usuario elegir cualquier función bajo su responsabilidad y necesidad.
Consistencia	.-	Las teclas que se usan dentro de las diversas funciones tienen gran similitud o en su caso la misma.
Manejo de errores	.-	En posibles situaciones de error, mandar a manera de mensajes una pista o guía de lo ocurrido, que permitan proceder al respecto.

III.2 Editor

Una de las funciones del procesador sobre la que recae gran peso, es sin duda la edición. Es el marco principal sobre la que se accesa al conjunto de funciones y comandos del sistema. Es la pantalla donde se permanece la mayor parte del tiempo durante la elaboración de un documento.

En el diseño del editor se tomaron en cuenta ciertas características de otros sistemas y se consideraron comentarios de expertos en la materia. Con la información adquirida se aportaron ciertas características a los elementos que lo componen:

-- Formato de la Pantalla.
-- Insertar texto.
-- Sobre-escribir texto.
-- Borrar texto.
-- Movimiento del cursor.
-- Grabar texto.
-- Recuperar texto.
-- Búsqueda.

III.2.1 Formato de la Pantalla

En la pantalla de edición se optó por trabajar en modo texto, con una resolución de 40x25. Esta resolución dispone de una gama estádar de colores y de una letra de mayor tamaño.

El uso de letra grande es para llamar la atención del niño y facilitar la visualización de la edición. En pedagogía suele utilizarse durante el proceso de enseñanza-aprendizaje, y al equipararse con el procesador es observable el mismo proceso, aunque en forma indirecta.

Además, la resolución de 40x25 ofrece el mismo estandar de colores en cada tipo de monitor, sin tener el problema de adaptadores gráficos (VGA, CGA, etc) como sucede en modo gráfico. Resta detectar si la computadora cuenta con monitor monocromático o a color.

Dentro de las características anteriores se empezó a diseñar lo que sería el formato de pantalla. Primero, enmarcando el área de trabajo con una ventana del tamaño de la pantalla para dar seguridad al niño en la edición. El área de trabajo dentro de la ventana mantiene una dimensión de 22 líneas de largo y 36 columnas de ancho por pantalla, dichos parámetros se presentan junto al número de página y recuadro de mensajes, en el renglón de estatus, línea 23. Esta última línea, indica la posición del cursor, al llevar un control interno del número de línea, columna y página que se edite en todo instante. Esto se aprecia en la figura siguiente.

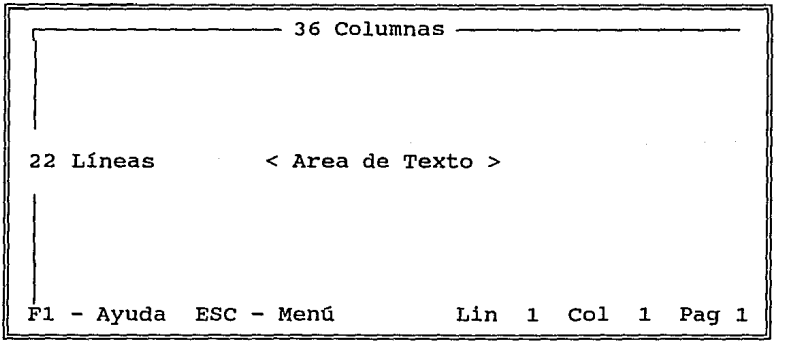


Fig. III.4 Formato de pantalla

III.2.2 Insertar Texto

Insertar quiere decir adicionar texto a un documento, ya sea éste nuevo o elaborado con anterioridad. Bastará con encontrarse en la pantalla de edición para poder escribir.

Al momento de entrar al procesador se ha dispuesto activar el modo de inserción y mantenerlo hasta que se decida sobre-escribir.

Dentro de sus características se ha considerado actualizar al texto en todo momento; cada vez que se teclee un carácter se moverán las palabras, líneas y oraciones pertinentes para tener al texto actualizado. Ello, con el fin de no confundir al niño en otras circunstancias, como sucede en procesadores que utilizan el movimiento de cursor para actualizar la información, conservando una línea demasiado larga hasta que se utilice éste recurso auxiliar o se rebase su límite.

Conforme se inserta texto sobre la línea, se checa su longitud. Si logra rebasar su límite en tamaño, se baja la última palabra de la línea a la próxima siguiente.

Internamente se obtiene la longitud de la palabra, se guarda en un buffer, se elimina de la línea actual y se adiciona en la siguiente. Pero, antes de bajar la palabra, se calcula el espacio existente en el renglón inmediato. Si satura su longitud con la palabra del buffer, el proceso se repite en forma recursiva hasta llegar a tener actualizada la información.

Además de la implantación del sistema anterior, se ha diseñado un algoritmo exclusivo para el manejo de las que hemos dado en llamar "palabras grandes". La finalidad es el control de texto en sus diversas modalidades. Conviene considerar al procesador cuando recibe cadenas de caracteres sin espacios o blancos, que llegan a

ocupar más de una línea de texto. Para indicar tal circunstancia, en pantalla se coloca un carácter (▸) para especificar al final de la línea la continuación de una cadena que rebasa el límite y que se extiende a la línea próxima siguiente, sin llegar a tener un límite en número de líneas. Dentro de estas cadenas continuas es posible insertar blancos, returns, etc., y dividir dichas líneas si así se desea, sin lograr crear algún problema.

III.2.3 Sobre-escribir Texto

La función sobre-escribir permite, al igual que la inserción, adicionar texto a un documento, la diferencia estriba en la forma de hacerlo. En modo de inserción, cuando se escribe un carácter, se conserva la información precedente al recorrer el texto a la derecha del cursor, mientras que en modo de sobre-escritura al incorporar el texto nuevo se sustituye el ya existente. Esta opción puede ser útil cuando se requiera eliminar y remplazar alguna parte del texto.

Para activar el modo de sobre-escritura, se presiona la tecla de <Insert>. Cuando se activa, aparece en la pantalla de edición un indicativo del modo de trabajo.

III.2.4 Borrar Texto

Borrar es la función que elimina texto no deseado. Su combinación con la inserción, permite modificar el contenido de un documento.

Las dos modalidades diseñadas para borrar texto consideran dos teclas: la tecla de <Backspace> y la tecla de <Suprime> o <Delete>. Por lo general, dichas teclas

son seleccionadas para llevar la tarea de borrado dentro de las diversas paqueterías del mercado, de ahí su designación en el procesador.

La tecla <backspace> borra un carácter a la izquierda del cursor, mientras que <Suprime> borra el carácter que se encuentra en la posición del cursor.

Varias de las características de inserción se implementaron para el borrado de texto, como acontece en la actualización de texto cada vez que se borra un carácter, y el manejo de cadenas de caracteres que rebasan el tamaño de una línea.

III.2.5 Movimiento del cursor

Esta característica permite recorrer todo el documento de trabajo, con movimientos entre líneas, columnas y páginas.

En el diseño se eligieron los movimientos más comunes del cursor para dar un empleo real a cada uno. Estos tipos de movimiento, permitirán dentro del procesador, desplazar el cursor a cualquier parte del documento, dependerá del lugar al que se desee posicionar el cursor para escoger la opción más conveniente.

Entre las teclas seleccionadas existe una correspondencia de función realizada y nombre o figura de la tecla.

La tabla siguiente muestra las teclas que dan movimiento al cursor junto a la función asociada:

Tecla(s)	Función o Movimiento
< ↑ >	Una línea hacia arriba.
< ↓ >	Una línea hacia abajo.
< → >	Un carácter a la derecha.

< ← >	Un carácter a la izquierda.
< Home > o < Inicio >	Movimiento al inicio de la línea.
< End > o < Fin >	Movimiento al final de la línea.
<Pg Dn> <Pg Up>	Avance de una página hacia abajo. Avance de una página hacia arriba.
<Control> <Home> <Control> <End>	Movimiento al inicio del documento. Movimiento al final del documento.

III.2.6 Grabar texto

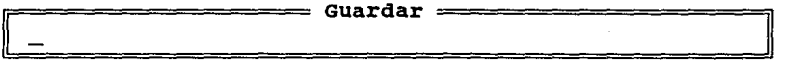
Grabar texto es una función que permitirá guardar el documento editado en el procesador y almacenarlo en disco para un uso posterior.

El medio de almacenamiento secundario (disco), mantendrá en un archivo el contenido del documento. Esto con la idea de guardar la información por largo tiempo, de tal forma que sea copiada a la memoria primaria sólo cuando se requiera esa información.

El tipo de archivo utilizado para grabar texto, es de acceso secuencial, a consecuencia de la clase de uso. Cada vez que se requiera guardar un documento, se vaciará en forma secuencial y total, cada una de las líneas de la memoria primaria. De la misma forma sucederá cuando se accese el archivo para leerlo y recuperarlo en pantalla, tendrá que recorrerse cada registro del archivo para depositarlo en la memoria del procesador.

Como medida de seguridad los archivos que se guarden incluirán como primer línea, una clave del procesador que indique su pertenencia al mismo.

Esta función se llama con la tecla <F3>. Al momento de activarse se despliega en la parte inferior de la pantalla un recuadro (véase fig. III.5) con el título de Guardar. Dentro de él se escribirá el nombre del archivo a guardar en disco. Dicho nombre identificará al documento de trabajo que se edita en ese momento.



Esc - Cancelar

Fig. III.5 Guardar Documento

Al momento de escribir el nombre del archivo, es posible especificar también el nombre del drive y directorio donde se desea guardar el documento. En caso de escribir sólo el nombre, se toma por omisión el directorio y drive que dio acceso al procesador de texto.

Ejemplos:

Si invocamos al procesador desde C:\PROCESADOR\ , y el nombre escrito en el recuadro es,

i) C:\DOCUMENTOS\TAREA.DAT

la función guarda el archivo TAREA.DAT en el disco duro (letra C:), subdirectorio DOCUMENTOS.

ii) TAREA.DAT

la función guarda en el disco duro (letra C:), subdirectorio PROCESADOR, el archivo TAREA.DAT.

ERRORES

Como complemento en el diseño de la función guardar, se ha creado un algoritmo que se encarga de detectar los errores en la especificación de nombre y grabación directa a disco.

El algoritmo reglamenta las siguientes situaciones:

a) Cuando el Nombre del archivo resulta equívoco.

b) Si el Directorio y/o drive no se encuentra en el disco.

c) Si el Drive especificado no existe.

d) Cuando el drive no tiene disco.

e) Si el disco se encuentra lleno.

f) Si el disco está protegido contra grabación.

Estas situaciones se engloban en 4 mensajes de error:

1) Drive o nombre de archivo erróneo.

2) Trayectoria mal especificada.

3) Disco lleno.

4) Drive sin disco.

Aunque estos errores se detectan por medio del sistema operativo, existe el inconveniente de despliegue en pantalla. Esta acción descompone la ventana de edición, al colocar el mensaje de error en la posición donde se encontrase el cursor. Esto sucede al encimar el mensaje de error en la información, y prescindir del patrón de colores en curso. A ello podemos adicionar lo incomprensible que puede resultar el mensaje, al mandar el sistema operativo avisos como: *Error número x*, que se debe consultar en el manual del sistema. De ello se desprende la importancia de contar con un algoritmo de errores.

Si durante el proceso de escritura del nombre, deseamos interrumpirlo, es posible cancelar la función a través de la tecla <Esc> y continuar con la edición o selección de otras opciones.

III.2.7 Recuperar un Documento

Esta función fue elaborada para recuperar de disco los archivos previamente grabados en el medio magnético.

Para tener acceso a esta opción es preciso presionar la tecla F4. Ella mostrará un recuadro con el título de **Recuperar** (véase fig. III.6). En su interior se esperará el nombre del archivo que se desea recuperar, junto a la inclusión de drive y directorio, si así se requiere.

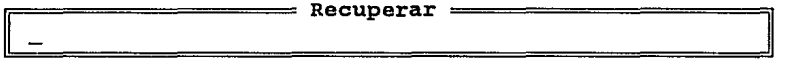


Fig. III.6 Recuperar documento

En el caso de accionar la función por error o al desear abandonar la opción de recuperación de un archivo, es posible cancelarla con la tecla de <Esc>.

ERRORES

De la misma forma que en la grabación de un documento se cuenta con un algoritmo para la detección de errores que contempla las siguientes circunstancias:

a) Cuando el nombre del archivo no existe.

b) Si el archivo está protegido contra lectura.

c) Cuando se especifica erróneamente el drive y/o directorio.

d) Si el archivo no pertenece al procesador de texto.

e) Si el drive se encuentra sin disco.

Dichas circunstancias se engloban en 4 mensajes de error:

1) Archivo no existente o protegido.

2) Trayectoria mal especificada.

3) Archivo no perteneciente al procesador.

4) Drive sin disco.

Si no existe error alguno se procede a la lectura secuencial del archivo vaciando la información en la memoria primaria que controla el procesador. En lo concerniente a los caracteres de control que utiliza el sistema, como parte de la estructura del archivo (returns y separadores de líneas), estos se convierten a un carácter que puede manipularse acertadamente con el software desarrollado.

Si por algún motivo el documento no cabe en la memoria de la máquina, se manda un mensaje de error al usuario. Esta situación se considera poco probable, al calcular el tamaño de archivos que pudiera manejar el niño.

Cuando se recupera un archivo en pantalla se utiliza un trozo de la memoria de la máquina o buffer para almacenarlo y editarlo rápidamente. Cada vez que se recupera un archivo se elimina la información que estuviese en memoria para dar cabida al nuevo archivo y asignar la memoria en forma dinámica.

III.2.8 Búsqueda

El propósito de la función es dar al niño el elemento eficaz para localizar palabras o cadenas de caracteres dentro del texto que escribe, cuando tiene la necesidad de encontrar frases, signos, números o caracteres específicos. Además de poder aplicarlo en la verificación de frecuencia de palabras.

En su estructura se contempla la búsqueda de caracteres hacia atrás o adelante del texto, tomando como referencia la posición del cursor.

Para dar acceso a la función se eligió la tecla <F7> de teclado. Al seleccionarla deberá exhibir en la parte inferior izquierda de la pantalla el título "Se buscará:", que espera la cadena de caracteres a examinar en el documento (véase figura III.7). El tamaño de la cadena no deberá exceder a 25 caracteres, considerando que esa longitud es cercana a la longitud máxima de una línea de texto. En cuanto a su contenido, se incluyen espacios en blanco, letras, números y signos que engloban al Código ASCII.

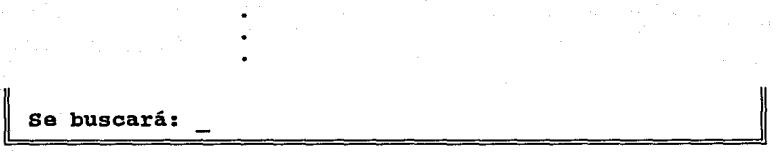


Fig III.7 Búsqueda

En el caso de llamar a la función por error o desear cancelar la busqueda, se debe presionar la tecla de <Esc>.

Si el proceso continúa, se pregunta la dirección de búsqueda (véase fig. III.8). Con dicha opción se pretende agilizar la función, al "dividir" al documento en dos y acceder a elegir alguna de las secciones de interés. Si no se encuentra la cadena, se manda un mensaje y un sonido indicando la situación. En caso contrario se resaltan los caracteres sobre el documento para que el niño pueda identificarlos y al mismo tiempo advertir el contexto sobre el cual se encuentran.

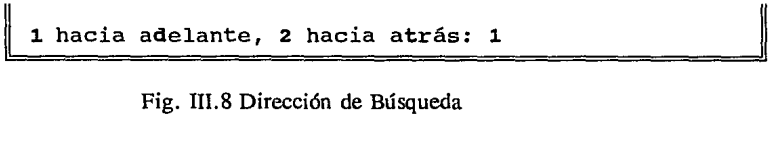


Fig. III.8 Dirección de Búsqueda

Cada vez que se tenga éxito en la búsqueda, se muestra el texto y se pregunta por el deseo de continuar con la misma operación o terminar el proceso.

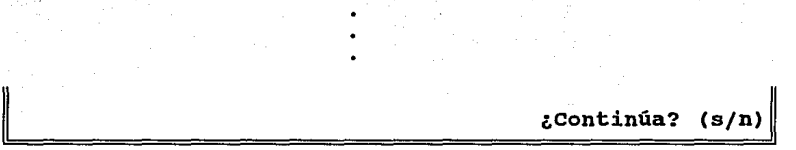


Fig. III.9 Continuación de Búsqueda

Después de finalizar el proceso de búsqueda por decisión propia o por circunstancias de la misma función se regresa a la edición de texto.

III.3 Diccionario

En esta sección del programa se creó una herramienta que apoyará al infante en los momentos de escribir un texto. Está ayuda se le proporcionará en el momento en que se le presente una palabra, y no no le conozca el significado o tenga duda de la ortografía. Con solo presionar una tecla, la función 6 del teclado (**F6**), se le introduce a un Diccionario. La peculiaridad de éste, radica en que es un Diccionario Palabra-Significado en el cual se podrá encontrar un acervo de 6,400 palabras, con la correcta ortografía y significado de cada una de ellas.

La idea de introducir al niño a un diccionario, es porque al mismo tiempo que aprende a usar la computación y las computadoras como herramientas, también se le enseñará un instancia donde pueda satisfacer sus dudas de dicción, ortografía y significado de una palabra. Y en una forma sencilla se le instruye en el manejo de uno. Esto sin perder de vista la importancia que tiene esto en la evolución de su formación y aprendizaje, en esta etapa de su deasarrollo.

Dentro del procesador de texto existen dos formas de poder invocar esta sección, en ambas se realiza la misma tarea. La diferencia que existe es la forma en que se proporcionará la palabra a consultar, como veremos a continuación :

Al iniciar el programa, nos aparecerá un menú en el cual se encontraran varias figuras y entre ellas la de un libro que representa gráficamente la opción de Diccionario, precedida por los caracteres **F6** (véase fig. III.10), que indica que podemos activar esta opción de Diccionario, presionando la tecla <F6>, o colocarse en esta parte, presionando las teclas de movimiento de cursor de hacia arriba o hacia abajo, y luego activando <Return>.

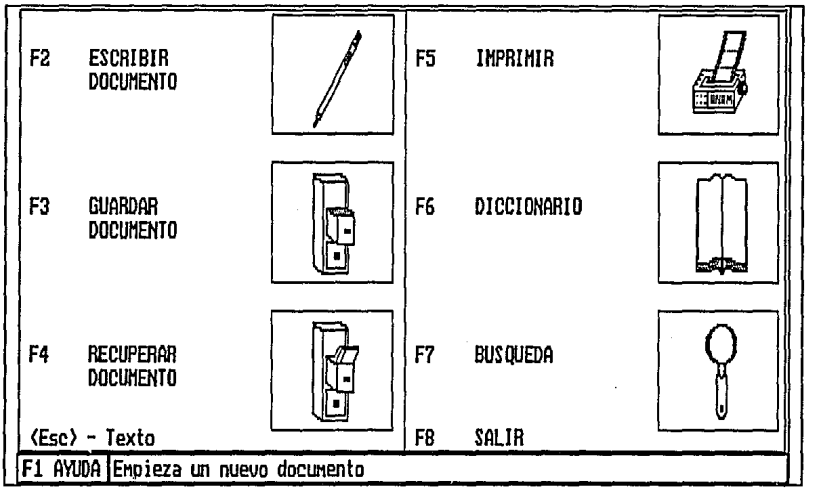


Fig. III.10 Menú principal

Al activar el diccionario, se abrirá una ventana de aproximadamente una tercera perte de la pantalla, ocupando la parte inferior de ésta. Si el diccionario se usa por primera vez en la sesión, aparecerá una ventana en la que indicará la lectura del acervo del diccionario (véase fig. III.11). Posteriormente ya no se realizará.

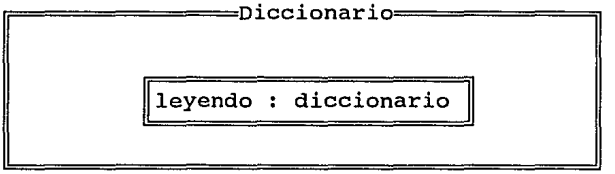


Fig. III.11 Ventana indicando la lectura de acervo del diccionario.

Posteriormente pedirá la palabra a consultar en el acervo, con el siguiente mensaje "Dame la palabra: " (véase fig. III.12), aquí se proporcionará la palabra a examinar.

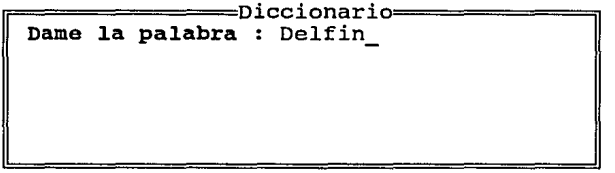


Fig. III.12 Ventana del diccionario llamado desde menú.

La otra forma de activar el diccionario, será dentro del texto. En él se deberá llamar con las teclas de movimiento de cursor (arriba, abajo, izquierda y derecha), colocarse en la palabra a consultar y presionar la tecla de función de Diccionario (F6). Sin importar la posición del cursor sobre la palabra, ya que se cuenta con un selector de palabra que determina sus elementos.

Para indicar que la función está activa, se destaca dentro del texto a la palabra seleccionada. Se abre una ventana de aproximadamente una tercera parte de la pantalla, ocupando la parte inferior de ésta y si el diccionario se usa por primera vez en la sesión, aparecerá una ventana que indicará la lectura del acervo del diccionario (véase fig. III.11). Posteriormente mostrará el siguiente mensaje "Palabra selecionada :" y la palabra a examinar.

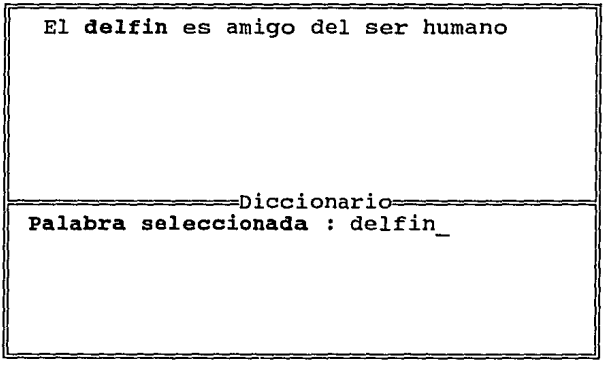


Fig. III.13 Ventana del diccionario llamado desde el texto.

En ambas opciones la palabra no debe contener ningún símbolo o signo que no sea alfanumérico[10] debido a que cuenta con un selector de letras. Por ello todo carácter no alfanumérico, no será tomado en cuenta y se asumirá éste como un espacio en blanco, y con ello el fin de la palabra. Después de obtenida la palabra, se realizará la busqueda en el acervo, para poder mostrar su significado, en caso de existir una mala acentuación en la palabra, se emitirá

10 Alfanuméricos: Todo carácter contenido en el alfabeto español, los dígitos contenidos en la base numérica decimal y signos de acentuación.

un aviso por medio de una ventana de mensaje[11], indicando el tipo de falla de acentuación (véase fig. III.14 y III.15).

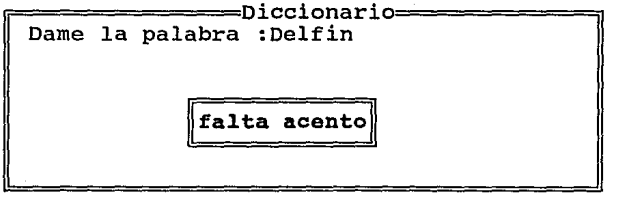


Fig. III.14 Ventana del diccionario llamado desde el texto, mostrando el mensaje de mala acentuación.

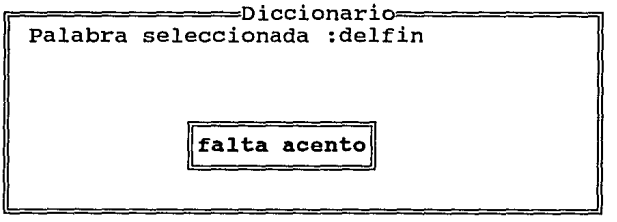


Fig. III.15 Ventana del diccionario llamado desde el menú, mostrando el mensaje de mala acentuación.

Posteriormente desaparecerá la ventana de mensajes y mostrará la palabra correctamente escrita, una línea abajo del término consultado, para que el menor observe su error, y lo asimile.

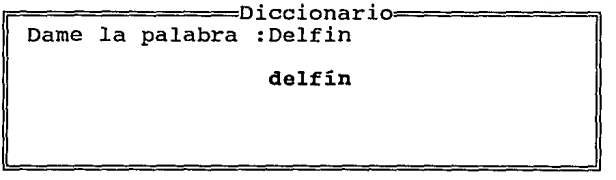


Fig. III.16 Ventana del diccionario llamado desde el menú, mostrando la palabra correcta.

---

11 Ventana de mensaje: Rectángulo de color rojo y fondo blanco con letras rojas.

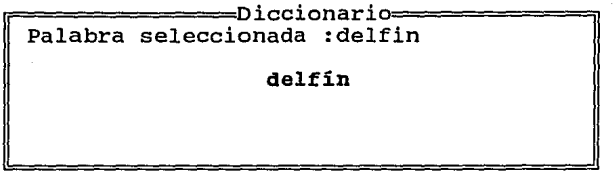


Fig. III.17 Ventana del diccionario llamado desde el texto, mostrando la palabra correcta.

Y unos segundos después mostrará el significado del término deseado, esto permanecerá en la ventana hasta que el menor presione cualquier tecla, como se le indica en la línea de estado

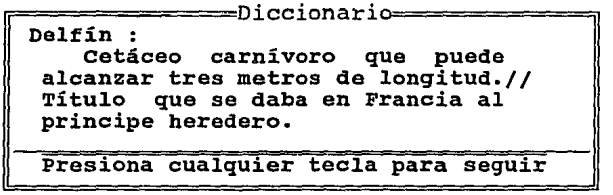


Fig. III.18 Ventana del diccionario mostrando el significado de la palabra.

En caso de que el término a consultar no se encuentre en el acervo, se indicará por medio de una ventana de mensaje, el siguiente aviso : "No se encuentra en el diccionario".

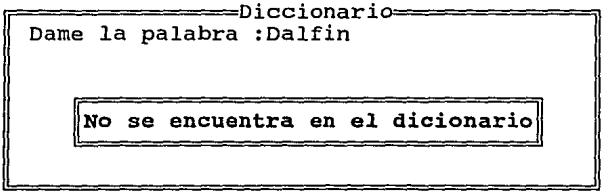


Fig. III.19 Ventana del diccionario llamado desde el menú, mostrando el mensaje de palabra no encontrada.

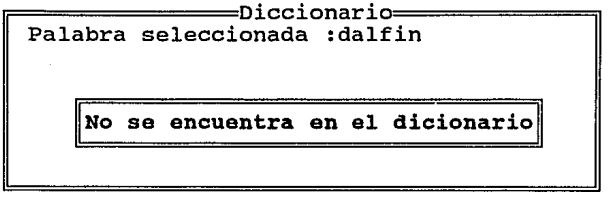


Fig. III.20 Ventana del diccionario llamado desde el texto, mostrando el mensaje de palabra no encontrada.

Después de ello invitará al pequeño a revisar la lista de términos contenida en el diccionario (véase fig. II.21). Esto con el fin de verificar que la palabra proporcionada se encuentra correctamente escrita, y a la vez poder explorar el repertorio del diccionario.

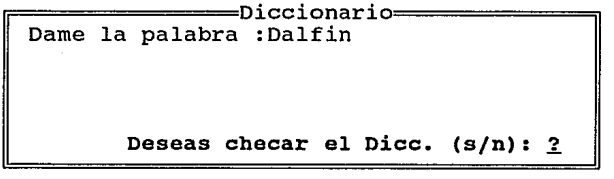


Fig. III.21 Ventana del diccionario cuando se llama desde el menú, se invita a revisar el diccionario.

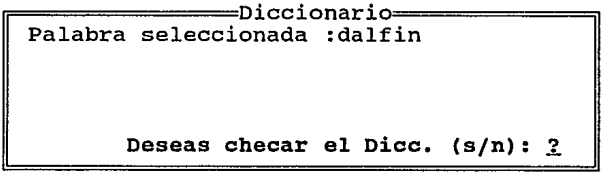


Fig. III.22 Ventana del diccionario cuando se llama desde el texto, se invita a revisar el diccionario.

Si se responde afirmativamente a dicha invitación, pasará a una lista de vocablos ordenados alfabéticamente. Se colocará dentro de ella en la palabra más cercana a la consultada

(véase fig. III.23). Esta opción la podemos aprovechar para ir directamente al inicio de una lista de términos que empiece con una o unas letras en especial.

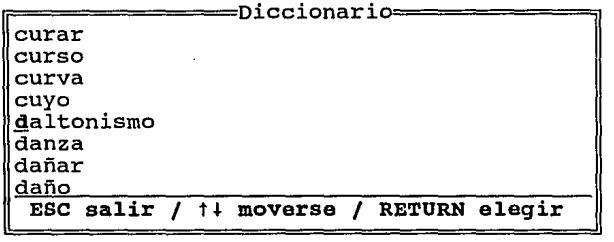


Fig. III.23 Ventana del diccionario en la que se muestra la lista de palabras cercanas a la consultada (que no se encuentra en el acervo).

Dentro de la lista de términos del diccionario, la forma en que la podremos recorrer, será de la siguiente manera :

a) Recorrerlo palabra por palabra

Se hará por medio de las teclas de movimiento de cursor: hacia arriba y hacia abajo.

b) Recorrerlo por páginas de 8 líneas

Se hará por medio de las teclas de movimiento de cursor por páginas: retroceso de página <Re Pág> y avance de página <Av Pág>.

c) Seleccionar una palabra

Colocar el cursor en el término deseado y presionar la tecla de RETURN y con ello se desplegara el significado de la palabra.

Esta última opción se creó con la idea de introducir al niño a emplear un diccionario, y con ello a conocerlo. Si hacemos la comparación de lo explicado en los párrafos anteriores sobre

la forma de manejar el diccionario Palabra-Significado, tema de esta sección y el manejo de un diccionario ordinario, veremos que son semejantes.

III.4 Impresión y salida

La impresión es el proceso terminal en la elaboración de todo documento. Permite obtener constancia escrita sobre papel de la información y disponer del texto fuera del equipo de computo, sin importar el uso o destinatario. En pocas palabras la impresión, representa el producto final del procesador de texto.

Una vez impreso, puede cotejarse el contenido para rectificar, corregir o aumentar texto en el documento. En caso de modificaciones se editará nuevamente hasta dar visto bueno y contar con la versión final del texto para impresión.

En lo que respecta al procesador de palabras propuesto, los puntos que se desarrollaron durante esta fase fueron principalmente dos:

1) La comunicación con la impresora, y

2) El formato de hoja para impresión.

III.4.1 Comunicación con la impresora

La comunicación representa el enlace entre los dos puntos de interés; el procesador de texto y la impresora.

Para llegar a establecer esta comunicación y obtener una salida se debe contar con las dos herramientas adecuadas de hardware y software. En el primero se incluye a la computadora, al cable de interfaz que une a computadora e impresora, y la impresora misma. En el segundo se

incluye al emulador de impresora que hace la transferencia de información hacia el dispositivo de salida.

El punto de interés en el software de comunicación es el emulador de impresora, al constituir la parte directa del procesador que realiza la conección lógica con la impresora.

III.4.1.1 Emulador de impresora

Al emprender la tarea de emulación existía un gran reto para comprender las diferentes clases de impresoras que permitieran llevar a efecto la impresión. Conforme se analizaban, se hallaron ciertas cualidades generales que recomendaron el manejo de dos impresoras: la Epson e IBM. La decisión para utilizar estas impresoras radica en dos razones principalmente. La primera consiste en advertir que la mayoría de las impresoras pueden emular sus particularidades. Y la segunda razón, por conocer de antemano las características demandadas por el procesador propuesto.[12]

El emulador de impresora está constituido por códigos de control y secuencias de escape que se mandan a la impresora a manera de mensajes especiales. Tales mensajes cambian la forma en que la máquina imprime y permite controlar la presentación del trabajo.

La impresora recibe los códigos de control como un único carácter de información. Los códigos mueven la cabeza de impresión en distintas direcciones y cambian la calidad del tipo que produce la impresora.

---

12 Conviene recordar que las impresoras Epson e IBM se distinguen dentro del mercado por marcar un estándar.

Las secuencias de escape constituyen dos o más caracteres de información. Al igual que los códigos de control, permiten modificar la forma en que la máquina está imprimiendo. Por lo tanto, el control que se ejerce sobre la presentación del trabajo es mayor.

El primer carácter de una secuencia de escape siempre es el carácter ESC. Dicho carácter indica a la impresora que la siguiente información es información de control y que no debe imprimirla. Tanto los códigos de control como las secuencias de escape se pueden ubicar en cualquier lugar del flujo de datos que se envía a la impresora.

Los códigos y secuencias de las impresoras Epson e IBM que se emplearon como parte del emulador se muestran en la tabla III.2.

Al encontrar semejanza entre los códigos Epson e IBM, se eligieron sus similitudes para poder crear exclusivamente un emulador de impresora. Estas secuencias seleccionadas permitieron englobar a otras impresoras que manejan un formato estándar.

Dentro de las opciones elegidas se observó la existencia de algunos códigos o secuencias de escape que se desactivaban en el momento de activar a otras, por lo que se experimentó con todo el conjunto de códigos para determinar su comportamiento. Una vez concluido, en su conformación se tuvo el cuidado suficiente para accionar nuevamente estos códigos y aprovechar sus características inherentes en caso de convenir al proceso. Sin embargo, como consecuencia de la construcción de un formato cercano al estándar, se pasaron por alto a las secuencias de escape más avanzadas para dar lugar a códigos más generales, y no por ello menos eficientes.

En la etapa final de esta fase se acordó enviar la información y códigos especiales a través del puerto paralelo. Esto por representar a la vía de salida con mayor empleo en la

impresión y ser el puerto de acceso principal entre computadora e impresor. Por esta razón al puerto estándar de impresión se le designa con las siglas LPT (Line PrinTer) o PRN.

Tabla III.2 Códigos de control y secuencias de escape

CODIGOS DE CONTROL Y SECUENCIAS DE ESCAPE		
	EPSON	IBM
Carácter Ancho	ESC 14	14, ESC 87 1
Fin de Carácter Ancho	20	20, ESC 87 0
Negritas	ESC 71	ESC 71, ESC 69
Fin de Negritas	ESC 72	ESC 72, ESC 70
Avance de Página	12	12
Avance de Línea	10	10
Tabulador Horizontal	9	9
Longitud de Papel	ESC 67 n n = núm. de renglones	ESC 67 n

III.4.2 Formato de impresión

Para dar presentación a un documento es preciso establecer sus características de impresión. En nuestro caso se reglamentaron los margenes, números de página, el espaciado, la justificación, letra de calidad y tipo de letra. Dichos puntos se dividieron de la siguiente manera:

- Tipo de papel y letra.
- Márgenes.
- Impresión de número de página.

- Justificación a la derecha.
- Letra de calidad.

III.4.2.1 Tipo de papel y letra

Lo primero que se debe tomar en cuenta para el formato de impresión es sin lugar a dudas la selección de tipo de papel y letra.

Para continuar dentro de los lineamientos establecidos, el tipo de papel seleccionado fué el tamaño "carta" (estándar). Las dimensiones de este papel son, 8.5 pulgadas de ancho por 11 pulgadas de largo (8.5" x 11"), permitiendo escribir 85 caracteres a lo ancho y 66 renglones a lo largo (85 x 66) cuando el tamaño de letra es 10 caracteres por pulgada (10 cpp).

Al elegir el tamaño de letra, se consideró la dimensión de la letra en el monitor de la computadora cuando se trabaja en el procesador de texto, la cual es más grande a lo normal. Por tal motivo, se eliminó a la letra de 10 cpp y se optó por un tamaño de 8 cpp, que permite escribir 42 caracteres a lo ancho y 66 renglones a lo largo. De esta manera, el carácter de 8 cpp muestra un mayor volumen en el ancho y no en lo largo.

III.4.2.2 Márgenes

Una vez establecido el tipo de papel y letra a usar, es posible fijar los márgenes de la hoja. Esto se contempla a los cuatro lados del papel de impresión:

- Margen superior : 4 renglones
- Margen inferior : 4 renglones
- Margen izquierdo : 3 caracteres
- Margen derecho : 3 caracteres

Para asignar los márgenes anteriores en la hoja, se envía al inicio una secuencia de control que enmarca el tamaño de la hoja (ESC,67,66). Enseguida un código de avance vertical (10) y la rutina de número de página, para que en conjunto muevan al papel y delimiten el margen superior. Mientras que el margen inferior queda predispuesto con el previo margen superior y el control en el número de renglones por hoja.

Para continuar con la asignación se coloca el margen izquierdo, al mandar en forma directa tres espacios antes de cada línea de texto. Y en forma semejante que los márgenes superior-inferior; el margen derecho queda prestablecido con el margen izquierdo que le precede y el control del tamaño de la línea.

III.4.2.3 Impresión de número de página

La colocación de número de página se hizo con la intención de mantener un orden en el documento y facilitar el seguimiento del texto.

La posición elegida para colocar este número de página fue la parte superior derecha de la hoja de texto, por considerarse la posición más común y aceptable.

El número de página tiene una relación directa con el margen superior, por lo que su disposición se realiza en forma conjunta. Ello queda manifiesto durante el proceso de impresión:

1) Se indica a la impresora el tamaño de papel;
2) se bajan dos espacios verticales;
3) se desplaza la cabeza de impresión en forma horizontal sin sobrepasar el margen derecho hasta colocar el número de página; y

4) finalmente se da otro espacio vertical para dejar la página lista e imprimir el contenido del documento.

III.4.2.4 Justificación a la derecha

Se entiende como justificación el espaciado que se efectua sobre cada línea de texto, con la intención de dejar a todos los renglones con la misma longitud; para ello se introducen blancos uniformemente entre palabra y palabra. Es decir, el proceso alínea el último carácter de cada renglón con el resto de los demás, formando un margen derecho imaginario. Las únicas líneas que exceptúan dicha justificación, son las terminadas en retorno de carro o <Return>. Ese carácter de control delimita la línea a una longitud específica; como suele suceder con títulos, fines de párrafos, líneas en blanco, etc.

Con base en el desarrollo, la justificación fue el punto que nos llevó más tiempo en diseñar, en lo que respecta al formato de impresión. Se tenía que distribuir simétricamente el número de espacios entre cada palabra de la línea de texto. Los pasos seguidos para su desarrollo grosso modo son los siguientes:

1) Detectar la necesidad de espaciado en la línea.
2) Checar el espacio faltante en la línea.
3) Checar el número de palabras de la línea.
4) Calcular el número de espacios faltantes entre palabras.
5) Guardar las posiciones donde se necesita espaciar.
6) Colocar los espacios sobre la línea, con base en los datos y cálculos anteriores.

III.4.2.5 Letra de calidad

La letra de calidad es una opción que se dará al procesador para imprimir en negritas cada página del documento. La opción en cuestión, obligará a la impresora a pasar dos veces por cada línea de texto, remarcando su contenido para ofrecer una mejor presentación.

III.5 Caracteres especiales del español

Nos referimos como caracteres especiales del español a las letras del castellano que no se contemplan en todos los teclados tipo QWERTY, que se fabrican para el idioma inglés:

Las vocales acentuadas : á, é , í, ó , ú

El signo de interrogación que abre : ¿

El signo de admiración que abre : ¡

La letra eñe : ñ, Ñ

La u con diéresis : ü, Ü

Dichos caracteres no son posibles de escribir directamente a menos que:

a) Se cuente con el teclado específico;

b) se cambie la configuración de teclado desde sistema operativo; o

c) se manden los códigos ascii correspondientes.

De estas opciones la más viable es la primera, por contar con el teclado específico del español. El único problema es la existencia del teclado en español dentro de toda la gama de computadoras del mercado, lo cual prácticamente es imposible. Por otro lado, la segunda opción tiene el gran inconveniente de cambio de configuración y conocimiento del sistema operativo

como antecedente. A consecuencia de ello, debe conocerse la posición de todos los caracteres, ya que las acciones de varias teclas son cambiadas y el carácter sobre la tecla no corresponde con la función. Es decir, cuando se desea escribir algún carácter debe conocerse muy bien la configuración declarada o por ensayo y error encontrar los caracteres buscados; por lo tanto se considera una mala elección. Finalmente para establecer la tercera opción, es necesario conocer los códigos ascii de cada carácter especial del español y aprender a escribir cada uno de ellos con la ayuda de la tecla <Alt>. Para ejecutar esta acción y mandar a pantalla uno de estos caracteres, se presiona la tecla de <Alt> al mismo tiempo que se digitan los números a través del teclado de calculadora (una sección del teclado completo).

Como lo demuestra el análisis anterior, si no se cuenta con un teclado en español se dificulta la escritura del abecedario completo de la lengua. Por lo tanto se ha buscado una solución al respecto; usar la combinación de sólo dos teclas para escribir los caracteres especiales del español, sin importar que se cuente con un teclado en español o no. Por ejemplo, en caso de contar con un teclado en español, podrá escribirse con las teclas normales del dispositivo o con la nueva opción habilitada. Es decir, la función se mantendrá latente en ambas posibilidades.

Las combinaciones seleccionadas para su implementación, se muestran a continuación:

Combinación				Acción
Tecla	+	Tecla	=	carácter
Control	+	a		á
Control	+	e		é
Control	+	i		í
Control	+	o		ó
Control	+	u		ú
Control	+	n		ñ

Control	+	N	Ñ
Alt	+	a	¡
Alt	+	i	¿
Alt	+	u	ü
Alt	+	U	Ü

III.6 Actualización del Diccionario

Al usar el procesador de texto y particularmente el diccionario, se tendrá la necesidad de agregar y/o modificar palabras y/o significados, ello se hará a través de esta opción.

Se desea realizar un módulo que nos permita adicionar una palabra con su respectivo significado.

Se desea realizar un módulo que nos permita modificar el contenido del significado de una palabra.

De ambos módulos arriba mencionados, las palabras y sus significados quedarán grabadas en un nuevo archivo del diccionario, mismo que será llamado: "DICC09.DIC". También el archivo de índices deberá ser modificado de acuerdo a las modificaciones hechas en dichos módulos. Estas modificaciones también serán direccionadas al archivo mencionado.

Al accionar esta subrutina aparecerán dos ventanas. En la primera ventana, que se llamará "AVISO", se desplegará la siguiente leyenda: "Este programa fue realizado para actualizar el diccionario, con nuevas palabras y nuevos significados; para ello se deberán usar letras minúsculas, sin olvidar la acentuación". En la segunda ventana, que se llamará "ACTUALIZA", se realizará todo el trabajo de actualización. Dicho trabajo empieza mostrando una ventana con el mensaje: "Leyendo el diccionario" que indicará el almacenamiento del índice en memoria. Después aparece el mensaje "Dame la palabra:"; una vez proporcionado el término que se va a agregar y/o modificar, el sistema analizará dicho vocablo, checando que la palabra

esté compuesta por caracteres en español y sin ningún tipo de signos ni números; ya que si hubiera algún signo diferente de los caracteres en español, sólo reconocerá estos últimos, cortando dicho término en donde aparecen esos signos. Después cambiará el mensaje "Dame la palabra:" por el mensaje "La palabra:"; agregando el término analizado. Preguntará si el término proporcionado es correcto, mediante el mensaje: "¿Es correcta? (s/n)". Si la respuesta es negativa pedirá de nuevo la palabra a agregar o modificar. Si se responde afirmativamente se verificará la existencia de éste en el acervo; si el término no existe en el diccionario, el sistema desplegará una ventana con el mensaje: "No se encuentra en el diccionario", preguntará enseguida, si se desea adicionar dicho término al acervo, mediante el mensaje "¿Deseas agregarla al diccionario? (s/n)". Si se responde negativamente, limpiará la ventana para que aparezca el mensaje:"¿Alguna otra modificación? (s/n)"; si se contesta que no, se dará por terminada la tarea de actualización y se saldrá del sistema. Si se responde positivamente, regresará al inicio de la tarea, solicitando la nueva palabra. Cuando el sistema pregunte si se desea anexar la palabra al diccionario y se conteste afirmativamente, se mostrará una ventana con la leyenda "RECUERDA: Debes acentuar las palabras correctamente". Enseguida aparecerá la palabra destacada en fondo rojo y la leyenda: "Tendrá como significado:" para que se pueda colocar inmediatamente el significado de la palabra elegida, mismo que puede tener una longitud máxima de doscientos ochenta caracteres. Una vez escrito el significado, el sistema validará dicho significado enviando el siguiente mensaje: "¿Es correcto el significado? (s/n)". Si la respuesta es positiva, actualizará el diccionario, de lo contrario, ignorará la adición. Ahora bien, si el término de interés existiera en el diccionario, aparecerá una ventana mostrando el significado de dicho término y unos segundos después preguntará: "¿Deseas cambiar el

significado? (s/n)". Si se contesta afirmativamente, mostrará la ventana: "RECUERDA: Debes acentuar los términos correctamente"; enseguida aparecerá la palabra destacada en fondo rojo y la leyenda: "Tendrá como significado:" para que se pueda modificar el significado de la palabra elegida, sin olvidar que dicho texto debe tener una longitud máxima de doscientos ochenta caracteres. Una vez escrito el significado, el sistema validará dicho significado enviando el siguiente mensaje: "¿Es correcto el significado? (s/n)". Si la respuesta es positiva, actualizará el diccionario, de lo contrario, ignora la adición. Si cuando preguntó si se deseaba cambiar el significado, se contesta negativamente, aparecerá el mensaje: "¿Alguna otra modificación? (s/n)". Si la respuesta es "s", pedirá el nuevo término a modificar y/o agregar, continuando la tarea de acuerdo con lo descrito anteriormente, de lo contrario se dará por concluida la actualización y se saldrá del sistema.

Capítulo IV **Desarrollo y pruebas**

El desarrollo del presente procesador de texto utiliza a C como lenguaje de programación. Posee, asimismo una estructuración modular, donde cada módulo o sección tiene una labor específica.

Los primeros módulos del programa inicializan variables, reservan una sección de memoria para el documento, checan el tipo de monitor y preparan el teclado para sensar las teclas oprimidas. Enseguida interviene el módulo de control que se encarga de activar o efectuar las propias funciones del procesador, según el grado de dificultad. Algunas de ellas las realiza por cuenta propia, mientras que otras las manda a ejecutar en módulos separados. Toma en cuenta, para ello, la cantidad de código requerido y la estructura de datos seleccionada.

A su vez, cada función administra y comparte varios submódulos, con el fin de establecer en conjunto una programación estructurada que conforme un árbol jerárquico de módulos y funciones.

En el diagrama que se muestra en la figura IV.1, se observa grosso modo el funcionamiento del sistema. Las divisiones enmarcadas representan las secciones desarrolladas:

1.- Inicialización

2.- Presentación

3.- Teclado

4.- Control y Funciones

5.- Inserción

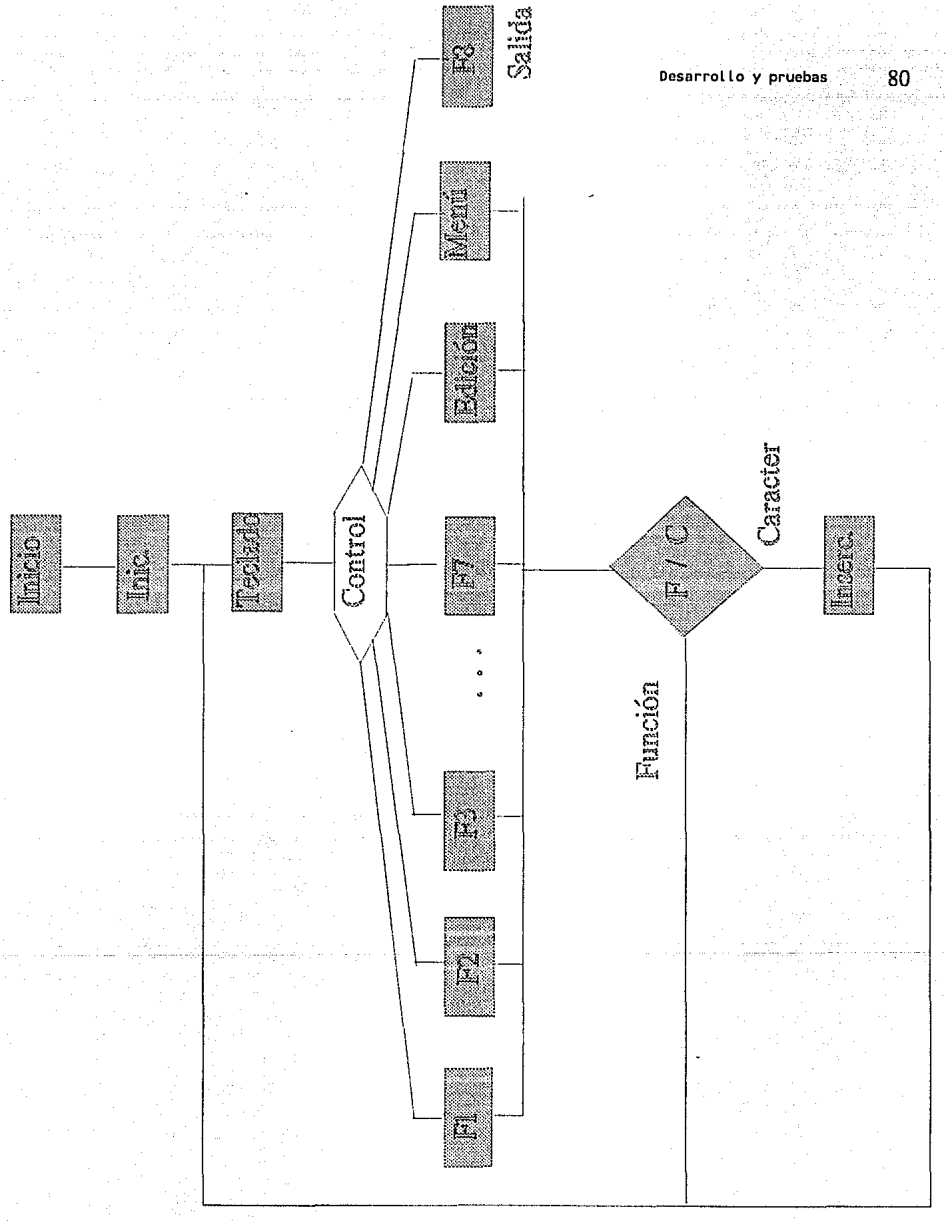


Fig. IV.1 Diagrama de flujo general

IV.1 Inicialización

En la inicialización se considera la declaración de variables, la asignación de memoria y la adición de valores en variables de control. Como segunda parte, se determina el tipo de monitor en la computadora para poner a punto el modo gráfico y el modo texto de trabajo.

Una vez especificadas las clases de variables (constantes, arreglos, estructuras, apuntadores, etc.) con valores iniciales, se aparta una sección de memoria RAM para contener al documento en edición; denominando a esta área *buffer* de trabajo. El buffer crecerá en forma dinámica, acorde al crecimiento de texto en el documento.

Para terminar con la inicialización, se identifica el tipo de monitor de la computadora, sea monocromático o de color.

Los módulos o subrutinas principales que guían la inicialización del procesador son las siguientes:

- *s_init()*
- *texmode()*
- *s_initex()*
- *newlin()*

IV.1.1 Módulo *s_init()*

Este módulo utiliza la interrupción número 11 hexadecimal del Basic Input Output System (BIOS) para obtener información del equipo en que se trabaja. Este módulo determina el tipo de monitor en la computadora, color o monocromático. Su propósito es inicializar una variable de control, con la dirección de segmento de memoria en video.

Este valor permitirá escribir directamente a la dirección de memoria en video, logrando mayor rapidez en el despliege de caracteres a pantalla.

Si el monitor es a color, se escribirá a la memoria de video en la dirección B800:0000; en caso contrario se escribirá en la posición B000:0000 del monitor monocromático.

IV.1.2 Módulo *texmode()*

El módulo textmode() es una librería del lenguaje C de programación, se encarga de activar el modo texto de trabajo. Se aplica en la edición y en el menú principal del procesador.

IV.1.3 Módulo *s_initex()*

Esta subrutina tiene a su cargo la Asignación Dinámica de Memoria (ADM) del buffer de trabajo, la cual contiene el texto en edición.

La sección de memoria RAM (Random Acces Memory) contenedora del documento de trabajo, conforme necesita espacio para alojar al texto, solicita al módulo un aumento de área en memoria.

Esta forma de asignación deja libre el resto de la memoria para ejecutar con mayor fluidez otros procesos. Cada vez que se llama a la subrutina, proporciona cantidades razonables de memoria que evitan desperdicio y lentitud.

Por consiguiente, no hay un espacio de memoria que se preestablezca para el documento desde el inicio del sistema, ni tampoco un tamaño fijo y exclusivo que llegue a desperdiciar espacio o resulte insuficiente para contener el texto. En cambio, la ADM permite un crecimiento acorde a las dimensiones del texto en curso, bajo el límite que marca la memoria RAM de la

computadora. Cabe mencionar que las dimensiones de los archivos que maneje un niño, no serán muy grandes.

IV.1.4 Módulo *newlin()*

Este módulo asigna 40 bytes de memoria para las variables tipo apuntador, en ella se incluyen principalmente a las líneas de texto. Con dicha asignación es posible liberar el espacio en memoria y usar el área en otros procesos.

IV.2 Presentación

La pantalla de presentación muestra en un marco gráfico animado, la versión, el año de exposición, los autores y el nombre del procesador.

En la primer fase, la rutina mgraf() se encarga de detectar y colocar el modo gráfico adecuado para el equipo. Trás examinar el hardware gráfico de la computadora se declara la modalidad apropiada de trabajo, bajo la concordancia en resolución de video y paleta de colores.

En su camino, la subrutina mgraf() analiza el driver gráfico del equipo y elige sus parámetros alternos de acuerdo a la Tabla IV.1.

Tabla IV.1 Modos graficos seleccionados

Driver gráfico	Modo gráfico	Columnas x renglones	Paleta de colores
CGA	CGAC1	320 X 200	C1
MCGA	MCGAC1	320 X 200	C1
EGA	EGALO	640 X 200	16 colores
EGA64	EGA64LO	640 X 200	16 colores
ATT400	ATT400C1	320 X 200	C1
VGA	VGALO	640 X 200	16 colores

Una vez instalado el modelo gráfico en pantalla, se lleva a cabo el movimiento de información con ayuda de las librerías getimage() y putimage(). Ellas efectúan el desplazamiento del recuadro o ventana con el nombre del procesador en pantalla, mientras que la librería outtext() adiciona a ese desplazamiento, un movimiento paralelo con los nombres de autor.

En la implementación de la presentación en pantalla, se usaron principalmente librerías del lenguaje C. El problema central al aplicar sus cualidades, fue el manejo, cálculo y selección de los parámetros; considerando la resolución asociada a cada paleta de colores del monitor.

Desde un perfil externo, un detalle interesante para la economización de memoria RAM en el desarrollo fue la creación de un programa aparte para la presentación, a consecuencia de su manejo y uso. Al analizar su posición, se advirtió que la presentación en pantalla sólo debe aparecer en el procesador al inicio de sesión y jamás volver a reutilizarse, por lo que ameritó su disgregación.

Al manejar un subprograma independiente para la presentación denominado PANIN.EXE, se logró liberar al programa principal de código extra, y a la memoria RAM se le demandó menor cantidad de espacio durante la ejecución.

IV.3 Teclado

La acción que se programó para el teclado, considera el sensado de sus propias teclas. Es decir, obtiene los códigos de cualquier caracter o función en el momento que se presionan sus teclas.

Para tal función se ha dispuesto a la subrutina teclado().

IV.3.1 Módulo *teclado()*

En el sensado de teclas se utiliza la interrupción número 16 hexadecimal del BIOS, la cuál actuá como guardián del teclado.

Aparte de su labor específica, teclado() controla la variable de entrada al menú principal, pasa los parámetros de teclado al área de control y los códigos deciden si es un caracter o función.

IV.4 Control y funciones

Esta sección se puede considerar como la más importante en el programa, lleva el control principal en el procesador y engloba sus funciones.

IV.4.1 Módulo de control

La subrutina o módulo de control porta el mismo nombre dentro del sistema. Se encarga de invocar al conjunto de funciones del procesador, así como la ejecución directa de los procesos de borrado y el movimiento de cursor entre caracteres, líneas, páginas y columnas.

El módulo recibe la posición que tiene el cursor en la pantalla de edición, junto a los códigos de teclado. Se analizan las dos secciones de código, tanto la parte de Scan Code como el código ASCII, para establecer la acción a seguir.

Tabla IV.2 Códigos de teclado y su función

<TECLA> (FUNCION)	SCAN CODE	CODIGO ASCII	<TECLA> (FUNCION)	SCAN CODE	CODIGO ASCCI
<F1>	59	0	<↑>	72	0
<F2>	60	0	<↓>	80	0
<F3>	61	0	<→>	77	0
<F4>	62	0	<←>	75	0
<F5>	63	0			
<F6>	64	0			
<F7>	65	0			
<F8>	66	0			
<Pg Up>	73	0			
<Pg Dn>	81	0			
<Control> <PgUp>	132	0			
<Control> <PgDn>	118	0			
<Home>	71	0			
<End>	79	0			
<Esc>	1	27			
<Return>	28	13			

Aparte de activar las funciones propias del procesador, también ejecuta labores de edición (excepto la inserción), como acontece en los casos siguientes:

- Movimiento de cursor
- Borrado
- Activar Inserción/Sobre-escritura
- Retorno de carro
- Entrada a Menu/Edición

- Caracteres especiales del español.

IV.4.1.1 Movimiento de cursor

Para realizar los movimientos de cursor a través del texto se deben considerar las dimensiones de la pantalla de edición, la longitud de cada línea, el número de líneas que componen al documento mismo, el punto de partida del cursor, tanto en su posición lógica (línea, columna, página) de texto, como en su posición física (x, y) del monitor.

Con los datos anteriores se ejecutan los cálculos correspondientes a la clase de movimiento, auxiliándose de los siguientes módulos:

- *s_colinp()*
- *s_scroll()*
- *s_ps()*
- *s_pl()*
- *s_posyx()*

IV.4.1.1.1 Módulo *s_colinp()*

Después de algún movimiento de cursor, el módulo s_colinp() se encarga de actualizar en el texto, los números de línea, columna y página. Las actualizaciones finales se presentan en una línea de status dentro de la pantalla de edición.

IV.4.1.1.2 Módulo *s_scroll()*

Esta subrutina hace un movimiento de bloques de texto en pantalla o sobre alguna ventana que le sea especificada. A través de ella se efectúan movimientos de información rápidamente, al

realizar las transacciones directamente en la memoria de video, con ayuda de la interrupción 10 hexadecimal del BIOS.

IV.4.1.1.3 Módulo *s_ps()*

Para complementar el proceso de scroll existe el módulo s_ps(), que escribe alguna cadena o string en pantalla. Con él se hace el acceso directo a la memoria de video, al escribir en las direcciones físicas. Ello, con el afán de agilizar la escritura de texto en el monitor.

IV.4.1.1.4 Módulo *s_pl()*

Este módulo es una extensión de s_ps(), su diferencia estriba en el mandato de rangos de líneas y en el chequeo de los límites prestablecidos de pantalla, durante la escritura.

IV.4.1.1.5 Módulo *s_posyx()*

Coloca el cursor en la posición física (x, y) del monitor, con el auxilio de la interrupción 10 hexadecimal del BIOS.

IV.4.1.2 Borrado

Para el proceso de borrado de caracteres se crearon dos alternativas o estándares; una con la tecla *Del*ete o *Supr*ime y otra con la tecla Backspace. El primero borra los caracteres a la izquierda del cursor y el segundo borra el carácter sobre el cursor.

Para ambas rutinas se desarrollo el módulo borcar(), cuya tarea específica es la eliminación de caracteres sobre la línea.

IV.4.1.2.1 Módulo *borcar()*

Esta subrutina es activada desde el módulo de control, recibe la posición de la columna texto, las posiciones físicas del cursor y la opción de Backspace o Delete.

En el proceso interno se inspecciona la línea de trabajo; se calcula su longitud, se detecta la existencia de un Return, algún carácter de fin de línea, un carácter indicativo de línea grande, una línea con espacios en blanco, etc.

De acuerdo con la información que se recibe y la inspección de la línea, se efectúa el movimiento de caracteres en el buffer del documento. La situación de la línea y el carácter que se borra indicarán la necesidad de movimiento sobre la línea, entre líneas o ambos. Efectuando un corrimiento de texto a la izquierda, tomando como referencia al cursor.

Es decir, si existiese el espacio suficiente para subir una palabra de la línea siguiente o el espacio para toda una línea; se efectuaría un corrimiento automático de caracteres y/o palabras de las líneas subsiguientes para mantener la estructura del texto.

La opción Backspace emplea para sus propósitos a borcar() en forma directa, mientras que Delete necesita hacer algunos ajustes después de usarla. La opción requiere hacer consideraciones adicionales en los caracteres de fin de línea, en líneas grandes y Returns.

Dichos ajustes se deben a la forma de borrar; en el Backspace el texto se mueve a la posición del carácter borrado (izquierda del cursor), mientras que en Delete el cursor se mantiene en la misma posición y el texto se mueve a esa posición del cursor.

El módulo borcar() se apoya esencialmente del módulo sube_p() para operar sus procesos.

IV.4.1.2.2 Módulo *sube_p()*

La subrutina sube la palabra o palabras de la línea próxima siguiente a la actual, entendiendo como actual al renglón donde es borrado el carácter.

El módulo se sirve de una programación recursiva que permite ahorrar tiempo en su funcionamiento y líneas de código en su desarrollo. La recursión cumple con su razón de ser al momento de subir de la línea próxima siguiente, la palabra o cadena de palabras y dejar a la misma con espacio suficiente para subir otras palabras. De esta manera, el módulo sube_p() invoca a su misma rutina el número de veces necesarias para ajustar el mínimo de renglones del documento. En sus llamadas, el proceso inicia con diferentes parámetros.

El primer paso determina la posibilidad de subir palabras al renglón actual, observar el espacio existente en ellas, calcular los caracteres que caben en la línea y subir palabras completas, no trozos de ella. En su ejecución se deben también examinar a los caracteres de control como Returns, fines de línea y caracteres de líneas grandes.

En el caso de trabajar con líneas sin espacios entre caracteres que rebasan el límite máximo de la línea (líneas grandes), se usa un módulo adicional que atienda sus características. La rutina en cuestión se llama sube_lg().

IV.4.1.2.2.1 Módulo *sube_lg()*

La subrutina se encarga de subir a la línea grande los caracteres necesarios para completarla y eliminar los posibles caracteres de control cuando se borra un carácter.

Al igual que sube_p(), emplea una programación recursiva al ajustar el conjunto de líneas grandes. Aunque su caso en especial, puede considerarse de doble recursión, dado que sube_p()

llama originalmente a sube_lg(), y ella puede volver a invocar al módulo original sube_p() o llamarse asi misma. Lo anterior establece una cadena o círculo recursivo, que finaliza cuando es imposible mover caracteres o palabras hacia la izquierda de la línea o hacia arriba de las líneas. En ese momento, el ajuste pudo incluir una sección de texto o en caso extraordinario todo el documento.

IV.4.1.3 Activar Inserción/Sobre-escritura

La sección de control, utiliza una bandera para activar el proceso de inserción o sobre-escritura. Por omisión, el procesador activa la inserción en el momento de entrar al sistema.

La tecla de Insert actúa a manera de interruptor que permite seleccionar a alguna de las dos opciones; al activar la opción de sobre-escritura, la parte inferior izquierda de la pantalla muestra un mensaje indicativo de la elección, en caso contrario la zona permanece sin alteración.

IV.4.1.4 Retorno de carro

El retorno de carro sirve para delimitar la longitud de alguna línea; como en títulos, párrafos, oraciones o en líneas vacías que sirven para abrir espacio interlinear.

Al retorno de carro también se le conoce como Return (RTN), Enter o Intro. Para colocar el carácter de control sólo basta presionar la tecla que lleva el mismo nombre.

Los módulos principales que usa en su ejecución, son escrilin(), sube_p() y sube_lg().

IV.4.1.4.1 Módulo *escrilin()*

La rutina escrilin() controla el movimiento de texto cuando se emplea un <Return>.

Mueve la información de la derecha del cursor, coloca un carácter de control e inserta la información en una línea nueva. A la línea nueva se le asigna otra dirección física en memoria, a través de una Asignación Dinámica (ADM) de la memoria-buffer del documento.

Después del módulo escrilin(), se utilizan los módulos sube_p() y sube-lg() si el caso lo requiere.

IV.4.1.5 Menú/edición

Dentro del módulo general de control se permite el acceso al Menú Principal o a la Pantalla de Edición. Como primera opción, el procesador activa al Menú.

IV.4.1.5.1 Pantalla de edición

Para accesar a la pantalla de edición es necesario accesar desde Menú a alguna función del editor o simplemente cambiarse con la tecla de <Escape>. Al instante de entrar en él, se cancela el modo gráfico y se activa el modo texto de trabajo que predispone a la computadora a trabajar en la pantalla de edición.

Las rutinas más importantes en su establecimiento son:

- *closegraph(), texmode()*
- *s_scroll()*
- *s_box()*
- *s_safe(), s_restore()*

IV.4.1.5.1.1 Rutinas *closegraph(), texmode()*

Las rutinas closegraph() y texmode() son librerías del Lenguaje C, que cancelan el modo gráfico y activan el modo texto respectivamente.

En un principio sólo se utilizó la librería texmode() para cambiar a modo texto, sin problema aparente.

Pero durante las pruebas generales del programa se percibió que la memoria RAM se decrementaba injustificablemente. Al seguir la ejecución del programa paso a paso, se encontró que en el cambio de modo gráfico a modo texto se consumía una porción de memoria a consecuencia de las variables de trabajo del modo gráfico; ellas se conservaban en la memoria después de salir de el, duplicando su área cada vez que se entraba al mismo. El problema central residía en la liberación de memoria por parte del procesador que fue resuelta con ayuda de la librería closegraph().

IV.4.1.5.1.2 Módulo *s_box()*

Este módulo dibuja un marco a las ventanas que se realizan en modo texto; con sólo indicar el punto superior izquierdo, inferior derecho, el color y una variable que le da posibilidad a colocar líneas horizontales en la parte superior e inferior de la misma.

Como ejemplo, el módulo enmarca la pantalla de edición al tamaño máximo del monitor.

IV.4.1.5.1.3 Módulo *s_safe()*

Este módulo salva el contenido de una ventana o porción de pantalla con indicar dos puntos, el superior izquierdo (x_1,y_1) y el inferior derecho (x_2,y_2).

Para guardar el contenido de una ventana se calcula el espacio en bytes para respaldarla, se asigna el espacio de memoria en RAM y se accesan las direcciones de video con el objeto de copiar los datos (caracteres) y atributos (colores) de la ventana.

IV.4.1.5.1.4 Módulo *s_restore()*

El módulo s_restore es el complemento del módulo s_safe(); s_safe() salva la ventana en memoria RAM y s_restore() recupera de memoria RAM el contenido de la ventana en pantalla.

El módulo se encarga de colocar la información en la misma posición original, lee las dos posiciones claves de la ventana y translada el contenido de una variable tipo apuntador, a la dirección respectiva de video. Una vez concluido el proceso, se libera la memoria RAM de la copia.

IV.4.1.5.2 Módulo *Menú()*

El módulo se encarga de colocar el menú principal en pantalla (modo gráfico) y conducir la elección de funciones del usuario.

En el primer paso del módulo, se establece el modo gráfico de trabajo con la rutina mgraf(). Al término de su inicialización, se dibuja el menú principal con ayuda del módulo venmen().

El segundo paso controla al movimiento y elección de funciones, con las flechas de teclado; arriba, abajo, derecha e izquierda. En el movimiento con las flechas a través del menú, la rutina efes() resalta la función elegida con un color diferente. Finalmente, la elección se

completa al sensar de teclado algún RTN, F1, F2,...F8 o ESC, en ese momento el menú() retorna el mando al módulo de control general, quien se encarga de activar a la función.

La única función que el módulo activa directamente es la rutina de ayuda ayd(), con la tecla <F1>. Su contenido muestra una información breve de cada una de las funciones.

IV.4.1.5.2.1 Módulo *mgraf()*

La subrutina mgraf() detecta y coloca el modo gráfico adecuado para el monitor de la computadora. Trabaja con resoluciones mayores o iguales a 320 x 200 pixeles.

Primero detecta el hardware gráfico de la computadora, enseguida elige el modo gráfico adecuado y al termino de ello, inicializa la pantalla gráfica.

IV.4.1.5.2.2 Módulo *venmen()*

Este módulo hace los trazos del menú principal, lee los códigos de las figuras y dibuja las mismas para cada función del procesador.

Con los datos de la rutina mgraf(), se determinan los colores que se deben usar en las diversas clases de monitores y se ajustan las posiciones de las figuras, considerando la diferencia en resoluciones de video.

El módulo empieza por limpiar el monitor, dibujar los recuadros de las figuras, escribir el texto y pintar las propias figuras pixel a pixel.

IV.4.1.5.2.3 Módulo *efes()*

Cuando se hace el movimiento a través de menú, debe resaltarse la "F" seleccionada y desplegar una información breve de la función.

Para tales propósitos se creó el módulo efes(), como complemento a la elección de funciones en menú principal.

En su funcionamiento el módulo efes() recibe los parámetros que le indican el número de "F" a resaltar y el tipo de monitor a usar. En respuesta, efes() modifica el color de la nueva elección, restablece la "F" previa con su color original y despliega el resúmen de su acción, en la parte inferior del monitor.

IV.4.1.6 Caracteres especiales del español

En caso de contar con un teclado diferente al español o al latinoamericano, se ha construido una rutina específica que permite escribir los caracteres especiales del español desde teclado. En el programa se denota a la rutina con el nombre tespec().

La rutina tespec() recibe y examina los códigos de teclado que proporciona el módulo teclado(), con las secciones ascii y scan del carácter.

Tabla IV.3 Códigos especiales

CARACTER	COMBINACION	CODIGO SCAN	CODIGO ASCII
á	Control + a	1E H	1 H
é	Control + e	12 H	5 H
í	Control + i	17 H	9 H
ó	Control + o	18 H	0F H
ú	Control + u	16 H	15 H
ñ,Ñ	Control + n,N	31 H	0E H
ü,Ü	Control + u,U	16 H	0 H
¿	Alt + i	17 H	0 H
¡	Alt + a	1E H	0 H

La tabla IV.3 muestra la combinación que da acceso al carácter especial del español, junto a los códigos internos que le corresponden. Con base en ella, se realiza el patrón de comparación de los códigos ascii y scan. Sólo en circunstancias necesarias se detecta con la librería bioskey(), si el bloque de mayúsculas se encuentra activo.

IV.4.2 Funciones

En este caso el término funciones engloba a los módulos principales del procesador de texto, base del menú principal y pantalla de edición, asimismo, registra a las efes de teclado, desde F1 hasta F8.

IV.4.2.1 F1 Ayuda

Desde menú principal la función F1 utiliza al módulo ayd() para abrir el archivo AYUDA.TXT, leer su contenido y desplegar su información en monitor.

Cuando se llama al módulo de ayuda desde menú, se detecta la posición del cursor sobre las funciones para mostrar en forma inmediata la información de esa función. Al abrir el archivo AYUDA.TXT, se guarda momentáneamente su contenido en una variable tipo apuntador, al efectuar una asignación dinámica de memoria. Una vez desplegado el contenido, la rutina posibilita el recorrido en pantalla de todo el módulo de ayuda y regresar al mismo punto de partida cuando se decida salir del mismo.

Por otra parte, al manejar el módulo de ayuda desde la pantalla de edición, sólo muestra una ventana en la parte superior izquierda con información mínima de cada función.

Antes de colocar la ventana, se salva en memoria RAM la región de texto que piensa ocupar la ayuda para recuperar la información rápidamente, una vez concluído el proceso.

IV.4.2.2 F2 Escribir documento

Con la tecla F2 se inicia la escritura de un documento nuevo, se liberan las líneas del buffer si existe texto en él, se limpia la pantalla de edición y se inicializan las variables de control.

IV.4.2.3 F3 Salvar documento

La función F3 hace la transferencia de un documento en buffer hacia un archivo en disco. En la ejecución se auxilia de varios submódulos ya explicados con anterioridad, entre ellos podemos mencionar a guarda(), s_scroll(), s_box(), s_safe(), s_restore(), etc.

Otro módulo importante en la concepción de F3 es tecla2(),
una rutina receptora de la información de teclado, al momento de asignar nombre al archivo.

IV.4.2.3.1 Módulo *tecla2()*

El módulo tecla2() recibe de teclado el nombre de algún archivo, en sus diferentes acepciones y condiciones. Tecla2() valida su estructura al ser invocado por la función Guardar, Recuperar, Imprimir o Buscar en documento o diccionario.

Al módulo se le indica la posición (x,y) del cursor, la clase de opción a validar y la variable que guardará la cadena o palabra.

El módulo dispone de un movimiento de cursor hacia la derecha e izquierda para escribir o borrar facilmente con backspace o delete. En el recuadro o ventana aparece el cursor en espera

de una cadena de caracteres, que puede al final aceptarse con la tecla de <Return> o anularse con la tecla de <Escape>.

Una vez escrita, la cadena se valida con el módulo errdsc(), si la rutina se activo con la función Guarda Documento o Recupera Documento.

IV.4.2.3.2 Módulo *errdsc()*

El módulo detecta posibles errores de lectura o escritura cuando guarda o recupera un documento de disco. Precisamente al examinar el nombre del archivo con su directorio y disco sobre la cadena compuesta.

La rutina en cuestión usa las interrupciones 11H y 13H del BIOS, junto a la interrupción 21H del DOS en sus ascepciones 19H, 3BH y 36H. Ellas en su conjunto validan el disco y el directorio del archivo.

Interrupciones del BIOS:

11H Obtiene información del Equipo.

13H Verifica y Escribe sectores del disco.

Acepciones de la Interrupción 21H del DOS:

19H Obtiene el drive actual de trabajo.

3BH Cambia el path de trabajo.

36H Detecta espacio en disco.

IV.4.2.4 F4 Recuperar documento

La Función F4 recupera de disco el contenido de un archivo que es previamente escrito. Copia la información del archivo en disco a la memoria principal de la computadora y coloca la sección

inicial del archivo en la pantalla de edición, siempre y cuando el archivo pertenezca al procesador de texto.

La ejecución de la función se lleva a cabo principalmente con los módulos s_safe(), s_restore(), s_scroll(), s_box(), tecla2() y s_colinp(). Con ellos se coloca una ventana que permite escribir el nombre del documento, aceptar el nombre del archivo y abrir el documento de trabajo si cumple con los requisitos esenciales de disco, directorio y nombre de archivo.

Si al recuperar información en el buffer, existe información previa de otro documento, se libera la memoria-buffer para ajustar su tamaño a las nuevas dimensiones que el archivo demanda. Con esa asignación dinámica, se trabaja con la memoria requerida por un documento sin exceder su uso.

En el proceso final, la función actualiza las variables de control y coloca la primera parte del texto en pantalla, para iniciar la edición.

IV.4.2.5 F5 Imprimir documento

La función F5 imprime el contenido de un archivo o en su defecto la información que se halla en el buffer de trabajo. Para tales propósitos, el módulo de control manda a activar la rutina impri().

IV.4.2.5.1 Módulo *impri()*

El módulo impri() realiza la impresión en su totalidad, al recibir los parámetros que declaran la impresión del documento en disco o en buffer.

Antes de enviar la impresión se efectúa un sondeo que determina la viabilidad del proceso, ahorra tiempo en la ejecución y evade problemas del sistema operativo con el procesador.

Uno de estos pasos es el monitoreo de errores a través de la subrutina errimp(), que usa a la interrupción número 17H del BIOS para identificar los problemas de comunicación con la impresora. Dicha interrupción, provee a la computadora con la interface de impresión al puerto paralelo. Al hallar algún problema, errimp() manda un mensaje de error con la opción a cancelar la impresión o reintentar en el proceso, una vez arreglado el desperfecto.

Si no existen problemas de comunicación, la rutina solicita la impresión del buffer de trabajo o el archivo de disco. En caso de imprimir un archivo de disco, se gestiona con el módulo tecla2() el nombre del documento; recibe la escritura del nombre, examina los errores de disco, checa el nombre del archivo y directorio. En respuesta el módulo acepta al archivo de impresión o envía los respectivos mensajes de error.

Si el chequeo de errimp() y tecla2() es exitoso, se pregunta al usuario por la calidad de letra para iniciar el proceso de impresión.

En el proceso inicial de impresión se mandan las Secuencias de Escape que declaran el tamaño de papel y letra, con ayuda de la subrutina cprint(). Enseguida impri() activa la subrutina pagina(), para colocar en la parte superior derecha el número de página del documento.

Una rutina importante para los propósitos de impri() es el módulo espaciar(), que realiza el espaciado general de las líneas del documento. Su labor consiste en examinar detalladamente las características de la línea, con el fin de colocar los espacios suficientes entre palabra y palabra, y dejar un tamaño estandar de línea, bajo una distribución uniforme de palabras.

En la implementación no se utilizaron rutinas o librerías muy sofisticadas, se realizaron estudios de todas las posibles clases de líneas para llegar a implementar una secuencia lógica que contara palabras, espacios, caracteres de control y decidiera por las posiciones claves en la línea, donde colocar la cantidad acertada de espacios. Las rutinas auxiliares del módulo espaciar(), son posep() y ponesp(); una obtiene las posiciones claves y la otra coloca los espacios en blanco.

Después de espaciar la línea, se específica con cprint() el margen izquierdo en la impresora y se imprime la línea completa a través de la rutina sprint(). El proceso se repite hasta acabar con todas las líneas del documento, a menos que durante su impresión se cancele con la tecla de <Escape>. Para ello se usa la librería kbhit() que checa cualquier presión de teclas.

Si la impresión no se cancela, el proceso continúa hasta acabar con todos los renglones o líneas del documento.

IV.4.2.6 F6 Diccionario

La función forma parte de la subrutina general Control ().

Con la tecla F6 se efectúa el llamado a la subrutina diccort(). Existen dos formas de poder hacer este llamado: una desde el menú principal, la otra desde el texto.

En la primera, la subrutina general Control, detecta que el llamado ha sido desde el menú principal y activa el diccionario con indicaciones al respecto. Al terminar la ejecución de esta tarea, la subrutina general Control() nos deja en el texto en la posición desde donde fue activado el menú principal.

En la segunda forma de llamado, la subrutina general Control() detecta que se ha activado desde el texto, busca la posición del cursor en el texto para recuperar el término seleccionado

y una vez resaltado, lo guarda en una variable para después llamar la subrutina Diccort(), a la que le indica el término seleccionado y el medio ambiente desde el cual fue activada dicha subrutina.

Para realizar las tareas antes mencionadas nos basamos en la subrutina Valida ().

La subrutina general control(), al detectar que el llamado fue hecho desde el texto, checa la posición del cursor. Si éste se encuentra después de la línea 13 de la pantalla, accionará la subrutina s_pl() para subir el texto a la posición de la línea 2, activando la subrutina s_ps() para resguardar el texto de la línea 13 en adelante.

IV.4.2.6.1 Módulo *diccort()*

Esta subrutina se encarga de buscar en una lista de términos, la palabra seleccionada. Si la encuentra, mostrará el significado de dicha palabra. De lo contrario, posicionará el cursor en el rango de términos más cercano a la palabra buscada, con el objeto de que el usuario consulte el rango de términos más cercano al significado de lo que busca.

En el momento en que se activa la subrutina diccort() , entra en función la subrutina s_box() misma que abre una ventana del tamaño de la tercera parte de la pantalla. En dicha ventana se trabajará las tareas encomendadas a la subrutina diccort().

La subrutina s_box() sólo funciona para abrir la ventana.

Una vez abierta la ventana, se activará la subrutina s_ps() misma que colocará el nombre de la ventana.

Si la subrutina diccort() es usada por vez primera dentro de la sesión, se activará el submódulo llena() para que cargue en memoria principal un archivo de índices del acervo del Diccionario Palabra-Significado.

La información que recibe la subrutina diccort() al momento del llamado será una **D** para indicar un llamado desde el menú principal y una **d** para indicar un llamado desde el texto. En este último se proporciona también la palabra.

Diccort() revisa la información del llamado y si éste ha sido desde el menú principal, enviará un mensaje en la primer línea de la ventana del diccionario, pidiéndo el término a consultar mediante el mensaje: "Dame la palabra :" y activará la subrutina tecla2() misma que gestiona la entrada de datos carácter por carácter, desde el teclado. Tecla2() también nos permite borrar caracteres mal escritos, variar la posición del cursor dentro de la línea con las teclas de direccionamiento izquierda/derecha e insertar los caracteres necesarios. Una vez escrita la palabra, se presionará la tecla enter para dar continuidad a nuestra tarea.

Si al revisar la información, diccort() se da cuenta que el llamado fue hecho desde el texto, enviará el mensaje "palabra seleccionada:", mostrando enseguida dicha palabra.

El término seleccionado es revisado para checar que los caracteres y símbolos de dicho término correspondan al idioma español. Si se detecta un carácter no válido, se enviará el mensaje: "Es un carácter no válido dentro de la palabra" y se tomará como espacio en blanco, terminando con ello la revisión.

Una vez asegurados de que los caracteres del término corresponden al idioma español, se realizará la búsqueda de dicho término en el acervo del diccionario, a través de la subrutina

busca(), comparando ambos términos; o sea, el término seleccionado y el término localizado en el acervo del diccionario. Esta comparación se hace para detectar posibles errores de escritura.

La comparación arriba mencionada se realiza por medio de la subrutina compara().

Si al hacer la comparación, se detecta cualquier error de escritura, será emitido un aviso por medio de una ventana, indicando el error de que se trate. Dicha ventana permanecerá activa durante cinco segundos. Una vez desaparecida dicha ventana, mostrará la palabra escrita correctamente en una línea abajo. Dicha palabra permanecerá en pantalla por cinco segundos.

Una vez localizada la palabra, se abrirá el archivo del diccionario que la contiene y extraerá su significado, mediante la subrutina obten().

Una vez extraído el significado del término seleccionado, se desplegará en la ventana del diccionario dicho significado, mediante la subrutina muestra().

En caso de no haber sido localizado el término seleccionado, en el acervo del diccionario, se mostrará en una ventana de mensajes la leyenda: "No se encuentra en el diccionario". Dicha ventana permanecerá activa por tres segundos. Después se enviará otro mensaje preguntando: "¿Deseas checar en el dicc.(s/n)?" Si la respuesta es afirmativa, activará la subrutina Recdicc(). Dicha subrutina nos permitirá recorrer el diccionario y poder seleccionar uno o varios términos para ver su significado. Si la respuesta es negativa, y el llamado fue hecho desde el menú principal, nos preguntará por una nueva palabra para buscar su significado. El llamado se hará mediante el mensaje: "Dame la palabra:" Si ya no se desea seguir consultando más términos, se presionará la tecla enter y nos enviará a un menú de dos opciones. Una de ellas nos permitirá salir definitivamente de esta tarea y la otra permitirá continuar con la tarea de seguir consultando el diccionario. Si la respuesta es negativa y el llamado fue hecho desde el texto, entonces

regresará a éste, colocando el documento y el cursor en la posición desde donde fué llamado al diccionario.

IV.4.2.6.2. Módulo llena()

Esta subrutina se encarga de leer el archivo de índices de los términos del diccionario y colocarlos en la memoria principal de la computadora.

Para ello, nos basamos en una instrucción del lenguaje "C": fgets(), el cual nos recupera del archivo un conjunto de caracteres que se le indiquen. Este conjunto de caracteres es dividido en tres subconjuntos. En el primer subconjunto se tiene el número correspondiente al archivo; en el segundo se tiene la posición que ocupa la palabra en el archivo y en el tercer subconjunto se tiene la palabra misma.

Posteriormente, se utilizó otra de las funciones del lenguaje "C" de asignación de memoria dinámica para almacenar la información de los subconjuntos. Cabe hacer notar que la asignación mínima de memoria hecha por el sistema es de 16 bytes, y la asignación máxima que estamos ocupando es de 32 bytes; repartidos de la siguiente manera: dos bytes para la localización de archivo, cuatro bytes para la posición de la palabra en el archivo y veinticuatro bytes para la palabra misma. Es importante destacar que veinticuatro bytes es la longitud máxima que puede tener una palabra y la longitud mínima es de un byte.

Es importante tener presente que el sistema da acceso a toda la memoria principal, pero el tamaño de la memoria ocupada para poder almacenar 6400 palabras es de 200 Kb, lo que significa que no se tendrán problemas con respecto a la capacidad de memoria en caso de que creciera el número de palabras.

Esta subrutina prevee los siguientes casos :

- Si en la memoria principal no se tiene el espacio mínimo suficiente para guardar todo el índice del diccionario, envia un mensaje en el cual indica que no existe la memoria suficiente para el diccionario.
- Si al momento de realizar las asignaciones de memoria principal no se tiene el espacio suficiente para guardar el índice del diccionario, envia un mensaje en el cual indica que la memoria se ha llenado y todas aquellas asignaciones hechas hasta antes de que se detecte este problema de memoria, serán desactivadas.
- Si las palabras que componen el diccionario son mayores a las que puede manejar, envia un mensaje indicando que el Diccionario NO fue cargado en su totalidad, e informa la cantidad de palabras que se encuentran en memoria.

IV.4.2.6.3 Módulo *Obten()*

Esta subrutina se encarga de obtener de los archivos el significado de una palabra, para ello es necesario indicar la siguiente información: a) nombre del archivo donde se encuentra el significado de la palabra b) la posición donde comienza el significado de la palabra c) la palabra misma con el fin de cotejarla.

IV.4.2.6.4 Módulo *Busca()*

Esta subrutina se encarga de verificar en el diccionario la existencia de una palabra. En caso de que dicha palabra se encuentre en la lista, devolverá un uno, nombre del archivo donde se encuentra la palabra, posición donde se localiza el significado de dicha palabra y la palabra

misma. En caso negativo, regresará el valor cero, la posición previa o posterior a la palabra buscada. Cabe destacar que esta subrutina se basa principalmente en la subrutina Compara().

El método de búsqueda utilizado en esta subrutina fué el binario, el cual consiste en seguir los siguientes pasos:

1.- La lista se ordena en forma ascendente (en este caso alfabéticamente).

2.- Se divide la lista en dos.

3.- Se selecciona la palabra que quede intermedia entre las dos partes; en la parte I estaran todos los términos que alfabeticamente son menores a la palabra selecionada y en la parte II estaran todos los términos que alfabeticamente son mayores a la palabra selecionada.

4.- La palabra seleccionada se compara con la palabra a buscar. Si esta última es mayor que el término seleccionado, entonces se selecciona la lista de términos que alfabéticamente son mayores a la palabra seleccionada. De lo contrario se selecciona la lista de términos que alfabéticamente son menores a la palabra seleccionada.

5.- Ahora bien, si la palabra seleccionada es igual al término buscado, entonces el propósito se ha conseguido, ya que esto indica que se ha localizado la palabra buscada y se da por terminado el método.

6.- Si la palabra seleccionada es diferente al término buscado, se tendrá que repetir el método de búsqueda a partir del paso no. 2 con la lista seleccionada en el paso no. 4. En caso de que dicha lista se agotara, se dará por concluído el método y se indicará que la palabra no fué encontrada.

IV.4.2.6.5 Módulo *Compara()*

La tarea de esta subrutina consiste en comparar dos palabras que le son proporcionadas como información. Dicha comparación se realiza carácter por carácter, sin importar si son

mayúsculas, minúsculas, eñes, diéresis y acentos, ya que aquí mismo se acciona la subrutina Checa(), que convierte esos caracteres a la forma apropiada.

Si la palabra uno es menor que la palabra dos, la subrutina Compara() nos regresara un valor negativo. Si la palabra uno es igual a la palabra dos, dicha subrutina nos regresará un valor cero. Si la palabra uno es mayor que la palabra dos, nos regresará un valor positivo.

IV.4.2.6.6 Módulo *Checa()*

El objetivo de esta subrutina es estandarizar caracteres especiales -tales como diéresis, acentos y eñes- a caracteres estándar, y convierte mayúsculas a minúsculas, con el fin de poder facilitar su comparación.

IV.4.2.6.7 Módulo *Erracen()*

La tarea de esta subrutina consiste en enviar los mensajes de error de acentuación que de acuerdo a ciertos códigos pondrá:

- Sobra acento
- Sobran diéresis
- Falta acento
- Faltan diéresis
- Sobran acentos
- Sobran diéresis y acentos
- Acento mal colocado
- Sobra acento y faltan diéresis
- Falta acento y sobran diéresis
- Diéresis mal colocado
- Faltan acentos
- Falta acento y diéresis

El mensaje respectivo permanecerá en la ventana de Dicconario cinco segundos.

IV.4.2.6.8 Módulo *Muestra()*

Esta subrutina se encarga de mostrar en la ventana del diccionario la palabra seleccionada junto con su significado, se encarga también de realizar la justificación izquierda y derecha para evitar que las palabras que recibe a través de la subrutina Obten() queden cortadas. Esta justificación se realiza en un espacio de 35 caracteres por línea, y se tienen 8 líneas por ventana.

IV.4.2.7 F7 Búsqueda

La función forma parte de la subrutina general control().

Con la tecla F7 se efectúa la busqueda de palabras, signos, símbolos o cadenas específicas de caracteres. Al accionarla se pide en forma inmediata la cadena de caracteres a buscar.

La subrutina a cargo de la primer etapa es tecla2(), que gestiona la entrada de datos. Esta subrutina recibe de teclado carácter por carácter hasta llegar a completar la cadena deseada y aceptar si cumple con la longitud no mayor a 25 caracteres.

Durante la escritura de la cadena, tecla2() permite borrar los caracteres mal escritos, variar la posición sobre la línea con el direccionamiento de cursor derecha e izquierda, insertar los caracteres y admitir finalmente la frase con la tecla <Rtn> o rechazar con la tecla <Esc>.

Para solicitar la dirección de búsqueda hacia adelante o hacia atrás se auxilia de las subrutinas:

s_ps()	Escribe un string con atributo.
s_posyx()	Posiciona el cursor en pantalla.
int86(16)	Librería de C para teclado (Interrupción 16H).

Después de la primera etapa de entrada, continuamos con la búsqueda directa en el documento. Para lo cual se implementó la subrutina buscapal().

En caso que la búsqueda sea exitosa, se muestra el nuevo contexto con la palabra en color diferente para enseguida preguntar y decidir si continúa con la misma búsqueda. El proceso termina cuando buscapal() ya no encuentra más palabras o al indicar directamente su finalización. Por consiguiente se regresa el mando a la subrutina general de programa, control().

Como tercera y última etapa, se actualizan las posiciones del cursor, se refresca con las rutinas s_colinp(), s_ps() y flesc() la sección inferior de la pantalla, y se translada a estado de espera para cualquier otra función.

IV.4.2.7.1 Módulo *buscapal()*

Esta subrutina se encarga de buscar un conjunto de caracteres en el documento de trabajo que se halla en la memoria RAM de la máquina.

Las variables de entrada que recibe la subrutina, llámese información, corresponden a la dirección de búsqueda, al conjunto de caracteres a buscar, a las posiciones del cursor sobre el documento y a las posiciones sobre la pantalla.

Con los datos de entrada se conoce la ubicación exacta de la edición, la(s) palabra(s) que desean buscarse así como la dirección de búsqueda. Por consecuencia se cuenta con el punto de partida sobre el documento íntegro.

La búsqueda en sí, se realiza bajo un patrón de comparación carácter por carácter entre el punto de partida en el texto y la cadena especificada, que van desde la posición de partida

(cursor) hasta el final o inicio del documento. Con la concordancia en el seguimiento de la cadena, desde el inicio al final o viceversa.

Si la búsqueda es hacia adelante, el patrón de comparación toma a los caracteres desde el punto de partida hasta el final, tanto para el texto como para la cadena especificada. Si la búsqueda es hacia atrás se aplica la misma fórmula pero a la inversa, empezando siempre desde atrás hasta el inicio.

Durante su comparación carácter por carácter se lleva el estado de la busqueda para indicar si los caracteres encontrados hasta el momento son semejantes en una parte, en su totalidad o en realidad no existen dentro del documento.

En esta comparación se tiene un cuidado especial con las palabras que se encuentran separadas por espacios en blanco, returns, caracteres de nueva línea y caracteres de continuación de palabra, que corresponden a las múltiples ocurrencias de cadenas inmersas en dos líneas de texto.

La subrutina buscapal() concluye su búsqueda al presentar en pantalla la sección de texto que contiene la cadena. Esta última fase actualiza las posiciones de línea, columna y página, y reacomoda el texto para exhibir en el monitor la fracción de documento que contiene la cadena.

Durante esta fase, un color diferente al resto de la información resalta a la cadena buscada, mientras se pregunta

"¿Continua? (s/n)",

cuando la respuesta es afirmativa, continua con la misma búsqueda de cadena y en la misma dirección. En caso contrario se cancela el proceso.

Si una búsqueda no es exitosa se indica en pantalla con la frase

"No encontrado"

y un sonido adicional.

Por otra parte, la implementación completa de la rutina buscapal() usa para impresión en pantalla a las subrutinas s_ps() y s_pl() principalmente. Mientras que para la búsqueda directa se elaboran sus controles con instrucciones propias del lenguaje.

IV.4.2.8 F8 Salir

Con la función Salir, se cancela la sesión dentro del procesador de texto. En el momento de presionar la tecla F8, el módulo de control libera la memoria ocupada y sale del procesador de texto.

IV.5. Inserción

Esta sección es de las más importantes en la edición, su desarrollo marca la pauta en el diseño de otras funcionalidades del procesador, por significar el punto de partida de pruebas.

Inscar() es el módulo encargado del proceso de inserción, su ejecución es "independiente" al de control general. El módulo de control sólo actúa como filtro al resto de funciones del procesador; en caso de no activar a ninguna se toma por omisión la entrada al módulo de inserción (su acción puede confirmarse en la fig. IV.1). Una doble función de inscar() es la sobre-escritura de texto.

IV.5.1.1. Módulo *inscar()*

La rutina inscar() inserta caracteres sobre la línea de edición. Recibe los parámetros de número de columna en buffer, la posición (x, y) en monitor y el carácter que desea escribirse; a cambio, inscar() examina el dato bajo una inspección rigurosa.

En este proceso se analiza la clase de edición (inserción o sobre-escritura), el tipo de carácter a insertar, la posición de la columna a escribir y la clase de renglón que recibe el carácter, tanto línea grande como línea normal.

Aunque en sobre-escritura se cambia el carácter viejo por el nuevo, debe considerarse el tipo de carácter a eliminar, como sucede con los caracteres de control. Si una línea grande es afectada con un carácter en blanco, se usa la subrutina sube_lg()[13] para manejar el proceso, mientras que los demás casos se solucionan fácilmente o se canalizan como inserción de caracteres.

En la inserción de caracteres, aparte de examinar a los caracteres de control, también se determina la saturación de la línea al momento de insertar un carácter. Después de realizar comparaciones en la inspección de la línea y caracteres, se mueve la información en el renglón (buffer) con la librería movmem(). Al completar la línea se checa su saturación, observando la posibilidad de bajar palabras o espacios a la siguiente línea con el módulo baja_p(). Si baja algún carácter, la línea en curso debe quedar actualizada.

Con baja_p() se mueven las palabras a las líneas subsiguientes hasta dejar al corriente cada uno de los renglones.

---

13 Esta rutina también la usa el módulo de borrado.

Después de salir del módulo baja_p() se termina con la actualización de la línea de status y el posicionamiento del cursor en el display.

Durante las pruebas finales de inserción se encontró la necesidad de actualización de texto sobre la línea anterior al cursor, al considerar la inserción de un espacio en blanco que separara la primer palabra de la línea en dos, y diera lugar a un movimiento de caracteres hacia la línea anterior. Es decir, normalmente se asocia la adición de caracteres con el movimiento de texto hacia la derecha del cursor y nunca hacia atrás o arriba, como sucede con la inserción de un espacio que divide la primer palabra y permite subir esa cadena a la línea anterior, por la sencilla razón de que existe espacio suficiente.

Con ese proceso se deja completamente actualizado al conjunto de líneas posteriores y anteriores al cursor, después de una inserción.

IV.5.1.1.1 Módulo *baja_p()*

El módulo baja_p() es la rutina que mueve las palabras o espacios a la línea siguiente del cursor, al momento de saturar la capacidad del renglón.

En su implementación utiliza una programación recursiva y una asignación dinámica de memoria.

Los parámetros que recibe de inscar() son:

tmp : El apuntador que contiene las palabras.

cont : La longitud de tmp.

n : Variable que indica el nivel de recursión.

lg : Indica si el proceso pertenece a una línea grande.

Cuando se activa el módulo baja_p(), se analiza antes de cualquier movimiento la línea que ha de recibir la información de tmp. En caso de corresponder al último renglón, se adiciona una línea nueva al buffer mediante una asignación dinámica de memoria, una vez concluída se traslada la información de tmp a la línea y concluye el proceso. En caso contrario, se calcula el espacio disponible en la línea para albergar a tmp o para insertar una línea al documento, con el módulo insnl().

Si el contenido de tmp satura la capacidad de la línea, se preparan las variables para ejecutar un proceso recursivo con baja_p(). La recursión permite llamar el módulo a sí mismo las veces que se considere necesario hasta completar el movimiento de palabras en forma automática.

Por otra parte, el conjunto de problemas resueltos para desarrollar al módulo baja_p(), incluyen el manejo de longitudes de palabras, las clases de líneas (línea, grande o normal), la afectación de caracteres de control en el movimiento, el cálculo de las posiciones en la línea para situar información, la inspección de renglones para mover palabras completas y no trozos de ellas, la asignación dinámica de memoria, etc.

IV.6 Actualización del Diccionario

Al accionar este programa "ACTUAL", apareceran dos ventanas en la pantalla. Una de ellas será titulada "Aviso" y constará de siete líneas con treinta y seis caracteres por línea. Esta ventana contendrá la leyenda: "Este programa fue realizado para actualizar el diccionario, con nuevas palabras y nuevos significados. Para ello, se deberán usar letras minúsculas, sin olvidar la acentuación" (véase fig. IV.2).

La otra ventana, se llamará "actualiza", y constará de catorce líneas con treinta y seis caracteres por línea; de las cuales trece serán utilizadas por el sistema para la realización de todo el trabajo de actualización. La línea restante se llamará línea de estado y será utilizada por el programa para mandar mensajes o avisos necesarios.

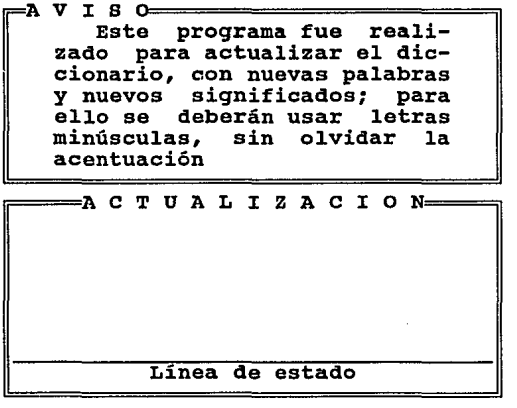


Fig. IV.2 Ventanas del programa actual

Ahora bien, la leyenda de la primera ventana permanecerá fija mientras dura la sesión y sólo serán remarcadas algunas partes de la leyenda cuando se considere necesario.

Siempre que se inicie una sesión aparecerá una ventana enmarcada en gris, con fondo negro y letras blancas; conteniendo el mensaje: "Leyendo: Diccionario" para indicarnos que el índice del diccionario está siendo almacenado en memoria.

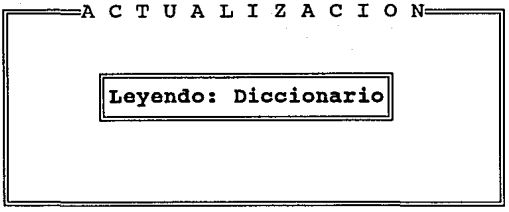


Fig. IV.3 Ventana indicando la lectura del índice

Cabe hacer notar que en caso de haber algún problema con el espacio de la memoria o la lectura del índice del diccionario, será desplegado un mensaje, por medio de una ventana enmarcada en blanco con fondo rojo y letras blancas centellantes para indicar la falla de que se trate; tres segundos después dará por terminada la sesión y saldrá del sistema.

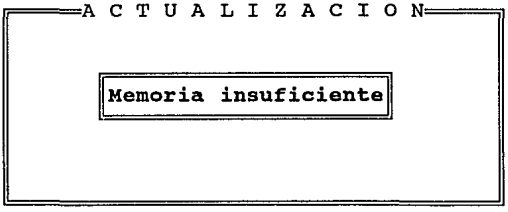


Fig. IV.4 Ventana indicando error en el almacenamiento del índice

Cuando desaparece la ventana que contiene el mensaje "Leyendo el diccionario", aparecerá en la línea uno de la ventana de actualiza, el mensaje "Dame la palabra:". Este mensaje se encarga de pedir el término que se va a agregar y/o modificar (véase fig. IV.5). El sistema analizará dicho término, verificará que la palabra esté compuesta por caracteres en español y sin ningún tipo de signos ni números; ya que si hubiera algún signo diferente de

los caracteres en español sólo reconocerá estos últimos, cortando dicho término en donde aparecen esos signos.

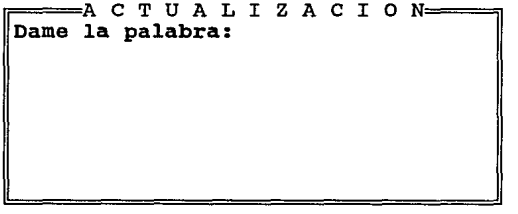


Fig. IV.5 Ventana solicitando el término a agregar y/o modificar

Después cambiará el mensaje "Dame la palabra:" por el mensaje "La palabra:"; agregando el término analizado. Preguntará si el término examinado es correcto, mediante el mensaje: "¿Es correcta? (s/n)".

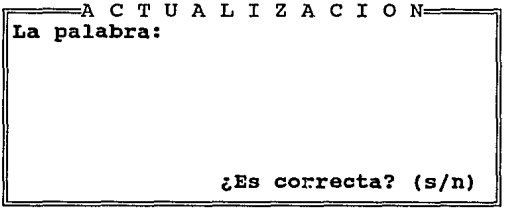


Fig. IV.6 Ventana mostrando el término analizado y solicitando su validez

Si la respuesta es negativa volverá a pedir nuevamente la palabra a agregar o/y modificar. Si se responde afirmativamente se verificará la existencia de éste en el acervo. Mientras se está realizando la búsqueda, aparece en la línea de estado a la izquierda, en

fondo gris, letras blancas y centellantes, el mensaje "Espera", el cual permanecerá centellando hasta terminar la búsqueda.

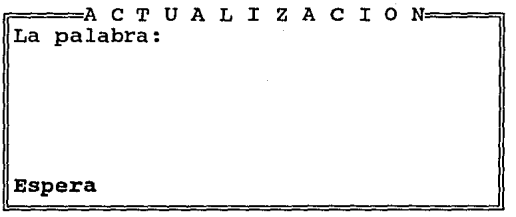


Fig. IV.7 Ventana indicando esperar mientras se realiza la búsqueda del término

Si no se encontró la palabra en el diccionario, el sistema desplegará una ventana enmarcada en gris, fondo rojo y con el mensaje en letras blancas y centellantes: "No se encuentra en el diccionario".

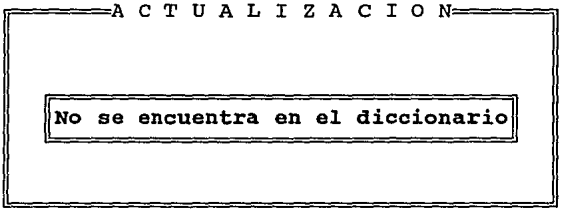


Fig. IV.8 Ventana indicando la no existencia del termino en el acervo

Tres segundos después, preguntará si se desea adicionar dicho término mediante el mensaje "¿Deseas agregarla al diccionario? (s/n)", mismo que aparecerá en la línea de estado en fondo gris y letras azules.

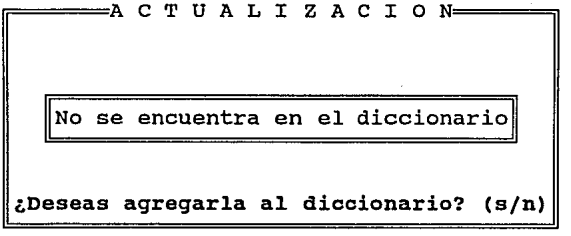


Fig. IV.9 Ventana interrogando si se agrega el término al acervo

Si a la pregunta arriba formulada, se responde negativamente, limpiará la ventana para que aparezca en la línea de estado, en fondo gris y letras azules, el mensaje: "¿Alguna otra modificación? (s/n)" (véase fig. IV.10). Si a esta nueva requisición se contesta que no, se dará por terminada la tarea de actualización y se saldrá del sistema. Si se responde positivamente, regresará al inicio de la tarea, solicitando la nueva palabra.

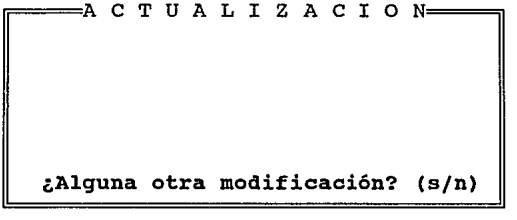


Fig. IV.10 Ventana preguntando si se desean realizar más actualizaciones

Cuando el sistema pregunta si se desea agregar el término al diccionario y se contesta afirmativamente, desaparecerá el contenido de esta ventana y en su lugar se mostrará a partir de la línea uno de dicha ventana, la leyenda "RECUERDA: Debes acentuar los términos correctamente". En este caso, la palabra "RECUERDA" aparecerá en fondo gris y letras azules y el resto del texto en letras grises y fondo azul. Enseguida se exhibirá el vocablo que se quiere agregar al acervo. Dicho vocablo será destacado en fondo rojo y letras blancas. En la siguiente línea se mostrará en letras blancas y fondo azul, la leyenda: "Tendrá como significado:" para que se pueda colocar en las siguientes ocho líneas el significado de la palabra que se va a agregar, mismo que puede tener una longitud máxima de doscientos ochenta caracteres y un mínimo de cuatro caracteres (véase fig. IV.11).

Cabe señalar que el programa cuenta con un pequeño editor que posibilita la inserción y borrado de caracteres, movimiento del cursor hacia arriba, hacia abajo y hacia los lados dentro de las ocho líneas arriba mencionadas. Dicho editor se encarga de indicar mediante un "bip" el exceso de caracteres en el momento de proporcionar el significado de los vocablos (véase fig. IV.12).

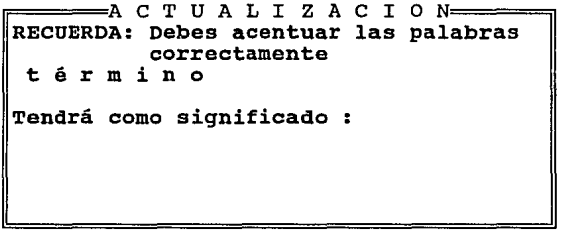


Fig. IV.11 Ventana mostrando el término y significado a modificar

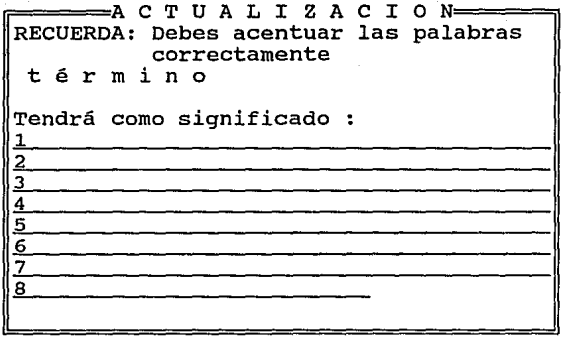


Fig. IV.12 Ventana en la que se exhiben las ocho líneas de modificación de significado

Otra función del editor, consiste en evitar salir de la captura del significado si éste contiene menos caracteres del mínimo requerido y enviará en la línea de estado, en fondo gris, letras azules y con una duración de tres segundos, el mensaje: "Tiene que ser mayor de cuatro caracteres".

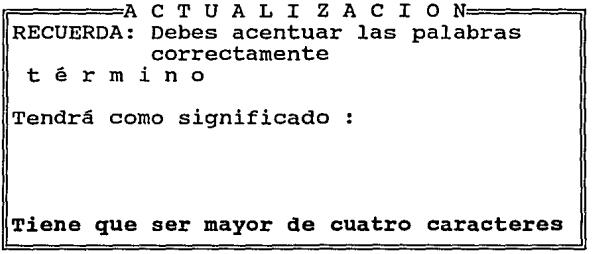


Fig. IV.13 Ventana indicando que el significado debe tener de más de cuatro caracteres

Una vez escrito el significado del término, el sistema validará dicho significado enviando en la línea de estado, en fondo gris y letras azules, el siguiente mensaje: "¿Es correcto el significado? (s/n)". Si la respuesta a esta pregunta es positiva, actualizará el diccionario, de lo contrario, ignorará la adición y regresa a pedir un nuevo término.

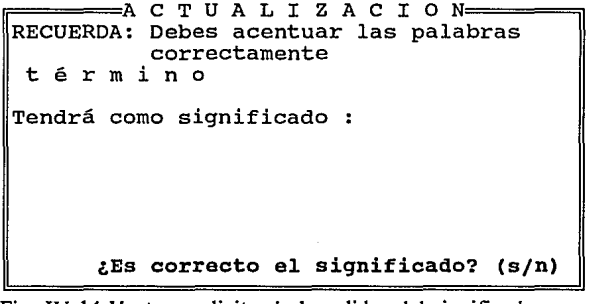


Fig. IV.14 Ventana solicitando la validez del significado

Ahora bien, si el término de interés existe en el diccionario, aparecerá en la ventana, en fondo azul y letras grises el término junto con su significado y tres segundos después, preguntará en la línea de estado, en fondo gris y letras azules: "¿Deseas cambiar el significado? (s/n)".

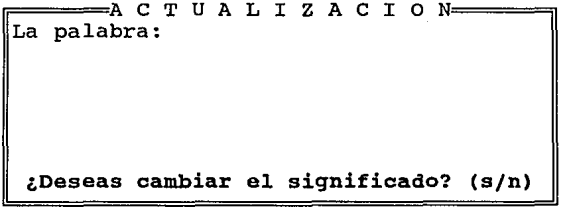


Fig. IV.15 Ventana preguntando si se desea modificar el significado

Si a esta nueva pregunta se contesta afirmativamente, se mostrará a partir de la línea uno de la ventana actualiza, la leyenda "RECUERDA: Debes acentuar los términos correctamente". En este caso, la palabra "RECUERDA" aparecerá en fondo gris y letras azules y el resto del texto en letras grises y fondo azul. Enseguida se exhibirá el vocablo que se quiere adicionar al acervo. Dicho vocablo será destacado en fondo rojo y letras blancas. En la siguiente línea se mostrará en letras blancas y fondo azul, la leyenda: "Tendrá como significado:" para que se pueda colocar en las siguientes ocho líneas el significado de la palabra que se va a agregar, mismo que puede tener una longitud máxima de doscientos ochenta caracteres y un mínimo de cuatro caracteres.

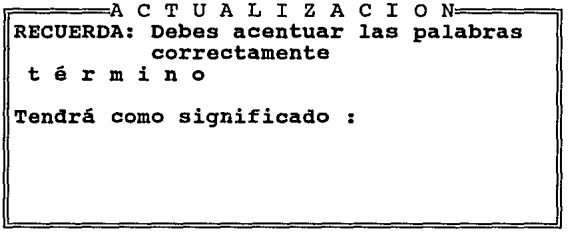


Fig. IV.16 Ventana presentando el término y el significado a modificar

Cabe señalar que el programa cuenta con un pequeño editor que posibilita la inserción y borrado de caracteres, movimiento del cursor hacia arriba, hacia abajo y hacia los lados dentro de las ocho líneas arriba mencionadas. Dicho editor se encarga de indicar mediante un "bip" el exceso de caracteres en el momento de proporcionar el significado de los vocablos.

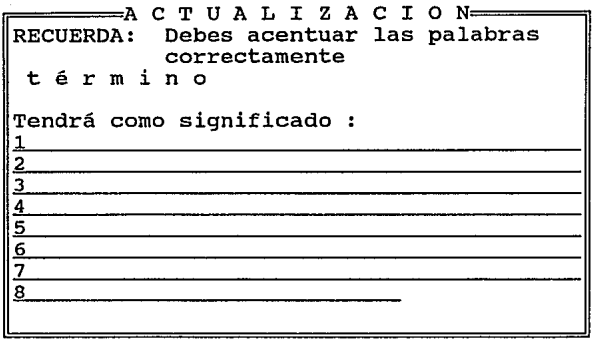


Fig. IV.17 Ventana en la que se exhiben las ocho líneas de modificación del significado

Otra función del editor, consiste en evitar salir de la captura del significado si éste contiene menos caracteres del mínimo requerido. En tal caso, enviará en la línea de estado, en fondo gris, letras azules y con una duración de tres segundos, el mensaje: "Tiene que ser mayor de cuatro caracteres".

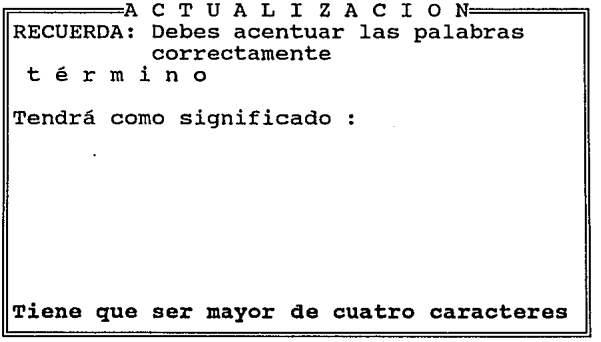


Fig. IV.18 Ventana indicando que el significado debe tener de más de cuatro caracteres

Una vez escrito el significado del término, el sistema validará dicho significado enviando en la línea de estado, en fondo gris y letras azules, el siguiente mensaje: "¿Es correcto el significado? (s/n)". Si la respuesta a esta pregunta es positiva, actualizará el diccionario, de lo contrario, ignorará la adición y regresará a solicitar un nuevo término.

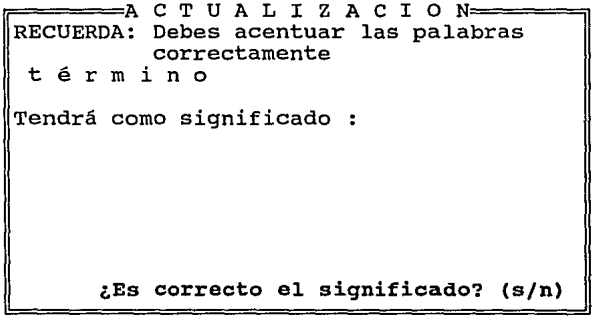


Fig. IV.19 Ventana preguntando si el significado es el correcto

Si cuando preguntó si se deseaba cambiar el significado, se contestó negativamente, aparecerá en la línea de estado, en fondo gris y letras azules, el mensaje: "¿Alguna otra modificación? (s/n)", si la respuesta es "s", pedirá el nuevo término a modificar y/o agregar, continuando la tarea de acuerdo a lo descrito anteriormente, de lo contrario se dará por concluida la actualización.

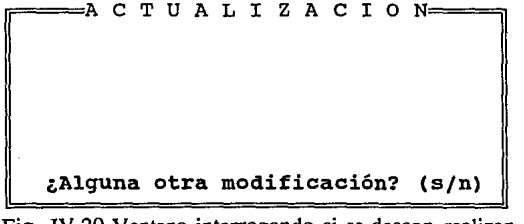


Fig. IV.20 Ventana interrogando si se desean realizar más actualizaciones

Antes de salir del sistema, a la izquierda de la línea de estado, en fondo gris y letras blancas centellantes, aparecerá el mensaje: "Actualizando" para indicar que se está haciendo la actualización del índice del diccionario y que se está grabando en uno de los archivos del diccionario, la(s) palabra(s) agregada(s) y/o modificada(s).

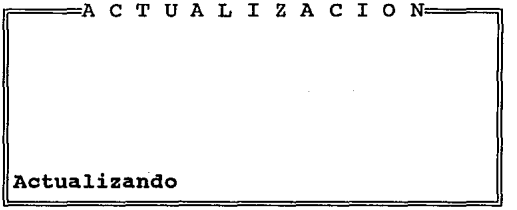


Fig. IV.21 Ventana indicando la actualización del diccionario y el índice

Cabe señalar que en la actualización del índice del diccionario, se grabará por cada palabra lo siguiente:

1) La palabra misma, en donde se utilizará mínimo un carácter y máximo veintitrés caracteres

2) El número del archivo en donde se encuentra la palabra y su significado, en donde se utilizará un carácter solamente

3) La posición del inicio de la palabra y su significado, en donde se utilizan cuatro caracteres, mismos que representarán la base numérica 50 y cuyos dígitos que se seleccionaron fueron los siguientes: **0** al **9**, **:**, **;**, **<**, **=**, **>**, **?**, **@**, **A** a la **Z**, ****, **[**, **]**, **^**, **-**, **'** y **a**, que dichos dígitos en código ASCII (American Standar Code for Information Interchange) corresponden del número 48 al 97. La base numérica 50 fue utilizada para poder disminuir la cantidad de dígitos que era variable entre 4 y 7. Con esto se evitó la sobredemanda variable de memoria principal y se logró reducir ésta a una cantidad estándar. Dado que la longitud que puede lograr un archivo en disco o memoria secundaria puede ser de

millones de bytes (megabytes) y puede llegar a necesitarse 7 dígitos para representar dicha cifra, en cambio, utilizando la herramienta antes mencionada, o sea la base numérica 50, esas necesidades de dígitos se reducen a 4.

Una vez actualizado el índice con la nueva localización de las palabras que se incluyeron y/o modificarón en el diccionario, se dará por concluida la tarea y se saldrá del sistema.

IV.7 Pruebas y adiciones finales

Durante las pruebas finales del sistema se consideró pertinente la afinación de ciertos aspectos del procesador, con la intención de mejorar el manejo y auxiliar al niño para no perder los documentos escritos.

Como medida de protección para evitar la pérdida de documentos se creó la subrutina alerta() y se modificaron las funciones para escribir (F2), guardar (F3) y recuperar (F4) un documento. Es decir, con las modificaciones finales se logra advertir una posible pérdida en los siguientes casos:

a) Cuando solicita escribir un documento nuevo (F2) y no ha guardado el documento de trabajo actual.

b) Cuando intenta recuperar un documento de disco (F4) y no ha guardado el trabajo actual.

c) Cuando desea salir del procesador de texto (F8) y no ha guardado el documento actual.

Se advertirá siempre y cuando sea necesario hacerlo, porque el sistema determinará si se recuperó un archivo sólo para observar su contenido o para editarse (observando incluso la inserción de un simple carácter).

Otra medida para proteger los documentos, se registra cuando existe un documento con el mismo nombre al momento de guardar el trabajo actual. Pregunta al niño si desea remplazar el contenido del archivo que existe en disco con el documento en edición.

Por otro lado, para hacer más flexible y amigable al procesador se han hecho las siguientes mejoras:

a) Permitir desde la función de ayuda de edición (F1), ejecutar el conjunto de funciones.

b) Colocar el nombre del documento de trabajo en la parte superior de la pantalla, al instante que se asigna nombre con guardar (F3) o recuperar (F4).

c) Al solicitar guardar un documento (F3) que ya tiene nombre, se checa inmediatamente el nombre actual y se coloca en la ventana de guardar para dar simplemente <Enter>. Con la posibilidad de modificarlo antes de aceptarlo.

d) Finalmente, para recuperar (F4) se ha diseñado un menú que permite observar y seleccionar el contenido de los discos disponibles en la computadora. De esta manera, se facilita la selección de un documento que se desea editar.

Las subrutinas que se mejoraron son:

control() : Control general.
errdsc() : Errores de disco.
ay() : Ayuda en edición.
salvar() : Guardar documento de trabajo.

Las subrutinas que se adicionaron son:

muesdir(), dir(), dislog() : Subrutinas para poder recuperar un documento a tráves de un menú de selección de archivos.

alerta() : Subrutina que detecta si el documento en edición no ha sido guardado.

Capítulo V Manual del procesador de texto

El Manual del Procesador de Texto es una guía o referencia ordenada que puede consultar el profesor o el analista para auxiliar al niño. Ofrece la posibilidad de consulta por iniciativa del niño, ya sea solo o en compañía del profesor. Para dichos propósitos se ha creado un Manual de usuario y un Manual técnico.

V.1 Manual del usuario

Para su aprendizaje, el Procesador de Texto se ha dividido en los siguientes puntos:

V.1.1 Entrada y salida del procesador
V.1.2 Menú principal
V.1.3 Ayuda
V.1.4 Escribir y modificar un documento
V.1.5 Modificar un documento
V.1.6 Guardar un documento
V.1.7 Recuperar un documento
V.1.8 Búsqueda
V.1.9 Diccionario
V.1.10 Impresión

V.1.1 Entrada y salida del procesador

Entrar al procesador

Para entrar al programa es necesario teclear en el prompt (o desplegado) de la computadora, el nombre **palabra**.

Ej. C:\palabra\palabra

Una vez ejecutado, dentro del procesador aparece la Pantalla de Presentación seguida del Menú Principal.

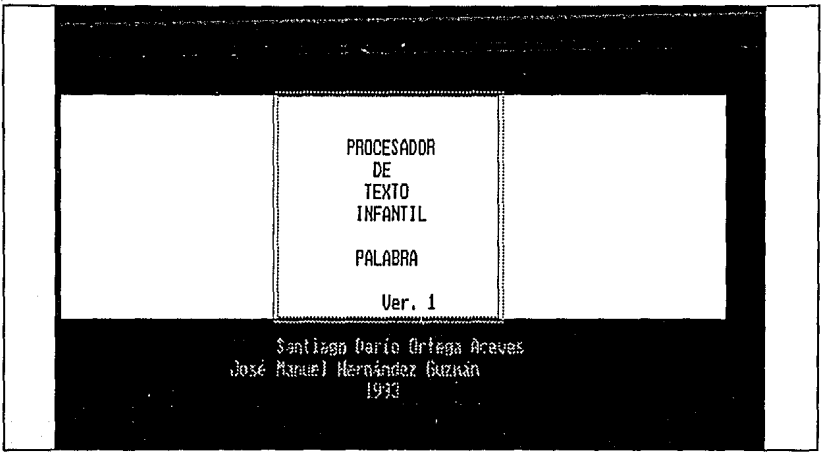


Fig. V.1 Pantalla de Presentación

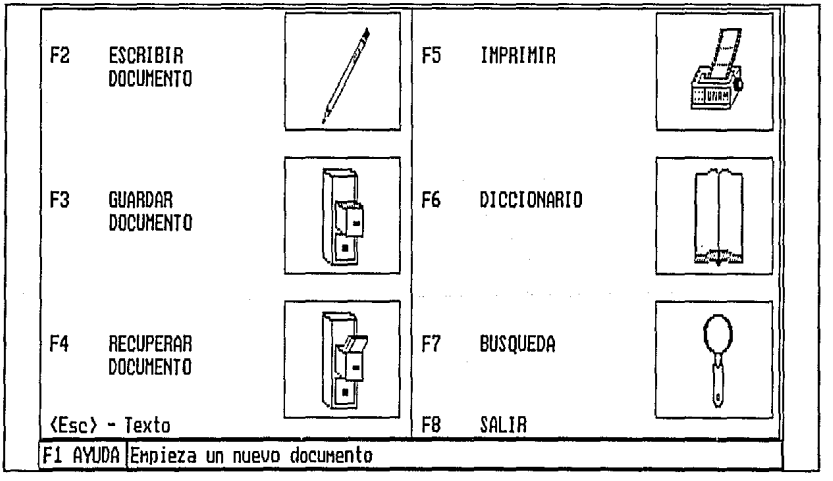


Fig. V.2 Menú Principal

Salir del procesador

La forma de salir del Procesador de Texto es a través de la tecla <F8>.

- Presionar la tecla <F8>

V.1.2 Menú Principal

El Menú Principal indica, a través de figuras, las funciones disponibles dentro del Procesador de Texto Infantil.

Modo de activar funciones

En el menú existen dos formas de activar las funciones;

Forma 1:

- Presionar, directamente en el teclado, la función correspondiente: <F1>, <F2>, <F3>, <F4>, <F5>, <F6>, <F7> o <F8>.

Forma 2:

- Seleccionar, con las teclas de movimiento de cursor (←, →, ↑, ↓), la función correspondiente y presionar enseguida <Return> o <Intro>.

Cambio de menu principal a pantalla de edición

Si se desea cambiar de menú principal a la pantalla de edición:

- Presionar la tecla <Esc>

Enseguida muestra el área de edición de texto.

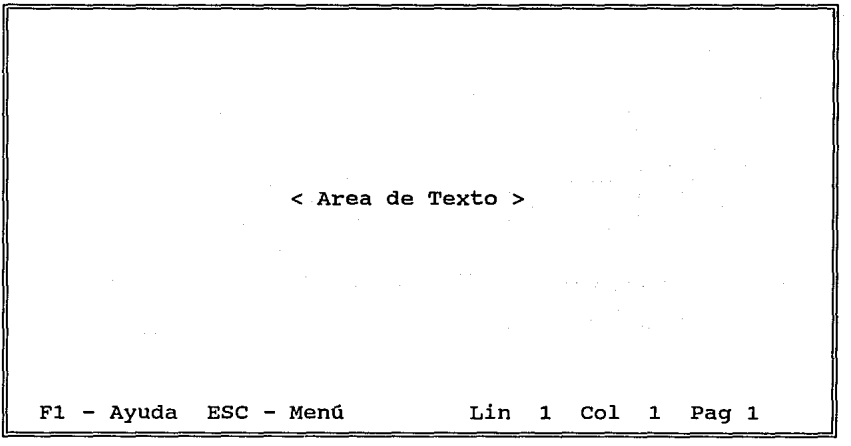


Fig. V.3 Pantalla de Edición

V.1.3 Ayuda

La ayuda es una explicación breve de las funciones principales del procesador. Para observarla basta presionar la tecla <F1>.

Ayuda en menu principal

- Presionar <F1>.

Muestra en Pantalla, la Figura V.4 :

- Si se desea recorrer la ayuda por páginas, se usa la tecla <Pg Up> o <Re Pág> para subir y <Pg Dn> o <Av Pág> para bajar.

- Si se desea salir de la ayuda, presionar la tecla <Esc>.

```
                    A Y U D A

F2 CREAR DOCUMENTO:
   Permite escribir un documento nuevo.

F3 EDITAR DOCUMENTO:
   Permite Modificar el contenido de un
   documento, ya sea para borrar o adi-
   cionar texto al mismo. Especificando
   para ello  el nombre del documento
   deseado y en caso necesario el nombre
   del "drive" y directorio.

F4 SALVAR DOCUMENTO:
   Después de crear o modificar un docu-
   mento es posible guardarlo en disco.
   Siendo requerido para ello un nombre
   junto con la especificación de direc-
   torio y "drive" si el usuario lo ne-
   cesita.
```

Fig. V.4 Ayuda

Ayuda en la pantalla de edición

- Presionar <F1>.

Aparece en Pantalla:

AYUDA
F2 ESCRIBIR F3 GUARDAR F4 RECUPERAR F5 IMPRIMIR F6 DICCIONARIO F7 BUSQUEDA F8 SALIR
Elige Función

Fig. V.5 Ayuda

- Para ejecutar una función basta presionar la tecla correspondiente.

V.1.4 Escribir y modificar un documento

Iniciar un documento

La opción que permite escribir un documento nuevo sin importar el lugar de selección, Menú Principal o Pantalla de Edición es la función **F2.**

- Presionar <F2>.

Se presenta una pantalla nueva para escribir, con la línea de estado en la parte inferior derecha.

Esta línea de estado dará continuamente la siguiente información :

Pag : Número de página en la que se encuentra el cursor.

Ln : Número de línea en la que se encuentra el cursor.

Pos : Indicador de posición del cursor sobre la línea.

Alerta

Cuando se presiona F2 y existe un documento sin guardar se alerta al usuario de la situación.

```
No has guardado tu documento
¿ Deseas guardarlo ? <S/N> : S
```

Fig. V.6 Alerta para el documento en edición

ESCRIBIR EL DOCUMENTO

Separación de palabras

- Para marcar el final de un párrafo o línea se debe pulsar la tecla <Return> o <Intro>.
- Para insertar una línea vacía (en blanco) se pulsa de la misma forma, <Return> o <Intro>.

Insertar o sobre-escribir

Insertar: El procesador en el momento de arrancar se encuentra en el modo de inserción; los caracteres tecleados se colocan en la posición del cursor, respetando los ya existentes. Si hay texto a la derecha del cursor, se desplaza en forma continua y automática para reorganizarse.

Sobre-escribir: En este modo de funcionamiento los caracteres tecleados sustituyen a los que ya están escritos.

Para cambiar de Insertar a Sobre-escribir debe pulsarse la tecla <Ins> o <Insert>. Al hacerlo, PALABRA muestra en la parte inferior izquierda de la pantalla la palabra SOBRE-ESCRIBIR. Para regresar al estado original de inserción, se vuelve a pulsar la tecla <Ins>.

Borrar

- La tecla <Backspace> borra el carácter o los caracteres que se encuentran a la izquierda del cursor, independientemente del modo: Insertar o Sobre-escribir.

.- La tecla <Del> o <Supr> borra el carácter o los caracteres que se encuentran en la posición del cursor.

Movimientos del cursor a través del texto

< → >	mueve el cursor una posición hacia la derecha.
< ← >	mueve el cursor una posición hacia la izquierda.
< ↑ >	mueve el cursor una línea hacia arriba.
< ↓ >	mueve el cursor una línea hacia abajo.
<Re Pag> o <Pg Up>	mueve el cursor al principio de la página anterior.
<Av Pag> o <Pg Dwn>	mueve el cursor al principio de la página siguiente.
<Inicio>	mueve el cursor al inicio de la línea de texto.
<Fin>	mueve el cursor al final de la línea de texto.
<Control> + <Inicio>	mueve el cursor al inicio del documento.
<Control> + <Fin>	mueve el cursor al final del documento.

Escritura de caracteres especiales del español

En caso de contar con un teclado diferente al español o al latinoamericano, se ha construido una rutina específica que permite escribir los caracteres especiales del español que no se encuentran en dicho teclado.

- Para escribirlos, se presiona en forma simultánea las siguiente combinación de teclas.

Tabla V.1 Caracteres especiales del español

COMBINACION	CARACTER
TECLA + TECLA =	LETRA
Control + a	á
Control + e	é
Control + i	í
Control + o	ó
Control + u	ú
Control + n	ñ
Control + N	Ñ
Alt + u	ü
Alt + U	Ü
Alt + i	¿
Alt + a	¡

Ejemplo:

Escribir en la pantalla de edición la letra "a" con acento (á), auxiliándose de la rutina especial.

- Mantener presionada la tecla de **<Control>**, mientras se pulsa la letra **<a>**.

V.1.5 Guardar un documento

Después de escribir un documento es preciso guardar su contenido en algún medio magnético, como es el disco flexible o disco duro que se encuentra dentro de la computadora. Para ello basta con dar un nombre al documento escrito para guardarlo, y en caso necesario especificar la trayectoria completa con la asignación del manejador de disco y directorio.

- Pulsar <F3> para archivar el documento en curso.

  Muestra en pantalla:

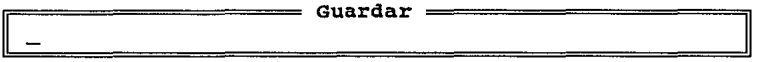


Fig. V.7 Guardar documento

- Escribir el nombre con el que se quiere archivar el documento y pulsar <Return>.

  (Al guardar el documento en disco, el texto continúa en la pantalla de edición. Para iniciar con otro documento se elige la opción de Escribir Documento o Recuperar documento.)

- Antes de dar un <Return> se puede cancelar la función con la tecla <Esc>.

Nombre del documento

Al momento de escribir el nombre del documento, es posible especificar el nombre del manejador de disco y directorio donde se desea guardar el documento. En caso de escribir únicamente el nombre, se toma por omisión el directorio y disco donde se invocó la entrada al procesador.

Ejemplo 1:

- Escribe el archivo TAREA en el disco A, subdirectorio ESPAÑOL.

Fig. V.8 Guardar documento

Ejemplo 2:

- Escribe el archivo TAREA en el disco y drive inicial. Justo en las posiciones que dio acceso al procesador de texto.

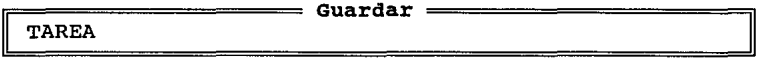


Fig. V.9 Guardar documento

Si el documento en edición ya tiene asignado un nombre, este se colocará automáticamente en la ventana de guardar para sólo aceptarlo con <Enter> o modificarlo.

Cuando se acepta el nombre para el documento se checa si existe otro documento con el mismo nombre. Si existe, se muestra lo siguiente:

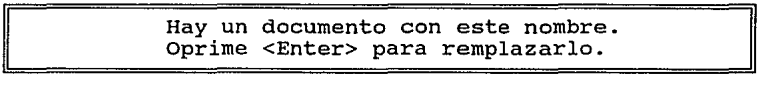


Fig. V.10 Alerta para el documento en edición

Como alternativa al mensaje, el programa permite cambiar el nombre o aceptar el remplazo del documento.

V.1.6 Recuperar un documento

Esta función fue elaborada para recuperar de disco los documentos previamente grabados en el medio mágnetico.

- Presionar <F4> para recuperar un documento.

Si la pantalla de edición contiene un documento sin guardar, se alertará de la situación al usuario antes de presentar el menú Recuperar.

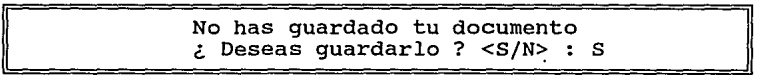


Fig. V.11 Alerta para el documento en edición

Si la respuesta es afirmativa se realiza el proceso de guardar y a continuación el de recuperar.

Menú Recuperar:

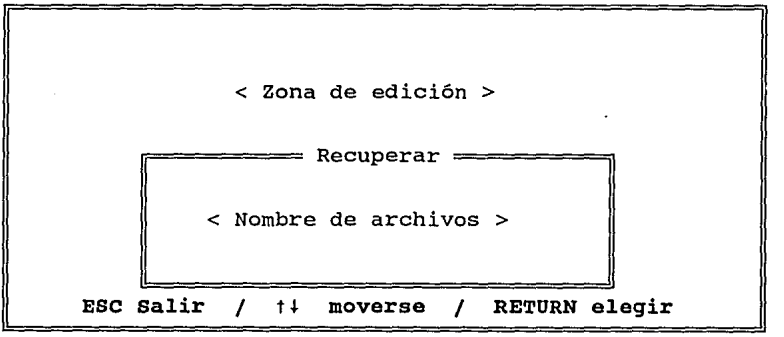


Fig. V.12 Recuperar un documento

En el menú recuperar se muestran los nombres de los archivos, su extensión y cantidad de bytes que ocupan. En el caso de directorios se muestran los nombres de estos y la nota <DIR>, en el caso de manejadores de disco aparece el nombre y la nota <DISCO>.

Ejemplo:

Si el acceso al procesador de texto se hizo desde C:\ESPAÑOL y la máquina contiene los discos A: y B:, aparecerá lo siguiente.

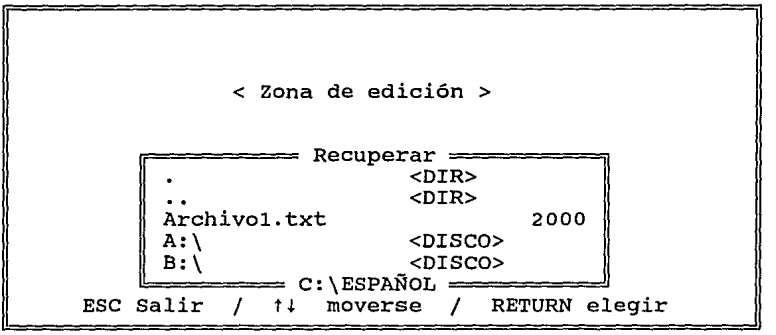


Fig. V.13 Recuperar un documento

Donde :

<Esc> Permite abandonar este modulo, sin la selección de un archivo.

↑↓ Podrá utilizar las teclas de movimiento de cursor hacia arriba y hacia abajo para recorrer la lista y seleccionar el disco, directorio o archivo. Si se requiere avanzar con mayor velocidad a través de la lista se harán uso de las teclas de avance de página o retroceso de página para avanzar siete líneas.

<Return> Al usar esta tecla, el sistema guarda la línea donde se encuentra el cursor, asumiendo que este es el nombre del archivo a recuperar o el directorio o disco a revisar.

V.1.7 Búsqueda

En algunas circunstancias será necesario buscar dentro del documento en edición, frases, signos, números o caracteres específicos.

La tecla que da el acceso a la función de búsqueda es **F7** y puede activarse desde el Menú General o desde la pantalla de edición.

- Presionar la tecla <F7> o posicionarse en la opción Búsqueda del Menú General y presionar <Return>.

Muestra en pantalla:

Fig. V.14 Búsqueda

- Escribir la cadena de caracteres a buscar y presionar <Return>.
- Antes de dar <Return> se puede cancelar la función con la tecla <Esc>.
- Después de aceptar la cadena indicar la dirección de busqueda.

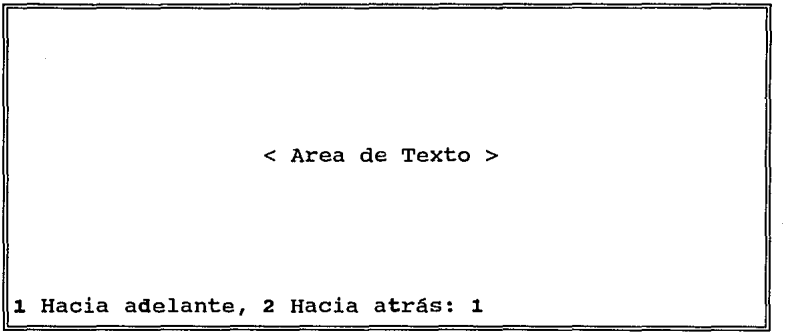


Fig. V.15 Dirección

Con **1**, **d**, **D** o <Return>, se selecciona hacia adelante.

Con **2**, **t** o **T** se selecciona hacia atrás.

- Si la cadena se encontró, indicar si continúa con la búsqueda.

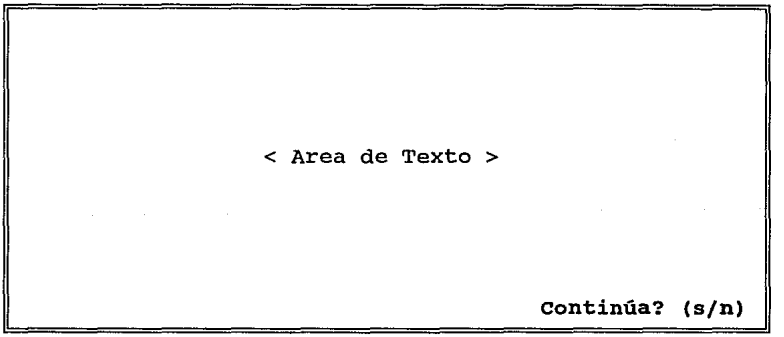


Fig. V.16 Continuación

Con **s** minúscula o mayúscula, se continua con la búsqueda de la misma cadena. Con **n** minúscula o mayúscula, se suspende el proceso.

- Si la cadena no se encontró, la función lo indica con un sonido y la frase "No encontrado".

V.1.8 Diccionario

En ocasiones, al escribir un texto, surgen algunas dudas, como: la forma correcta de utilizar una palabra, la ortografía o el significado de una palabra.

Aquí en esta sección se podrá hacer uso de un diccionario **Palabra-Significado,** con un acervo de aproximadamente 6,400 palabras con significado. Para poder tener acceso a éste existen dos formas diferentes de hacerlo :

- Consulta del Diccionario desde el menú principal
- Consulta del Diccionario dentro del texto.

V.1.8.1 Consulta del Diccionario desde el menú principal

Al consultar el diccionario desde este lugar nos permite checar el acervo por palabras especificas.

La forma de activar esta tarea será estando en el menú principal:

- Para activar el menú principal, pulsar **<Esc>**.

- Posicionarse en la figura de Diccionario de este menú (véase fig. V.17), con las teclas de movimiento de cursor y presionar <Return>, o presionar la tecla de función que se utiliza para llamar el diccionario: **<F6>**.

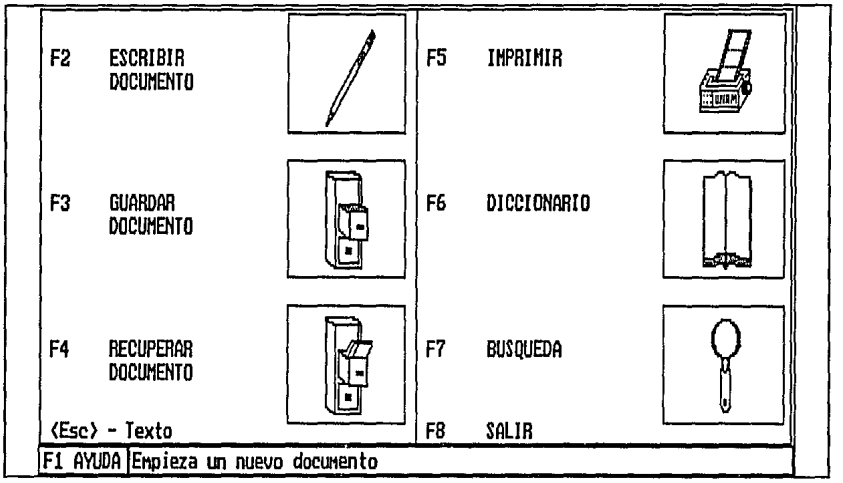


Fig. V.17 Menú Principal

- Aparecerá una ventana en la parte inferior de la pantalla la cual ocupara una tercera parte de esta pantalla y en la que se realizarán todas las tareas propias del diccionario. La primer subventana que veremos será la que indica que se está leyendo el diccionario (véase fig. V.18), esto sucederá si se utiliza por primera vez el diccionario en la sesión.

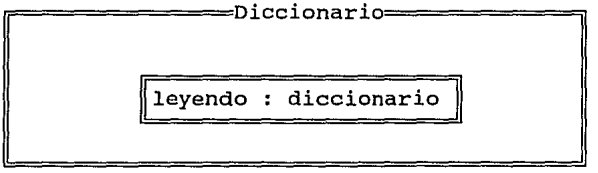


Fig. V.18 Ventana indicando la lectura de acervo de diccionario.

- Posteriormente desaparecerá la ventana de lectura del diccionario (en caso de existir) y se enviará un mensaje donde se pedirá escribir la palabra a consultar (véase fig. V.19). Proporcionar el término y presionar <Return>.

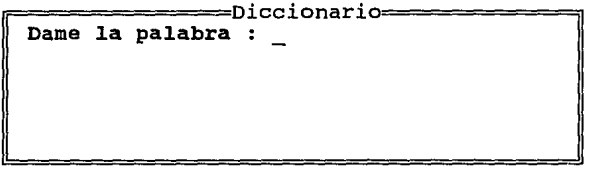


Fig. V.19 Ventana del diccionario llamado desde menú.

- Si se desea cancelar la función del diccionario, NO escribir nada y presionar <Return>, con ello aparecerá en la línea de mensajes un menú con dos opciones; <Esc> y <Return> (véase fig. V.20)

Donde :

<Esc> Es para salir del diccionario y regresar al texto.

<Return> Es para no salir y continuar con la consulta de palabras. (Esto por si se presiona un <Return> o <Enter> por equivocación).

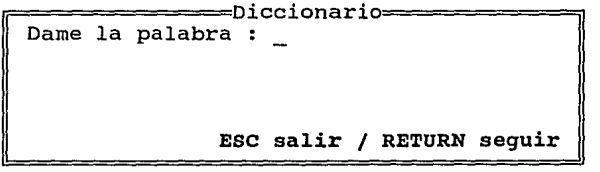


Fig. V.20 Ventana mostrando el menú de opción de cancelar o seguir.

- El término a consultar no debe contener ningún símbolo o signo que no sea perteneciente al alfabeto español, debido a que cuenta con un selector de letras. Por consiguiente, todo carácter no perteneciente al alfabeto español, no será considerado y se asumirá como un espacio en blanco, descartando el resto de la palabra.
- Se realizará la busqueda en el acervo para mostrar su significado, en caso de existir una mala acentuación en la palabra, se emitirá un aviso indicando el error de acentuación.

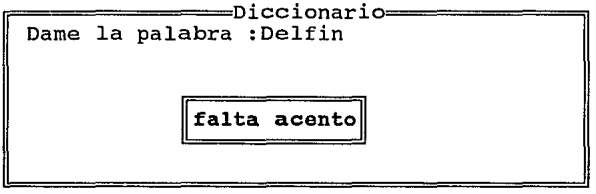


Fig. V.21 Ventana visualizando un error de acentuación.

- Posteriormente al desaparecer la ventana de mensajes, se muestra sobre la línea tres la palabra correcta. Permitiendo con ello, analizar y asimilar el error.

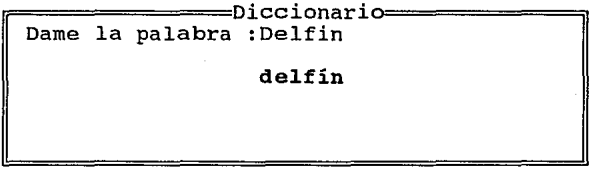


Fig. V.22 Ventana con la palabra correcta.

- Unos segundos después aparece el término consultado con su significado. Este permanecerá en pantalla mientras no se presione alguna tecla.

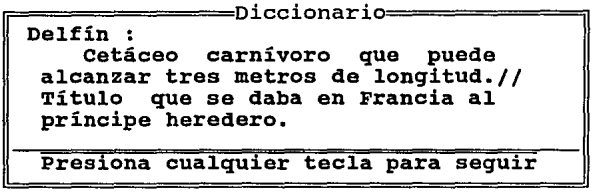


Fig. V.23 Ventana del diccionario con el significado de la palabra.

- En caso de que el término no se encuentre, aparece un mensaje donde se indica que la palabra no existe en el diccionario.

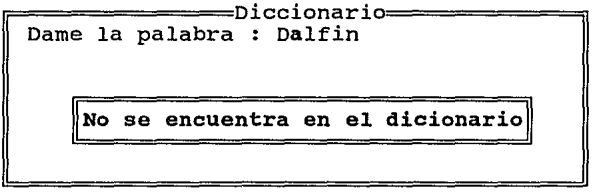


Fig. V.24 Ventana que indica la ausencia de la palabra.

- Después de unos segundos se invita a consultar la lista de términos contenida en el diccionario (véase fig. V.25). Esto con el fin de checar que la palabra proporcionada se encuentra correcta o incorrectamente escrita y al mismo tiempo permitir explorar el repertorio del diccionario.

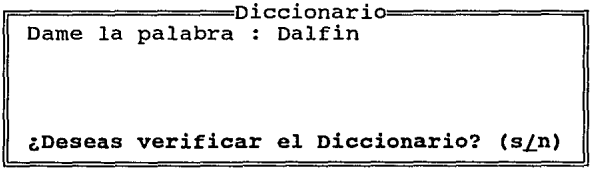


Fig. V.25 Ventana que invita a revisar el diccionario.

- Si la respuesta es negativa entonces se pasa a otra ventana que muestra un menú con dos opciones <Esc> y <Return> (véase fig. V.26).

Donde :

<Esc> Es para salir del diccionario y regresar al texto.

<Return> Es para no salir y continuar con la consulta de palabras. (Esto por si se presiona un <Return> o <Enter> por equivocación).

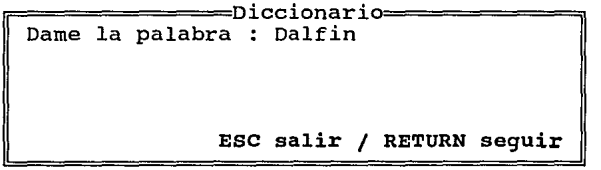


Fig. V.26 Ventana que muestra el menú de opción de cancelar o seguir.

- Si la respuesta es afirmativa se pasa a una ventana que muestra una lista ordenada de términos. El cursor es colocado sobre la lista en la palabra más cercana al término solicitado.

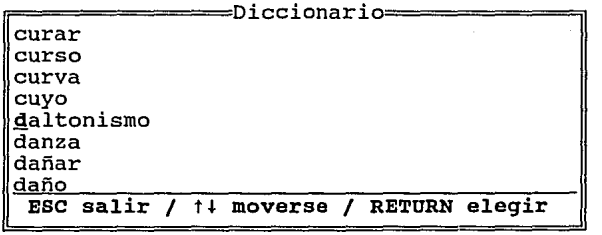


Fig. V.27 Ventana que muestra la lista de palabras.

- La manera de usar esta sección es la siguiente:

a) Salir

Para salir de esta opción del diccionario se presiona la tecla <Esc> que nos regresa a la ventana de consulta del diccionario (véase fig. V.25).

b) Recorrer

- Por palabra.

Si se desea inspeccionar la lista término por término en forma ascendente o descendente, se usan las teclas de movimiento de cursor: hacia arriba o hacia abajo, según sea el caso.

- Por ventanas.

Si se quiere realizar el recorrido de la lista rápidamente, se usan las teclas de movimiento de páginas: (Re Pág) retroceso de página y (Av Pág) avance de página. Esto provoca un avance en la lista de ocho palabras en ocho palabras, en ambos sentidos.

c) Seleccionar

Si encontramos un término que nos interese, colocamos el cursor sobre el mismo y presionamos la tecla <Return> para desplegar el significado.

V.1.8.2 Consulta del Diccionario dentro del texto

Con el fin de que se pueda observar el significado o significados de palabras y tomar la que más nos convenga o acomode a la idea que se esta escribiendo, se ha habilitado la consulta del diccionario desde el texto, para checar palabras especificas contenidas en él.

La forma de activar esta tarea es:

- Estar en el texto.

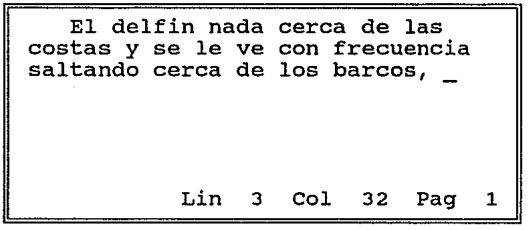


Fig. V.28 Ventana principal del procesador de texto.

- Colocarnos con las teclas de movimiento de cursor en la palabra a consultar y presionar la tecla de función de llamado al diccionario (**F6**), sin importar en que parte de la palabra se coloque el cursor.

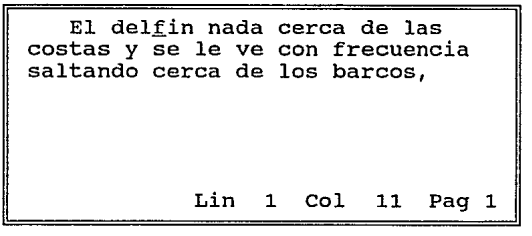


Fig. V.29 Ventana principal del procesador de texto.

- Aparecerá un ventana en la parte inferior de la pantalla que ocupará una tercera parte de ésta, en la cual se realizarán todas las tareas propias del diccionario. El término se coloca en negrillas para poderlo destacar del texto, y se recorre la palabra junto con el texto a

la parte superior de la pantalla en caso de ocupar el área donde se coloca la ventana de diccionario.

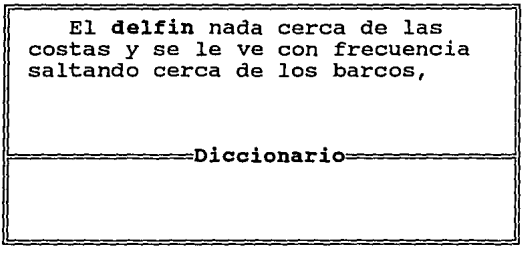


Fig. V.30 Ventana principal del procesador de texto con la ventana del diccionario.

- El primer mensaje que vemos es el que indica la lectura del diccionario (véase fig. V.27) (Esto sucede al utilizar el diccionario por primera vez en la sesión).

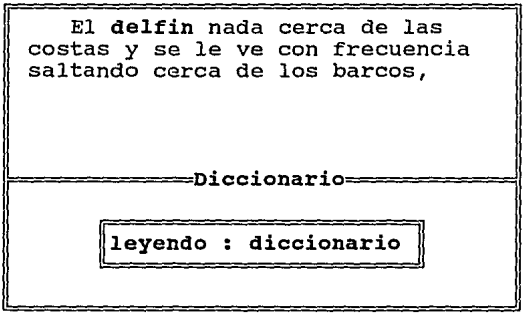


Fig. V.31 Ventana indicando la lectura del diccionario.

- Posteriormente desaparece la ventana de lectura del diccionario (en caso de existir) y es enviado un mensaje que indicará la palabra seleccionada.

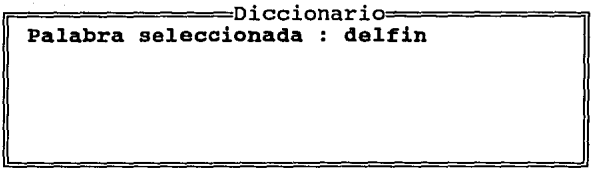


Fig. V.32 Ventana del diccionario llamado desde texto.

- La palabra a consultar no debe contener ningún símbolo o signo que no sea perteneciente al alfabeto español, debido a que cuenta con un selector de letras. Por consiguiente todo carácter no pertenecientes al alfabeto español, no será considerado y se asumirá como un espacio en blanco, descartando el resto de la palabra.
- Se realizará la busqueda en el acervo para mostrar su significado, en caso de existir una acentuación incorrecta en la palabra, se emitirá un aviso indicando el error de acentuación.

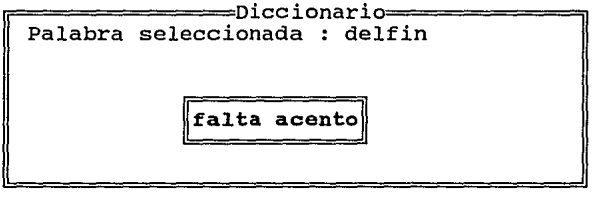


Fig. V.33 Ventana mostrando el mensaje de error de acentuación.

- Posteriormente, al desaparecer la ventana de mensajes, se muestra sobre la línea tres la palabra correcta. Permitiendo con ello, analizar y asimilar el error.

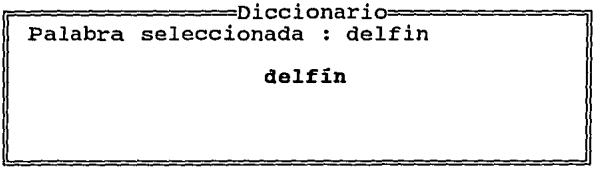


Fig. V.34 Ventana mostrando la palabra correcta.

- Unos segundos después aparece el concepto consultado con su significado. Este permanecerá en pantalla mientras no se presione alguna tecla.

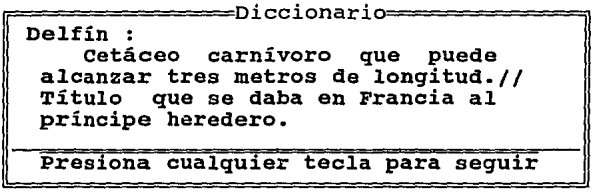


Fig. V.35 Ventana del diccionario con el significado de la palabra.

- En caso de que el término no se encuentre, aparece un mensaje donde se indica que la palabra no existe en el diccionario.

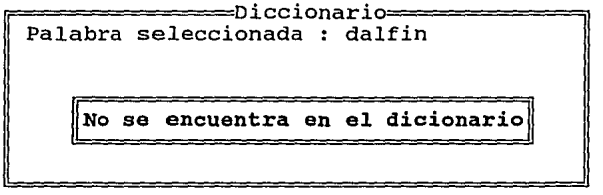


Fig. V.37 Ventana indicando la ausencia de la palabra.

- Después de unos segundos se invita a consultar la lista de términos contenida en el diccionario (véase fig. V.37). Esto con el fin de verificar que la palabra proporcionada se encuentra correcta o incorrectamente escrita y al mismo tiempo permitir explorar el repertorio del diccionario.

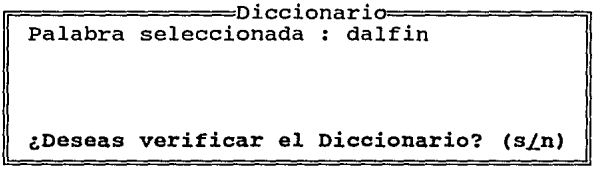


Fig. V.37 Ventana invitando a revisar el diccionario.

- Si la respuesta es negativa abandona la opción de diccionario y regresa al texto en el punto donde se llamó éste.
- Si la respuesta es afirmativa se pasa a una ventana que muestra una lista ordenada de terminos. El cursor es colocado sobre la lista en la palabra más cercana al término solicitado.

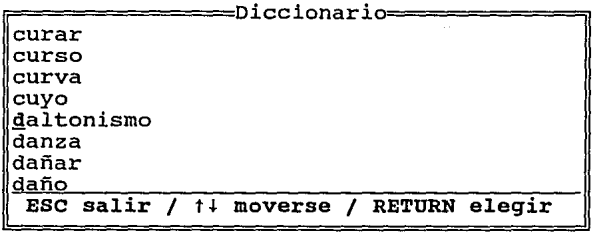


Fig. V.38 Ventana mostrando la lista de palabras.

- La manera de usar esta sección es la siguiente:

a) Salir

Para salir de esta opción del diccionario se presiona la tecla <Esc> que nos regresa a la ventana de consulta diccionario (véase fig. V.38).

b) Recorrer

- Por palabra.

Si se desea inspeccionar la lista palabra por palabra en forma ascendente o descendente, se usan las teclas de movimiento de cursor: hacia arriba o hacia abajo, según sea el caso.

- Por ventanas.

Si se quiere realizar el recorrido de la lista rápidamente, se usan las teclas de movimiento de páginas: (Re Pág) retroceso de página y (Av Pág) avance de página. Esto provoca un avance en la lista de ocho palabras en ocho palabras, en ambos sentidos.

c) Seleccionar

Si encontramos una palabra que nos interese, se coloca el cursor sobre el mismo y presionamos la tecla <Return> para desplegar el significado.

V.1.9 Impresión

La impresión es la etapa final en la escritura de un documento. A través de la tecla <F5> se logra imprimir el contenido de un archivo o el texto que se halla en la pantalla de edición.

Existen dos métodos para imprimir un documento en PALABRA:

. Impresión desde Menú Principal.

. Impresión desde Pantalla de Edición.

V.1.9.1 Impresión desde menú principal

Este método imprime el contenido de algún archivo en disco.

- Presionar la tecla <F5> de impresión

Muestra en pantalla:

```
Imprimir
_
```

- Escribir el nombre del documento a imprimir y pulsar la tecla <Return>.

Nombre del documento

Al momento de escribir el nombre del documento, es posible especificar el nombre del manejador de disco y directorio que contienen al documento. En caso de escribir únicamente el nombre, se toma el directorio y disco donde se invocó al procesador.

Ejemplo 1:

- Imprimir el archivo TAREA, localizado en el disco A, subdirectorio ESPAÑOL.

```
Imprimir
A:\ESPAÑOL\TAREA
```

Ejemplo 2:

- Imprimir el archivo TAREA, localizado en el disco y drive original (posición que dio acceso al procesador de texto).

```
Imprimir
TAREA
```

- Después de escribir el nombre del archivo a imprimir, se muestra en pantalla lo siguiente:

```
¿ QUIERES LETRA DE CALIDAD < S \ N > ? _
```

- Responder a la letra de calidad con:

  S para letra de calidad.
  N para calidad normal.

- En caso de cancelación de impresión, presionar la tecla <Esc>.

V.1.9.2 Impresión desde la pantalla de edición

En la Pantalla de Edición, la función F5 imprime el documento ubicado en la memoria de la computadora. Es decir, imprime el texto que en ese instante se corrige o edita.

- Presionar la tecla <F5> de impresión.

  Muestra en pantalla:

```
¿ QUIERES LETRA DE CALIDAD < S \ N > ? _
```

- Responder a la letra de calidad con:

  S para letra de calidad.
  N para calidad normal.

- En caso de cancelación de impresión, presionar la tecla <Esc>.

V.1.10 Actualización del Diccionario

Al usar el procesador de texto y particularmente la opción de consulta del diccionario Palabra-Significado, existirá la necesidad de agregar y/o modificar palabras y/o significados a los ya

existentes. Este tipo de cambios se hará a través del programa ACTUAL(), el cual consta de dos módulos :

1.- El que nos permita adicionar una palabra con su respectivo significado; y

2.- el módulo que nos permita modificar el contenido del significado de una palabra.

Al accionar este programa aparecerán dos ventanas en la pantalla. Una de ellas, la que se encuentra en la parte superior de la pantalla, se llama "AVISO" y la otra ventana que se encuentra en la parte inferior de la pantalla se llama "ACTUALIZA" .

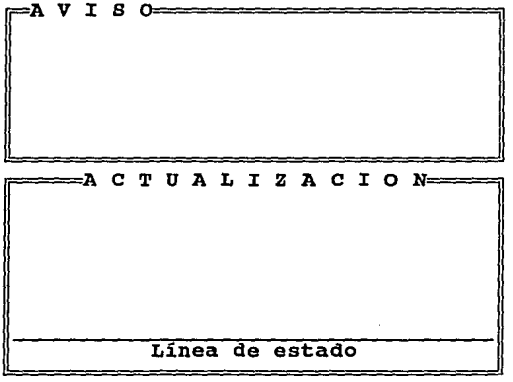


Fig. V.39 Ventanas del programa actual

En la ventana "AVISO" se desplegará la leyenda: "Este programa fue realizado para actualizar el diccionario, con nuevas palabras y nuevos significados; para ello se deberán usar letras minúsculas, sin olvidar la acentuación", con el propósito de que el contenido del diccionario Palabra-Significado sea correctamente escrito al momento de ser capturada la nueva información (véase fig. V.40).

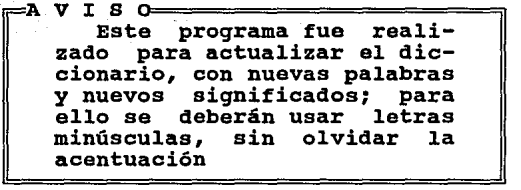


Fig. V.40 Ventana AVISO

En la ventana "ACTUALIZA", se realizará todo el proceso de actualización. Dicho trabajo empezará mostrando una subventana con el mensaje: "Leyendo el diccionario" para indicarnos que está almacenando en memoria el índice de términos del diccionario.

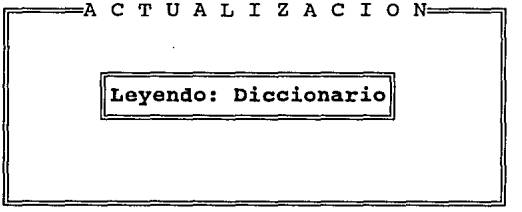


Fig. V.41 Ventana indicando la lectura del índice del diccionario

Cabe hacer notar que en caso de haber algún problema con el espacio de la memoria o la lectura del índice del diccionario, será desplegado un mensaje, por medio de una ventana con letras blancas centellantes para indicar la falla de que se trate; y poco después dará por terminada la sesión y saldrá del sistema.

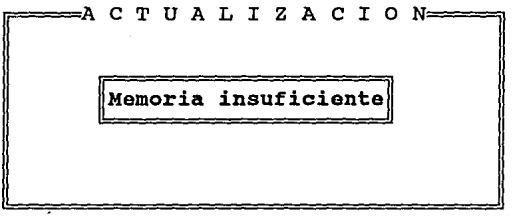


Fig. V.42 Ventana indicando error en el almacenamiento del índice

Una vez terminada la lectura del diccionario, aparecerá en la primera línea, el mensaje "Dame la palabra:" (véase fig. V.43); donde deberá proporcionarse la palabra a modificar o agregar. Una vez suministrado el término, el sistema examinará dicho término, analizando que la palabra esté compuesta por caracteres en español y sin ningún tipo de signos o números; Si hubiera algún signo diferente a los caracteres en español, cortará dicho término en el espacio en donde aparezcan los signos mencionados.

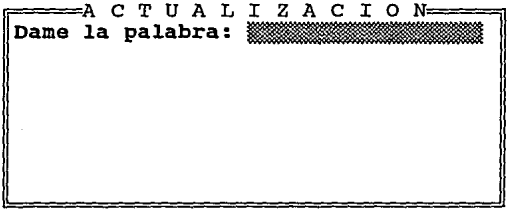


Fig. V.43 Ventana solicitando el término a agregar y/o modificar

Después cambiará el mensaje "Dame la palabra:" por el mensaje "La palabra:"; agregando el término analizado. Y a su vez preguntará si el término proporcionado es correcto,

mediante el mensaje: "¿Es correcta? (s/n)" (véase fig. V.44). Si la respuesta es negativa pedirá de nuevo una palabra a modifiacar o agregar.

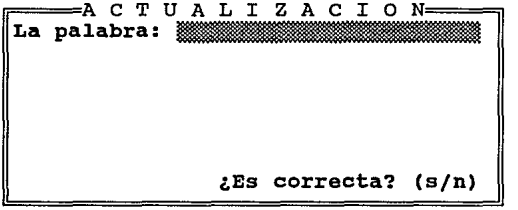


Fig. V.44 Ventana mostrando el término analizado y solicitando su validez

Si la respuesta es afirmativa, se checará la existencia del término en el acervo. Mientras se realiza la búsqueda, aparecerá a la izquierda de la línea de estado con letras blancas y centellantes, el mensaje "Espera", el cual permanecerá centellando hasta terminar la búsqueda.

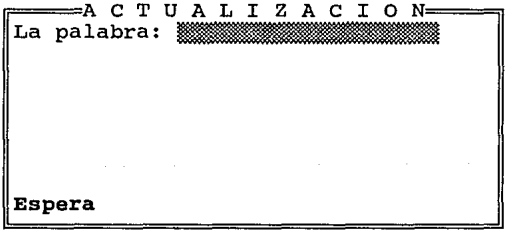


Fig. V.45 Ventana indicando esperar mientras se realiza la búsqueda del término

Si el término no existe en el diccionario, el sistema desplegará una ventana con letras blancas y centellantes el mensaje: "No se encuentra en el diccionario".

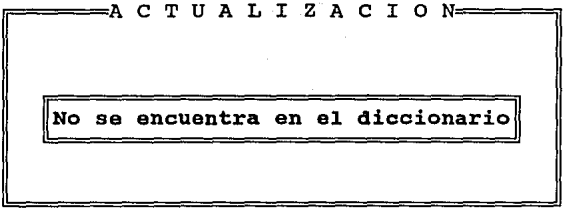


Fig. V.46 Ventana indicando la no existencia del término en el acervo

Después de un momento, preguntará si se desea adicionar dicho término en el diccionario, mediante el mensaje "¿Deseas agregarla al diccionario? (s/n)".

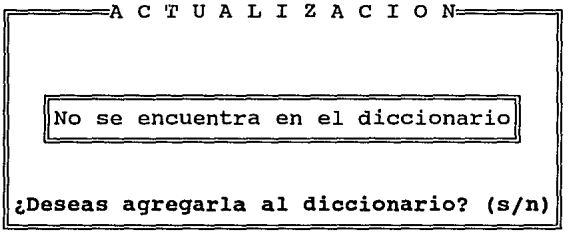


Fig. V.47 Ventana preguntando la adición del término al acervo

Si se responde negativamente, limpiará la ventana y aparecerá el mensaje:"¿Alguna otra modificación? (s/n)" (véase fig. V.48); si se contesta "n", se da por terminada la tarea de actualización y sale del sistema.

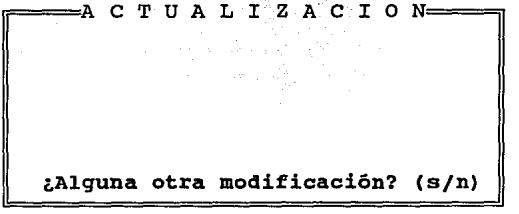


Fig. V.48 Ventana preguntando si se desean realizar más actualizaciones

Si se responde positivamente (s), regresa al inicio de la tarea, solicitando la nueva palabra.

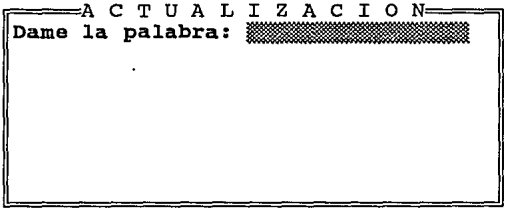


Fig. V.49 Ventana solocitando el término a agregar y/o modificar

Cuando el sistema pregunta si se desea anexar el término al diccionario y se contesta afirmativamente, se mostrará una ventana con la leyenda "RECUERDA: Debes acentuar las palabras correctamente"; enseguida aparecerá la palabra destacada y la leyenda: "Tendrá como significado:" (véase fig. V.50), para que se pueda colocar inmediatamente el significado de la palabra elegida, mismo que puede tener como mínimo cuatro caracteres y como máximo doscientos ochenta.

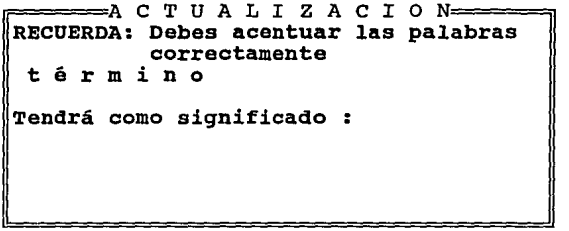


Fig. V.50 Ventana presentando el término y el lugar donde se colocará el significado

Cabe señalar que el programa cuenta con un pequeño editor que posibilita la inserción y borrado de caracteres, movimiento del cursor hacia arriba, hacia abajo y hacia los lados dentro del espacio proporcionado para capturar el significado de las palabras (véase fig. V.49). Dicho editor se encarga de indicar mediante un "bip" el exceso de caracteres en el momento de proporcionar el significado de los vocablos.

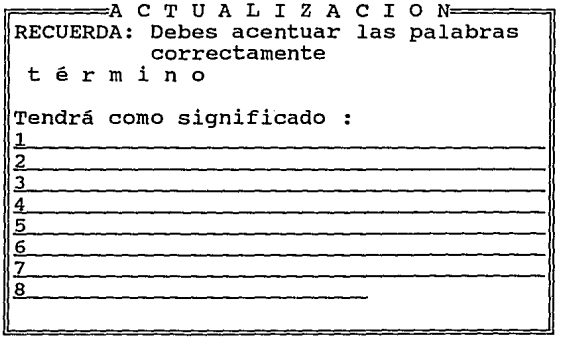


Fig. V.51 Ventana en la que se exhiben las ocho líneas donde se modifica el significado

Otra función del editor, consiste en evitar salir de la captura del significado si éste contiene menos caracteres del mínimo requerido y enviará en la línea de estado, el mensaje: "Tiene que ser mayor de cuatro caracteres".

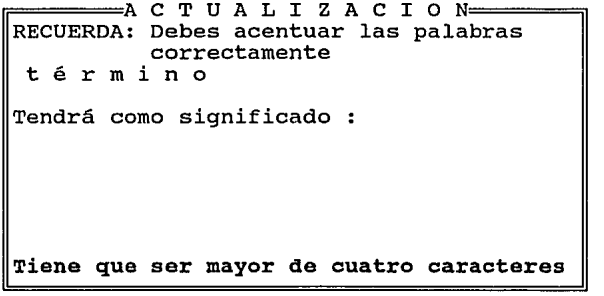


Fig. V.52 Ventana indicando el mínimo requerido para significado

Una vez escrito el significado de la palabra, el sistema enviará en la línea de estado, el mensaje: "¿Es correcto el significado? (s/n)" (véase fig. V.53). Si la respuesta a esta pregunta es "s", actualizará el diccionario, de lo contrario, ignorará la adición y regresará para pedir un nuevo término.

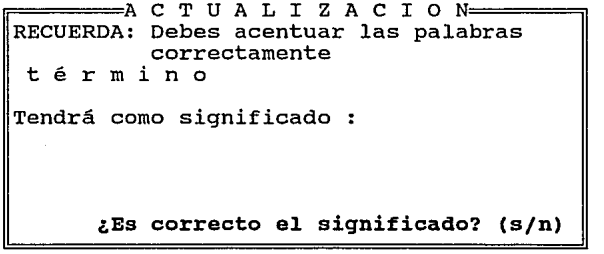


Fig. V.53 Ventana solicitando la validez del significado

Ahora bien, si la palabra solicitada existe en el diccionario, aparecerá el término junto con su significado y después un momento, preguntará en la línea de estado si se desea cambiar el significado de dicha palabra.

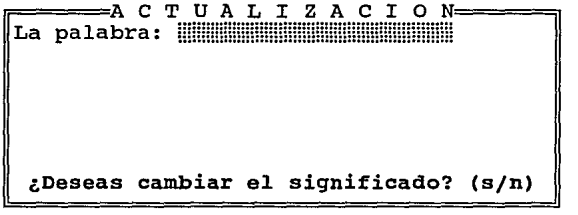


Fig. V.54 Ventana interrogando si se desea modificar el significado

Si a esta nueva pregunta se contesta afirmativamente, se mostrará a partir de la línea uno de la ventana "Actualiza", la leyenda "RECUERDA: Debes acentuar los términos correctamente"; donde la palabra "RECUERDA" aparecerá resaltada. Enseguida presentará en forma destacada el vocablo que se quiere modificar. En la siguiente línea mostrará la leyenda: "Tendrá como significado:" para que se pueda colocar en el espacio asignado, el significado de la palabra que se va a modificar (véase fig. V.55), es importante no olvidar que se tienen doscientos ochenta caracteres máximo y cuatro mínimos y que el programa cuenta con un pequeño editor que posibilita la inserción y borrado de caracteres, movimiento del cursor hacia arriba, hacia abajo y hacia los lados dentro del espacio proporcionado.

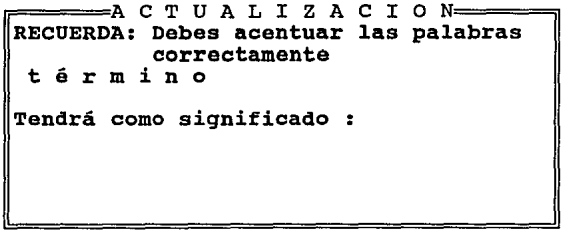


Fig. V.55 Ventana mostrando el término y el significado a modificar

Una de las funciones del editor arriba mencionado, consiste en indicar mediante un "bip" el exceso de caracteres en el momento de proporcionar el significado de los vocablos. Otra función del editor, consiste en evitar salir de la captura del significado si éste contiene menos caracteres del mínimo requerido enviando en la línea de estado, el mensaje: "Tiene que ser mayor de cuatro caracteres".

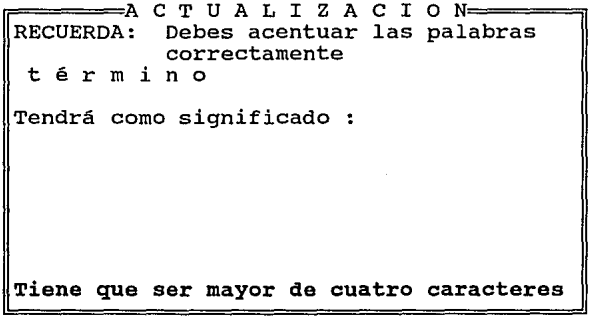


Fig. V.56 Ventana indicando que el significado debe ser de más de cuatro caracteres

Una vez escrito el significado del término, el sistema validará dicho significado enviando en la línea de estado, el siguiente mensaje: "¿Es correcto el significado? (s/n)" (véase fig. V.55). Si la respuesta a esta pregunta es "s", actualizará el diccionario, de lo contrario, ignorará la adición y regresará a solicitar un nuevo término.

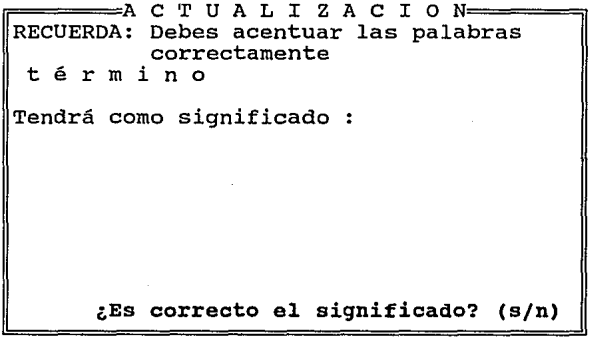


Fig. V.57 Ventana preguntando si el significado es el correcto

Si cuando preguntó si se deseaba cambiar el significado, se contestó negativamente, aparecerá en la línea de estado, el mensaje: "¿Alguna otra modificación? (s/n)".

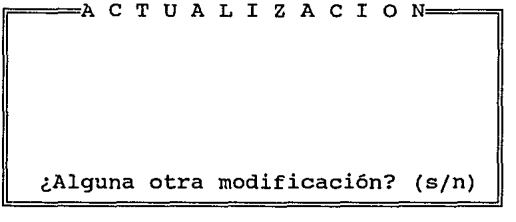


Fig. V.58 Ventana interrogando si se desean realizar más actualizaciones

Si la respuesta es "s", pedirá el nuevo término a modificar y/o agregar, continuando la tarea de acuerdo a lo descrito anteriormente, de lo contrario se dará por concluida la actualización.

Antes de salir del sistema, a la izquierda de la línea de estado, en letras centellantes, aparecerá el mensaje: "Actualizando" para indicar que se está haciendo la actualización del índice del diccionario y que se está grabando en uno de los archivos del diccionario, la(s) palabra(s) agregada(s) y/o modificada(s).

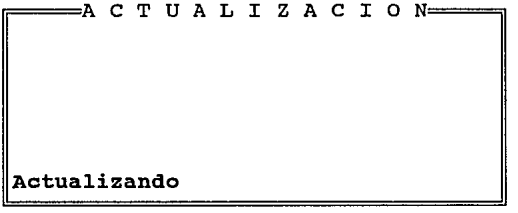


Fig. V.59 Ventana indicando la actualización del diccionario y del índice.

Una vez actualizado el índice con la nueva localización de las palabras que se incluyeron y/o modificarón en el diccionario, se dará por concluida la tarea y se saldrá del sistema.

V.2 Manual técnico

- Requerimientos
- Instalación
- Programa

V.2.1 Requerimientos

Para el correcto funcionamiento del Procesador de Texto Infantil, se requiere:

. Un equipo PC XT\AT o superior.

. 512 Kbytes de memoria RAM como mínimo.

. Una unidad de disco flexible o duro.

. Sistema Operativo DOS

V.2.2 Instalación

Los archivos de programa del procesador de texto para uso infantil, están contenidos en un disco flexible de 3 1/2 pulgadas, doble densidad (720KB).

Disco Flexible

Si la computadora no tiene disco duro, puede trabajar directamente con un disco flexible de 3 1/2 pulgadas, doble densidad. Para ello es recomendable hacer un respaldo o copia del disco original y trabajar con la copia.

-Comando del sistema operativo para obtener una copia:

A:\>DISKCOPY A: B:

donde A: será el disco original y B: la copia.

Disco Duro

Si se desea colocar el procesador en el disco duro, es necesario ejecutar el programa de instalación que viene con el disco de programa.

- Colocar el disco de programa en la unidad de 3 1/2 pulgadas (A: o B:) y ejecutar *instala*.

A:\>INSTALA

Al ejecutarlo, se piden los siguientes datos:

1) La unidad de disco donde se encuentra el disco de programa, A: o B:.

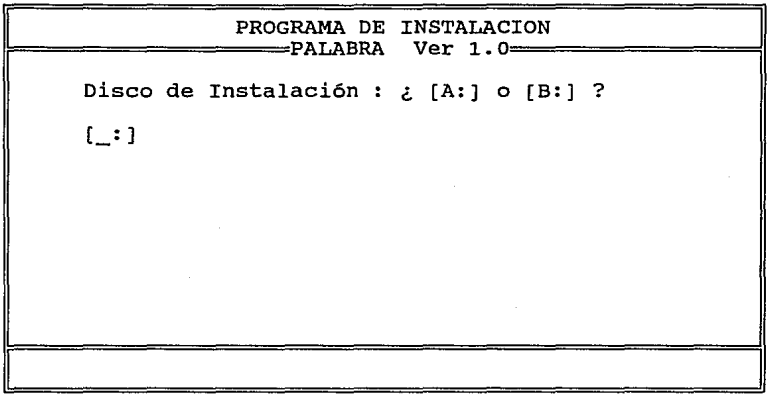


Fig. V.58 Solicitud de disco de instalación

2) A continuación se pide el lugar de instalación para el procesador de texto: unidad de disco duro y directorio deseado.

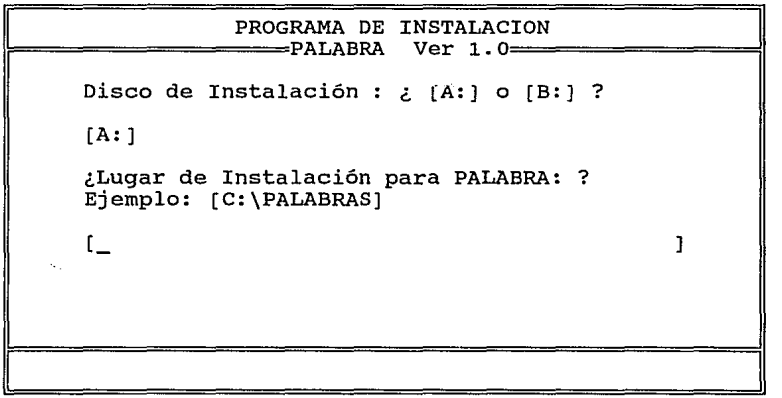


Fig. V.59 Solicitud de lugar de instalación

Si el directorio especificado no existe, se indica con un mensaje la creación del directorio en el disco duro.

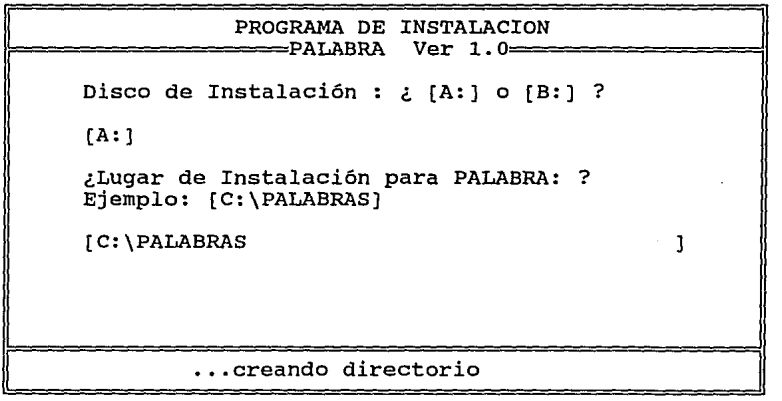


Fig. V.60 Mensaje de creación de directorio

Conforme se instala, se presenta en pantalla cada archivo copiado al disco duro. Para indicar el término de la instalación se manda el mensaje:

"Instalación Completa"

...presiona cualquier <tecla> para salir.

Nota: Los datos que se requieren en los puntos 1) y 2) en caso de ser erroneos, podrán ser corregidos con ayuda de los mensajes de instalación.

- Drive sin disco.
- Trayectoria mal especificada.
- Disco lleno.

V.2.3 Programa

Esta sección en particular va dirigida a los analistas que desean conocer en mayor medida, las estructuras del Procesador de Texto.

Presenta a las subrutinas más importantes del sistema en su totalidad.

V.2.3.1 Subrutina control()

Dentro de todo el programa o sistema, a control() se le considera como una de las subrutinas más trascendentales, al mediar entre la entrada y la salida.

La subrutina en cuestión se activa por el llamado de la rutina teclado() que sensa la presión de alguna tecla. Al activarla recibe los códigos ascii y scan que permiten dirigir la acción o función del programa.

Utiliza en su disposición estructural a la instrucción "switch()" del Lenguaje.

```
/*************************************************************************/
/* Subrutina que lleva el control de llamadas al conjunto de funciones del */
/* procesador, así como el borrado y movimiento entre líneas, páginas y    */
/* columnas.                                                               */
/*************************************************************************/
 control(yy,xx,dato,ascii)
  char *yy, *xx;
  unsigned char dato, *ascii;
    { char x,y,lon;
      char *nombre, *linea;
      char *s_safe[];
      int k=2;      /* Bandera para controlar el uso de esta rutina */
      int l,cont,linup;
    x = *xx;
    y = *yy;
   while (k == 2)
   switch (dato) {
     case 0x1c : { if (tex[bliny][bcolx] == 0x10)      /* RTN */
           { tex[bliny][bcolx] = '\r'; /* Elimina caracter de lg */
             s_pchar(y,38,0x20,atri); /* Quita en screen 0x10 */
           }
             else
           { if (bcolx == 0 && tex[bliny-1][36] == 0x10)
               tex[bliny-1][36] = '\0'; /* Con RTN en inicio de línea se corta lg */
             if (y == LIMREN)
              s_scroll(0,1,INIREN,INICOL,LIMREN,LIMCOL+1);
             else
              ++y;
             x = INICOL;
             escrilin(y);
             ++liny;
             if (liny > LIMPAG)
             { liny = 1; ++pag; }
             s_colinp(y,x,liny,pag);
             lon = loncar(tex[bliny]);
             if (tex[bliny-1][bcolx] != '\r' && tex[bliny][0] != '\0')
              if (tex[bliny][lon] == 0x10 && tex[bliny-1][0] != '\r')
               sube_lg(LIMCOL-lon-1,0,y);
              else
```

```
          sube_p(LIMCOL-lon-2,1,y);
          s_posyx(y,x);
        }
          k=1;
          break;
        }
case 0x52 : { ins = !ins; /* INSERCION / SOBRE-ESCRITURA */
          f1esc(ins);
          s_posyx(y,x);
          k = 1;
          break;
        }

case 0x0e : { borcar (bcolx,&y,&x,0);          /* <-- BACKSPACE */
          k = 1;
          break;
        }
case 0x53 : if (*ascii != 0) /* DELETE */
          k = 0;
        else
        { cont = 0;
          if (tex[bliny][bcolx] != '\r')
        { if (tex[bliny][bcolx] == '\0')
            cont = 1;       /* Control:Cursor al fin de linea */
          ++x;
        }
          else
        { cont = 1;   /* Indicación de borrar blanco entre las dos lineas */
          if (bcolx == 0)
           { ++bliny;  /* Mantener cursor en la misma posición */
             ++liny;   /* al Borrar linea (solo con RTN)      */
             if (liny>52)
              { liny = 1;++pag;}
             ++y;
           }
        }
        if (tex[bliny][bcolx] == 0x10) /* Manejo de palabras grandes */
         { cont = 0;
           ++bliny;
           ++y;
           l = 3;
           borcar(1,&y,&l,1);
           if (tex[bliny+1][0] == 0x20)
            { tex[bliny][36] = '\0'; /* Elimina marca de palabra grande */
              s_ps(y-1,2,tex[bliny].atri);
              s_pchar(y-1,38,0x20,atri);
            }
           s_colinp(--y,--x,liny,pag);
         }
        else
         borcar(bcolx+1,&y,&x,1);
          if (cont && tex[bliny][bcolx] == 0x20)        /* Borra blanco */
        { lon = strlen(tex[bliny]);
          movmem(&tex[bliny][bcolx+1],&tex[bliny][bcolx],lon-bcolx);
          s_pl(bliny,bliny,y);
          s_colinp(y,x,liny,pag);
        }
          k = 1;
        }
        break;
case 0x4b : if (*ascii != 0) /* IZQUIERDA */
          k = 0;
```

```
        else
        {if ( x>INICOL  )
         --x;
        else
          if (bliny > 1)
             { --liny;
            if (liny == 0)
             { liny = (pag == 1) ? 1 : 52;
               if (pag > 1)
                 --pag;
             }
            if (y == INIREN)
             { s_scroll(1,1,INIREN,INICOL,LIMREN,LIMCOL+1);
               s_ps(INIREN,INICOL,tex[bliny-1],0x17);
               ++y;
             }
            --y;
            lon = loncar(tex[bliny-1]);
            x = lon + 2;  /* Posición a fin de linea */
             }
         s_colinp(y,x,liny,pag);
         k=1;
        }
        break;
case 0x4d : if (*ascii != 0) /* DERECHA */
         k = 0;
        else
        { if ( bliny <= bmax )
        { lon = loncar(tex[bliny]);
          if (x > (lon+1))
           { if (bliny != bmax)
              { x = INICOL;
              ++y;
              ++liny;
             if (liny == (LIMPAG+1))
              { liny = 1;
                ++pag;
             }
              if (y > LIMREN)
             { --y;
               pantup(y,x);
               s_posyx(y,x);
             }
              }
           }
          else
            ++x;
          s_colinp(y,x,liny,pag);
        }
          k=1;
        }
        break;
case 0x48 : if (*ascii != 0)   /* ARRIBA */
         k = 0;
        else
        { if( y>INIREN )   /* Solo mueve cursor */
         { --y;
           --liny;
           if (liny==0) {liny=52;--pag;}
           s_colinp(y,x,liny,pag);
         }
        else          /* Scroll, escribe linea al inicio */
```

```
        {if ( pag >= 1 )
         { --liny;
           if ( liny == 0)
             { --pag;
            if (pag > 0)
             { liny = 52;
               s_colinp(y,x,liny,pag);
               s_scroll(1,1,INIREN,INICOL,LIMREN,LIMCOL + 1);
               s_ps(INIREN,INICOL,tex[bliny],0x17);
               s_posyx(y,x);
             }
            else {pag=1; liny=1;}
             }
            else
             {
           s_colinp(y,x,liny,pag);
           s_scroll(1,1,INIREN,INICOL,LIMREN,LIMCOL + 1);
           s_ps(INIREN,INICOL,tex[bliny],0x17);
           s_posyx(y,x);
            }
         }
        }
        lon = loncar(tex[bliny]);
        if (x>(lon+1))                  /* Chequeo de posición de cursor-fin de linea */
         x = lon + 2;
        s_colinp(y,x,liny,pag);
        k = 1;
       }
       break;
case 0x50 : if (*ascii != 0)  /* ABAJO */
        k = 0;
       else
       {
        if ( bliny < bmax)
         {
        if ( y >= LIMREN )
         { ++liny;
           if (liny == (LIMPAG+1) )
            { liny = 1;
              ++pag;
            }
           pantup(y,x);
           s_posyx(y,x);
         }
        else
         { y++;
           liny++;
           if (liny == (LIMPAG+1) )
            { liny = 1;
              ++pag;
            }
           s_colinp(y,x,liny,pag);
         }
        lon = loncar(tex[bliny]);
        if (x>(lon+1))                  /* Chequeo de posición de cursor-fin de linea */
         x = lon + 2;
        s_colinp(y,x,liny,pag);
         }
        k=1;
       }
       break;
case 0x84 : if (bmax > 1)  /* Ctrl-PG UP (Inicio de Texto) */
```

```
        { s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL+1);
        s_pl(1,22,1);
        x = INICOL;
        y = INIREN;
        liny = 1; pag = 1;
        s_colinp(y,x,liny,pag);
        }
        k=1;
        break;
case 0x49 : if (*ascii != 0)    /* PG UP */
        k = 0;
        else
        { if (bmax > 1)
        { s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL+1);
          if (pag > 1)
           --pag;
          linup = (pag * 52) - 51;
          s_pl(linup,linup+21,1);
          x = INICOL;
          y = INIREN;
          liny = 1;
          s_colinp(y,x,liny,pag);
        }
        k=1;
        }
        break;
case 0x51 : if (*ascii != 0) /* PG DN */
        k = 0;
        else
        { if (bmax > 1)
        { s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL+1);
          pmax = bmax/LIMPAG;
          if ((bmax%LIMPAG) > 0)
           ++pmax;
          if (pag < pmax)
           ++pag;
          linup = (pag * 52) - 51;
          s_pl(linup,linup+21,1);
          x = INICOL;
          y = INIREN;
          liny = 1;
          s_colinp(y,x,liny,pag);
        }
        k=1;
        }
        break;
case 0x76 : if (bmax > 1)  /* Ctrl-PG DN (Final de Texto) */
        { s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL+1);
        pmax = bmax/LIMPAG;
        if ((bmax%LIMPAG) > 0)
          ++pmax;
        pag = pmax;
        linup = (pag * 52) - 51;    /* Indice a 1ra línea de página-max */
        liny = 1;
        while((bmax-linup) > 21)    /* Avanza indice, dada la cantidad */
        { linup+=21;                /* de líneas de la página          */
          liny+=21;
        }
        s_pl(linup,linup+21,1);
        y = bmax-linup+INIREN;
        liny = bmax-linup+liny;
        linup = bmax-linup+linup;   /* Indice a última línea */
```

```
        lon = loncar(tex[linup]);
        x = lon+2;
        s_colinp(y,x,liny,pag);
        }
            k=1;
        break;
case 0x3b : ay();  /* AYUDA -F1- */
        s_posyx(y,x);
        k=1;
        break;
case 0x3c : { if (bmax > 1)  /* DOC. NUEVO -F2- */
        while (bmax > 1)       /* Libera lineas de buffer */
        free(tex[bmax--]);     /* si existen              */
       tex[bmax][0] = '\0';
       s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL+1);
       bmax=1;bliny=1;
       colx=1;liny=1;pag=1;
       pmax=1;bcolx=0;
       x=INICOL;y=INIREN;
       s_colinp(y,x,liny,pag);
       k=1;
       break;
       }
case 0x3d : { if (bmax == 1 && tex[1][0] == '\0');  /* SALVAR -F3- */
       else
       { nombre=(char *) newlin();
        s_ps(23,1,"Esc ",0x74);
        s_ps(23,5,"- Cancelar    ",0x70);
        guarda = (char *) s_safe(20,INICOL,LIMREN,LIMCOL);
        do /* Repite en error */
         { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
           s_box(20,INICOL,LIMREN,LIMCOL,0,7);
           s_ps(20,16," Salvar ",0x7f);
         } while(tecla2(nombre,21,INICOL+1,1));   /* Control: Tecleo de nombre */
        f1esc(ins);
        if (nombre[0] != 0x1b)
         if ( (archivo=fopen(nombre,"w+b")) == NULL)
          { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
            s_box(20,INICOL,LIMREN,LIMCOL,0,atri);
            s_offcur();
            s_ps(21,3,"Drive o Nombre de Archivo erroneo ",0x4F);
            delay(4000);
            s_oncur();
          }
          else
          { for (cont=0; cont<=bmax; cont++)
             { fputs(tex[cont],archivo);
            fputc(0x0A,archivo);
             }
            fclose(archivo);
          }
        s_restore(guarda);
        s_colinp(y,x,liny,pag);
        free(nombre);
       }
       k=1;
       break;
       }
case 0x3E : { FILE *archivo;     /* RECUPERA -F4- */
       nombre=(char *) newlin();
       s_ps(23,1,"Esc ",0x74);
       s_ps(23,5,"- Cancelar    ",0x70);
```

```
guarda = (char *) s_safe(20,INICOL,LIMREN,LIMCOL);
do /* Repite en error */
 { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
  s_box(20,INICOL,LIMREN,LIMCOL,0,7);
  s_ps(20,15," Recupera ",0x7f);
 } while(tecla2(nombre,21,INICOL + 1,0)); /* Control: Tecleo de nombre y errores de disco */
f1esc(ins);
if (nombre[0] != 0x1b)
 if ( (archivo=fopen(nombre,"r+b")) == NULL)
  { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
    s_box(20,INICOL,LIMREN,LIMCOL,0,atri);
    s_offcur();
    s_ps(21,3," Archivo no existente o protegido ",0x4F);
          delay(4000);
    s_restore(guarda);
    s_oncur();
        }
       else
  { if (bmax > 1)
     while (bmax)       /* Libera lineas de buffer */
      free(tex[bmax--]);     /* si existen              */
    linea = (char *) newlin();
    cont = 1;
    s_scroll(1,0,INIREN,INICOL,LIMREN,LIMCOL + 1);
    fgets(linea,80,archivo);
    linea[strlen(linea)-1] = '\0'; /* Chequeo de archivo, para determinar */
    if !strcmp(tex[0],linea) == 0 )       /* si es del procesador de texto.      */
     while (fgets(linea,80,archivo))
      { l = strlen(linea);
     linea[l-1] = '\0';
     if (cont > = mmax)
      tex = (char **) s_initex();  /* Asignación dinámica de memoria */
     tex[cont] = linea;
     if (cont < = LIMREN)
      s_ps(cont,INICOL,tex[cont],0x17);
     linea = (char *) newlin(linea);
     s_posyx(23,1);
      ++cont;
      }
    else
     { s_ps(21,1,"Archivo no perteneciente al procesador",0x4f);
       delay(3000);
       s_ps(21,1,"                                    ",atri);
     }
    bmax = cont - 1;       /* Inicializa Variables */
    pmax = cont/LIMPAG;
    if (!cont%LIMPAG);
     ++pmax;
    bcolx = 0;
    colx = 1;
    liny = pag = 1;
    x = INICOL;
    y = INIREN;
    fclose(archivo);
    free((char *)guarda);
        }
else
 s_restore(guarda);
s_colinp(y,x,liny,pag);
k=1;
break;
 }
```

```
case 0x3f : { if (bmax==1 && tex[1][0]=='\0')  /* IMPRIME -F5- */
      impri(1);     /* Archivo */
     else
      impri(0);     /* Screen */
           s_colinp(y,x,liny,pag);
     k = 1;
     break;
      }
case 0x40 : { linea = newlin();  /* DICCIONARIO -F6- */
     if (l != 6)
      { cont = 0;
        l  = bcolx;      /* columna */
       if (tex[bliny][l] != 0x20 && tex[bliny][l] != '\0')
        { while(valida(tex[bliny][l]) && l > 0 && tex[bliny][l] != 0x20)
          --l;  /* Busca inicio de palabra */
         if (!valida(tex[bliny][l]))
          ++l;
         lon = l;
         while(valida(tex[bliny][l]) && tex[bliny][l] != '\r' && tex[bliny][l] != '\0' && tex[bliny][l] != 0x20)
          linea[cont++] = tex[bliny][l++]; /* Guarda palabra */
         if (linea[0] != '\0')
          { s_ps(y,lon+INICOL,linea,0x7f);
            if (y>13)
            s_scroll(0,9,ININEN,INICOL,LIMREN,LIMCOL); /* Ultimas 13 lneas */
            numele = diccort('2',linea,numele); /* Dicc. Texto */
            if (y>13)
             s_pl(bliny-y+1,bliny-y+22,1); /* Todo el screen */
            else
             s_pl(bliny-y+14,bliny-y+22,14); /* De diccionario */
            s_ps(y,lon+INICOL,linea,atri);
           }
        }
      }
     else
      { numele = diccort('1',linea,numele); /* Dicc Teclado */
        s_pl(bliny-y+14,bliny-y+22,14); /* De diccionario */
      }
     f1esc(1);
     s_ps(23,19,"Lin   Col   Pag   ",atri);
     s_colinp(y,x,liny,pag);
     free(linea);
     k=1;
     break;
      }
case 0x41 : nombre=(char *) newlin(); /* BUSQUEDA -F7- */
      linea = newlin();
      s_ps ( 23,2,vacia,atri );
      s_ps ( 23,1,"Se buscará:",0x70 );
      tecla2 ( linea, 23, 12, 2);
      if (linea [0] != 0x1b )
      { s_ps ( 23,1,linvac,atri );
        s_ps ( 23,1,"1 hacia adelante, 2 hacia atras.",0x70 );
        s_ps ( 23,1,"1",0x74 );  s_ps ( 23,10,"d",0x74 );
        s_ps ( 23,19,"2",0x74 );  s_ps ( 23,28,"t",0x74 );
        s_ps ( 23,34,"1",0x7f );  s_posyx ( 23, 34 );
        int86 ( 0x16, &in, &out );
        switch (out.h.al)
         { case '1': case 'D': case 'd':
           case '2': case 'T': case 't':
           case 0x0d : lon = '+';
               nombre [0] = out.h.al; nombre [1] = '\0';
               switch (out.h.al)
```

```
            { case '1' : case 'D' :
              case 'd' : s_ps ( 23,34,nombre,0x7f );
                         break;
              case '2' : case 'T' :
              case 't' : s_ps ( 23,34,nombre,0x7f );
                         lon = '-';
            }
            delay ( 50 );
            do
             { l = buscapal ( &lon, &bliny, &bcolx, linea, &y, &x );
               if (l == 3) out.h.al = 78;   /** No **/
             } /** fin do_while(S/s) **/
            while ((out.h.al == 83) || (out.h.al == 115)); /** S/s **/
            pag = bliny/LIMPAG;
            liny = bliny%LIMPAG;
            if (liny > 0)
              ++pag;
             else if (liny == 0)
                liny = 52;
            break;
          default : break;
        } /** fin switch(out.h.al) **/
      } /** fin if(lineal=ESC) **/
      k=1;
      f1esc();
      s_ps(23,19,"Lin   Col   Pag   ",atri);
      s_colinp(y,x,liny,pag); s_oncur ();
      break;
case 0x42 : if (*ascii != 0)    /* SALIR -F8-   */
       k = 0;
      else
       k=0;
       break;
case 0x01 : { if (*ascii == 0x1B)        /* ESC : Fin de edición */
      { guarda = (char *)s_safe(INIREN,INICOL,LIMREN,LIMCOL+1);
        if (ll=menu()) < 7)
         { closegraph();
           textmode(1);
           s_scroll(1,0,0,0,24,79);
           s_box(0,0,24,39,0,0x17);
           f1esc(ins);
           s_restore(guarda);
           s_ps(23,19,"Lin   Col   Pag   ",atri);
           s_colinp(y,x,liny,pag);
         }
         else
          { free((char *)guarda);
            restorecrtmode();
          }
         dato = out.h.ah;
         *ascii = out.h.al;
         k = 2;
      }
      }
      break;
case 0xff : s_colinp(y,x,liny,pag);  /* Cuando sale de Menu con ESC */
      k = 1;
      break;
case 0x4f : if (*ascii != 0) /* FIN */
       k = 0;
      else
      { x = loncar(tex[bliny]) + 2;
```

```
        s_colinp(y,x,liny,pag);
        k = 1;
        }
        break;
   case 0x47 : if (*ascii != 0) /* INICIO */
        k = 0;
        else
        { x = INICOL;
        s_colinp(y,x,liny,pag);
        k = 1;
        }
        break;
   default   : { tespec(dato,ascii); /* Checa definición de CARACTERES ESPECIALES DEL ESPAÑOL*/
        if (*ascii < 0 || *ascii > 31)
        k = 0;
        else
        k = 1;
        }
  }

  *xx=x;
  *yy=y;
  return(k);
 }
```

V.2.3.2 Subrutinas **s_initex(), *newlin()

Aunque estas rutinas son muy pequeñas no dejan de ser valiosas. Proporcionan la memoria RAM suficiente para las líneas de texto y el apuntador general de las mismas, bajo una asignación dinámica de memoria.

```
/*************************************************************************/
/*      Realiza la Asignación Dinámica de Memoria de texto               */
/*      aumentando la memoria asignada en cada llamada.                  */
/*************************************************************************/
 char **s_initex()
 { char **tex2;
  int static n=1;
  ++n;
  mmax = n * MAXLIN;   /* # de líneas asignadas a buffer */
  if (n == 2)
   tex2 = (char **) malloc (mmax * sizeof(char *));
  else
   tex2 = (char **) realloc(tex,mmax * sizeof(char *));
  if (tex2 == (char **) NULL )
   { s_ps(23,1,"Memoria Insuficiente",0x4f);delay(2000);
     s_ps(23,1,"                    ",0x4f);
     return(NULL);
   }
  else
   return (tex2);
 }
```

```
/*************************************************************************/
/*        Asignación dinámica de memoria para lineas, dejando        */
/*        su contenido con caracteres de fin de cadena.              */
/*************************************************************************/
char *newlin()
 { char *linea;
   char i;
   if ( (linea = (char *)malloc(LIMCOL-INICOL+3) ) == NULL)
    { s_ps(23,1,"Memoria insuficiente",0x4f);delay(2000);
      s_ps(23,1,"                    ",0x4f);
      return(NULL);
    }
   else
    { for (i=0;i < 38;i++)
      linea[i] = '\0';
      return((char *)linea);
    }
 }
```

V.2.3.3 Subrutinas *s_safe(), *s_restore()

Salvan momentáneamente una porción de pantalla en memoria RAM con la intención de realizar otra función sobre el área y después restablecerla rápidamente.

```
/*************************************************************************/
/*              Salva el contenido de una ventana                */
/*************************************************************************/
char *s_safe(y1,x1,y2,x2)
char y1,x1,y2,x2;
{
char far *p;
char  *salva;
char  *salva1;
int cuantos,rows,columns,c,n;
  cuantos = 4 + (y2-y1+1) * (x2-x1+1)*2;
  if ((salva1=salva = malloc(cuantos)) == NULL)
   { s_ps(23,1,"Memoria insuficiente",0x4E);delay(3000);
     s_ps(23,1,"                    ",0x4E);
   }
  else
   { *salva1++ = y1;
     *salva1++ = x1;
     *salva1++ = y2;
     *salva1++ = x2;
     rows = y2-y1 + 1;
     columns = (x2-x1 + 1) * 2;
     for (c = 0 ; c < rows ; c++)
     {
     p = (char far *)(((long)(seg) + ((long)((y1+c) * 80l) + (x1*2l))));
       for(n = 0 ; n < columns ; n++)
      *salva1++ = *p++;
     }
   }
  return (char *)salva;
}
```

```
/*****************************************************************/
/*          Recupera el contenido de una ventana                 */
/*****************************************************************/
s_restore(salva)
char  *salva;
{
char y1,x1,y2,x2;
char far *p;
char *salva1 =salva;
int rows,columns,c,n;
  y1 =(*salva++);
  x1 =(*salva++);
  y2 =(*salva++);
  x2 =(*salva++);
  rows =y2-y1 + 1;
  columns =(x2-x1 + 1)*2;
  for (c =0;c <rows;c++)
  {
    p =(char far *)(((long)seg) + ((y1 +c)*80l) + (x1*2));
    for (n =0;n <columns;n++)
    {
    *p++ =*salva++;
    }
  }
  free((char *)salva1);
  return;
}
```

Nota: Esta subrutina necesita conocer el tipo de monitor que tiene el equipo, ello le permitirá manejar la dirección de video correcta, B800:0000 o B000:0000.

V.2.3.4 Subrutina inscar()

Es la subrutina que se ejecutará por omisión cuando la rutina control() no la relacione con alguna función del procesador y el programa principal la llame.

Como parte de la edición, inscar() es el estado que más frecuencia y variedad tiene, por ende uno de los más laboriosos. En su elaboración fue necesario tener especial cuidado con las líneas grandes (que no tienen espacios entre caracteres), la inserción normal y la sobre-escritura, los límites en longitud de línea, la inserción de espacios en blanco que separan palabras y saturan su longitud, el tratamiento de caracteres de control (\0, \r, 0x10) que requieren distintos manejos, entre otros.

El caso se complica al resolver cada una de las combinaciones que se derivan del punto anterior. Las soluciones emprendidas que se distinguen al respecto son : la subrutina sube_lg() para líneas grandes y la subrutina baja_p() para líneas normales (con al menos un espacio en blanco sobre la línea); por implantarse en ambas un proceso recursivo.

```
/*************************************************************************/
/*                  Inserta caracteres en la linea                       */
/*************************************************************************/
 inscar(col,y,x,ch)
 char col;   /* # de columna */
 char y,*x;  /* Col en Monitor */
 char ch;    /* caracter a insertar */
 {char lon,lon1,lonc,n=0,cont=0,*tmp;
  char lg=0;
  if (lins && ch != ' ' && tex[bliny][col] == 0x10) /* Tyovr al final de la linea, se    */
   tex[bliny][col] = '\0';                          /* coloca '\0' para tomar otra opción */
  lonc = loncar(tex[bliny]);
  if (lins == 0 && ch != ' ' && tex[bliny][col] != '\0' && tex[bliny][col] != '\r') ||
     (ins == 0 && ch == ' ' && col == 0 ))
   { if (ch == ' ' && tex[bliny][lonc] == 0x10) /* Tyovr, blanco al inicio de lg */
     { movmem(&tex[bliny][1],tex[bliny],lonc+1);
     sube_lg(1,0,y);          /* Baja linea grande */
     }
     else
     { tex[bliny][bcolx] = ch;
     s_ps(y,INICOL,tex[bliny],0x17);   /* Reestablece linea en Display */
     ++(*x);
     }
     if (col == 0 && ch == ' ')
     { tex[bliny-1][36] = '\0'; /* Elimina posible 0x10 de linea anterior */
     if (y > INIREN)
     s_pchar(y-1,38,0x20,atri);  /* Quita en screen 0x10 */
     }
   }
   else
   { tmp = (char *) newlin ();
     col = bcolx;
     if (lins && tex[bliny][lonc] == '\r')
      tex[bliny][lonc] = '\0';
     lon1 = lon = strlen(tex[bliny]);
     if (ins || tex[bliny][lonc] == 0x10) /* Checa inserción */
      movmem(&tex[bliny][col],&tex[bliny][col+1],lon-col+1); /* Mueve texto en Linea-Buffer */
     else
      tex[bliny][lon+1] = '\0'; /* Asegura EOL cuando se inserta al final en INS = 0 */
     tex[bliny][col] = ch;  /* Inserta nuevo caracter */
     if (lonc == (LIMCOL-1))
      { while ((tex[bliny][lon1] != ' ') && (lon1 >= 0))
      { ++cont;
        --lon1;
      }
     if (lon1 == lon)       /* Caso en el que se baja un espacio */
      { --lon1; ++cont; }
     if (cont >= LIMCOL && tex[bliny][0] != 32)      /* Control: Longitud maxima de palabra */
      { --cont;
        if (tex[bliny][cont] == '\r')
         { movmem(&tex[bliny][--cont],tmp,2);
           lonc = 2;
         }
        else
         { if (tex[bliny][cont] == 0x10)
         { --cont;
           lg = 1;
         }
           movmem(&tex[bliny][cont],tmp,1);
           lonc = 1;
         }
        tex[bliny][cont] = 0x10; /* Caracter de control de palabra grande */
```

```
  tex[bliny][cont+1] = '\0';
  cont = lonc;
 }
else
 { movmem(&tex[bliny][lon1+1],tmp,cont+1);  /* Copia palabra con EOL o RTN */
  if (lon1 < 0 )
   lon1 = 0;
  if ((tex[bliny][lon1] == 0x20 && lon1 > 0) && *tmp != 0x20)
   tex[bliny][lon1] = '\0';
  else
   tex[bliny][lon1+1] = '\0';    /* Cuando se baja espacio, no se borra espacio de la linea */
 }                 /* o se inserta espacio al inicio de linea grande .        */
if (*tmp != 0x10)
 n = baja_p(tmp,cont,0,lg);
else
 tex[bliny][36] = '\0';
if (n == LIMREN && y > INIREN)
 s_pl(bliny-1,bliny+n,y-1);  /* Rectifica algún 0x01 borrado */
else
 s_pl(bliny,bliny+n,y); /* Actualiza lineas en display */
lonc = loncar(tmp);
if ( *x >= (38-lonc) && (tmp[0] != ' ' || ch == ' '))
 { *x = *x + lonc - 36;
   ++y; ++liny;
   if (liny == (LIMPAG+1))
    { liny = 1;
      ++pag;
    }
   if (y > LIMREN)
    { --y;
      pantup(y,*x);
    }
 }
else
  ++(*x);
}
  else
{
s_ps(y,INICOL,tex[bliny],0x17);   /* Reestablece linea en Display */
if (tex[bliny][0] == 32 && bliny>0) /* Asegura eliminar 0x10 de bliny-1 al */
 tex[bliny-1][36] = '\0';          /* insertar (o existir) ' ' al inicio */
 ++(*x);
lon1 = lonc = loncar(tex[bliny]);
if ((*x) > (lonc+1))
  *x = lonc + 2;
}
  free(tmp);
  if (bliny > 0 && ch == 32 && bcolx<36)  /* Caso en el que cabe información en linea anterior */
{ lonc = loncar(tex[bliny-1]); /* Linea anterior a inserción de 32 */
  if (y > INIREN && col == 0)
   s_pchar(y-1,38,0x20,atri);  /* Quita en screen 0x10 */
  lon=0;
  while (tex[bliny][lon] != 32 && tex[bliny][lon] != '\0' && lon < 37)
    ++lon; /* Long. palabra */
  if (tex[bliny-1][lonc] != '\r' && lon < LIMCOL-lonc-1 && lon > 0)/* Si lin.ant. esta sin \r y 0<lon< =espacio,sube pal.*/
   { if (tex[bliny][lon] == '\0' )
      { sube_p(LIMCOL-lonc-1,0,y-2); /* Sube toda la linea en bliny-1 */
     --y; --liny;
     if (liny == 0) {liny = 52; --pag;}
      }
     else
      { if ((LIMCOL-lonc-1) == lon1)
```

```
      ++lonc; /* Si linea completa "casi cabe" en la anterior */
     sube_p(LIMCOL-lonc-1,0,y-1);
     (*x)-= ++lon;
      }
    }
    else
     if (n > 0 && lon > 0)/* Checa si se bajaron palabras, para subir sobre linea actual */
      { lon1 = loncar(tex[bliny+1]); /* Linea donde se bajo inf. */
       if (tex[bliny+1][lon1] == 0x10)
        { if (!ins)
        { movmem(&tex[bliny+1][1],tex[bliny+1],lon1+1);
          ++lon;
        }
        sube_lg(lon,1,y);
        }
      }
   }
  } /* else INS = "1" */
  s_colinp(y,*x,liny,pag);
  return(y);
}
```

Nota: Para el movimiento de posiciones sobre la línea y entre líneas cuando se ejecuta la inserción, se toma a la librería movmem() que permite cambiar la dirección de los apuntadores de memoria.

V.2.3.5 Subrutina borcar()

La subrutina borcar() envuelve los mismos problemas que inscar() aunque su propósito sea inverso; mientras borcar() borra caracteres, inscar() los adiciona.

En condiciones análogas, borcar() utiliza el proceso recursivo para solucionar en gran medida la función de borrado. Implanta su proceso en las subrutinas sube_p() y sube_lg(). El gran detalle en la rutina borcar() es la llamada indirecta a sube_lg(), al invocar a sube_p() y este a sube_lg(), estableciendo una "doble" recursión.

El módulo borcar() se activa desde control() con las teclas Backspace, Delete o Suprime.

```
/*************************************************************************/
/*                  Borra caracteres en la linea                         */
/*************************************************************************/
 borcar(col,y,x,z)
  char col;   /* # de columna-texto */
  char *y,*x;  /* Posiciones en Display */
  char z; /* 0:Backspace, 1:Delete */
  { char lon,lonc;
    lon = strlen(tex[bliny]);
    lonc = loncar(tex[bliny]);
    if (z)    /* Cierto:Delete */
     bcolx = col;
    if (!(*x) > INICOL)
     { if (tex[bliny][col-1] == '\r') ++(*x);    /* Para no mover cursor si existe RTN */
       movmem(&tex[bliny][col],&tex[bliny][col-1],lon-col+1);    /* Corrimiento a la izquierda de linea-buffer */
       if ((tex[bliny][0] == ' ' || tex[bliny][0] == '\0' || tex[bliny][0] == '\r') && tex[bliny-1][36] == 0x10)
```

```
{ tex[bliny-1][36] = '\0'; /* Elimina 0x10 al borrar el resto de lg */
  if (!*y) > INIREN)
    s_pchar(*y)-1,38,0x20,atri);  /* Elimina 0x10 de pantalla */
}
  --(*x);
  if (bliny < bmax) && (bmax > 1) && (tex[bliny][lon-2] != '\r'))   /* Checa si entra a sube_p */
{ if (tex[bliny][0] == '\0') /* Linea vacia */
    borln(bliny,(*y));  /* Dato de espacio libre en la linea */
   else
    { lonc = loncar(tex[bliny + 1]);
      lon = LIMCOL-lon;
      if (lon == lonc)
       --lon;  /* Linea completa con '\r' y/o '\0'*/
      sube_p(lon,1,(*y));    /* Dato de espacio libre en la linea */
    }
}
  else
{ s_ps((*y),INICOL,vacia,0x17);   /* Actualización cuando se borra en la linea máxima */
  s_ps((*y),INICOL,tex[bliny],0x17);
}
}
else /* x = INICOL */
 if (bliny > 1)    /* Cierto: checa corrimiento de palabra(s) hacia arriba */
  { if (*y > INIREN )
  --(*y);
 else
   s_scroll(1,1,INIREN,INICOL,LIMREN,LIMCOL);
 lon = strlen(tex[bliny-1]);
   lonc = loncar(tex[bliny-1]);
 if (tex[bliny][0] == '\r' && z )
  { tex[bliny-1][lon-1] = '\r';
    borln(bliny,(*y) + 1);
    (*x) = lonc + 2;
  }
 else
  {
    if (tex[bliny-1][lon-1] == '\r')        /* Borra RTN */
     { tex[bliny-1][lon-1] = '\0';
       sube_p(LIMCOL-lonc-1,0,*y); /* Ojo -1 */
       movmem(&tex[bliny-1][lon],&tex[bliny-1][lon-1],35);  /* Suprime blanco de RTN */
       s_pchar(*y,39,0xba,atri);
       (*x) = lonc+2;
     }
    else
     { if (tex[bliny-1][lon-1] != 0x10) /* Checa si es palabra grande */
     { if ((lon-1) == 0 && !z)
        borln(bliny-1,*y);    /* Elimina linea de un caracter (BS) */
       else
        { tex[bliny-1][lon-1] = '\0';
          sube_p(LIMCOL-lonc-1,0,*y);
          --lonc;
        }
     }
       (*x) = lonc+2;
       if (*x == 1) ++(*x); /* Posible corrección de x (si lonc = 0)*/
     }
 }
 s_pl(bliny-1,bliny-1,*y);
 --liny;
 if ((pag >= 1) && (liny == 0))
  { liny = 52; --pag; }
  }
```

```
 s_colinp((*y),(*x),liny,pag);
}
```

Nota: Para el movimiento de posiciones sobre la línea y entre líneas al instante de borrar, se usa la librería movmem() que permite modificar la dirección de apuntadores de texto en memoria.

V.2.3.6 Subrutina impri()

La subrutina realiza la impresión de algún documento que se encuentra en disco o memoria principal, a través del puerto paralelo de la computadora con una letra o font de 8 cpp. Comprende en conjunto, el manejo de la comunicación y el establecimiento del formato de hoja para impresión.

En la comunicación se elabora un emulador estándar a partir de secuencias de escape y códigos de control de impresoras Epson e IBM. Y detalla en la subrutina errimp() los errores de enlace.

En el formato de hoja se crea a la subrutina pagina() para colocar el número de página, a espaciar() para el espaciado de líneas en forma horizontal y a sprint() para imprimir líneas.

```
/*****************************************************************/
/*                  Realiza la impresión                         */
/*****************************************************************/
 impri(op)
 char op;    /* 1:Archivo, 0:Screen */
{ int cont=0,pag=1,nl = 1;
  char *copia,*nombre,lon,s=1;
  char *espaciar();
  guarda = (char *) s_safe(20,1,LIMREN,38);
  if (errimp())
   { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
     s_box(20,1,LIMREN,38,0,atri);
     copia = newlin();
     nombre = newlin();
     if (op)
      { s_ps(23,1,"Esc ",0x74);
        s_ps(23,5,"- Cancelar  ",0x70);
        do /* Repite en error-disco */
         { s_scroll(1,0,20,INICOL,LIMREN,LIMCOL);
           s_box(20,1,LIMREN,38,0,atri);
           s_ps(20,14," Imprimir ",0x7f);
         } while(tecla2(nombre,21,INICOL,0));      /* Op. Archivo */
        s_ps(23,1,"F1      Esc",0x74);
        s_ps(23,3,"-Ayuda",0x70);
        s_ps(23,13,"-Menu",0x70);
        if (nombre[0] != 0x1b)
        { if ((archivo=fopen(nombre,"r+b")) == NULL)
            { s_ps(21,3,"Archivo no existente o protegido",0x4f);delay(3000);
              s_ps(21,3,vac2,atri);
              cimp = -1;
            }
        }
        else
         cimp = -1;
```

```
 }
if (cimp == 1)
 { s_ps(21,4,"¿Deseas Letra de Calidad (S/N)?  ",0x7f);
   s_posyx(21,36);
   if (sn(21,36))
    { cprint(27);cprint(71);} /* Doble golpe */
   s_ps(21,4,"Deseas Número de Página (S/N)?  ",0x7f);
   s_posyx(21,36);
   s = sn(21,36);
   s_ps(21,4,vac2,atri);
   s_ps(21,5,"*Imprimiendo*",0x97);
   s_ps(21,21,"ESC : Cancelar",atri);
   cprint(27); cprint(87); cprint(1);  /* Activa Doble Ancho */
   pagina(pag,s);      /* Número de página */
 }
while (cimp == 1)
 { if (kbhit())  /* Checa presión de teclas */
 { in.h.ah = 0;
   int86(0x16,&in,&out);
   if (out.h.al == 0x1b)
    { nl = bmax;
      cimp = 0;
    }
 }
if (op)  /* Op. Archivo */
 { if (fgets(copia,80,archivo))
    { lon = strlen(copia);
      copia[--lon] = '\0';
    }
   else
    cimp = 0;
 }
else
 { lon = strlen(tex[nl]);
   movmem(tex[nl],copia,lon + 1);
 }
if (cimp == 1)
 { ++cont;
   if (cont > 52)
    { cprint(12);  /* Salto a la siguiente línea */
      cprint(27); cprint(67); cprint(67);  /* Longitud de página */
      cprint(27); cprint(87); cprint(1);  /* Activa Doble Ancho */
      cont = 1;
      ++pag;
      pagina(pag,s);
    }
   espacier(copia,lon);
   cprint(32);cprint(32);cprint(32);  /* Margen izquierdo */
   sprint(copia);
   if (lop)
    { ++nl;  /* Número de línea en screen */
      if (nl > bmax)
     cimp = 0;
    }
 }
 }
if (cimp == 0)
 biosprint(0,0x0c,0);
free(copia);
free(nombre);
cimp = 1;
if (op)
```

```
    fclose(archivo);
    cprint(27);cprint(72); /* Cancela posible letra de calidad */
   }
  s_restore(guarda);
}
```

Nota: Una de las rutinas más problemáticas fue errimp() para errores de comunicación y espaciar() para justificación a la derecha. La primera en el uso de interrupciones y la segunda en el cálculo de la posición, cantidad y lugar de inserción de espacios.

V.2.3.7 tespec()

Esta subrutina da la alternativa para colocar los caracteres especiales del español cuando no se encuentran sobre teclado. Cambia la combinación asignada de teclas por el carácter del español.

```
/*************************************************************************/
/* Definición de caracteres especiales: Dado el código scan y caracter   */
/* ASCII de teclado, se obtiene el caracter especial del español :       */
/* á   é   í   ó   ú   ü   Ü   ñ   Ñ   ¡   ¿ */
/*************************************************************************/
tespec (scan,ascii)
 char scan;   /* Scan Code  */
 char *ascii;  /* Caracter ASCII */
 { char entrada;
  entrada = *ascii;
  switch (scan) {

   case 0x1e : if (*ascii == 0x1)    /* Ctrl A */
          *ascii = 0xa0;     /* á */
         else
          if (*ascii == 0x0) /* Alt A */
           *ascii = 0xad;    /* ¡ */
         break;
   case 0x12 : if (*ascii == 0x5)    /* Ctrl E */
          *ascii = 0x82;     /* é */
         break;
   case 0x17 : if (*ascii == 0x9)    /* Ctrl I */
          *ascii = 0xa1;     /* í */
         else
          if (*ascii == 0x0) /* Alt I */
           *ascii = 0xa8;    /* ¿ */
         break;
   case 0x18 : if (*ascii == 0x0f)   /* Ctrl O */
          *ascii = 0xa2;     /* ó */
         break;
   case 0x16 : if (*ascii == 0x15)   /* Ctrl U */
          *ascii = 0xa3; /* ú */
         else
             if (*ascii == 0x0) /* Alt U */
           if (bioskey(2) & 0x0040)
            *ascii = 0x9a;       /* Ü (mayúscula) */
           else
            *ascii = 0x81;       /* ü (minúscula) */
         break;
   case 0x31 : if (*ascii == 0x0e)   /* Ctrl N */
          if (bioskey(2) & 0x0040) /* Checa si CAPS on */
           *ascii = 0xa5;       /* Ñ (mayúscula) */
```

```
        else
          *ascii = 0xa4;        /* ñ (minúscula) */
        break;

    }
  if (*ascii != entrada)
   return(0); /* Selección de alguna definición */
  return(1);  /* Definición no encontrada */
}
```

Nota: Checa los dos códigos que recibe de teclado: código scan y código Ascii.

V.2.3.8 menu()

El fundamento central de menu() es la colocación del Menú Principal y el chequeo de la función elegida.

Se auxilia de la rutina mgraf() que detecta y establece el modo gráfico adecuado para el equipo. Junto a venmen() que traza y dibuja el Menú Principal.

```
/*****************************************************************************/
/*        Coloca menu principal en pantalla ( Modo Gráfico ) y          */
/*        determina que función es requerida por el usuario .           */
/*****************************************************************************/
menu()
 { char f=2, nf=1;
   char cfs,cfns;
   char fx;

   mgraf(&fx);
   if (fx == 2)
    { cfs = 3; cfns = 5; gris = 15;}
   else
    { cfs = 1; cfns = 2; gris = 3;}
   venmen(fx,f,cfs,cfns);
   while (nf)
    { in.h.ah = 0;
      int86(0x16,&in,&out);
      if (out.h.al == 0)
       switch (out.h.ah) {
      case 75: if (f > 4) /* Izq */
            { efes(fx,cfns,f,0);
           if (f == 8) --f;
           f-=3;
             efes(fx,cfs,f,1);
            }
           break;
      case 77: if (f < 5) /* Der */
            { efes(fx,cfns,f,0);
             f+=3;
             efes(fx,cfs,f,1);
            }
           break;
      case 72: if (f > 2 )    /* Arr */
            { efes(fx,cfns,f,0);
             --f;
             efes(fx,cfs,f,1);
```

```
        }
       break;
    case 80: if (f < 8)    /* Aba */
         { efes(fx,cfns,f,0);
           ++f;
           efes(fx,cfs,f,1);
         }
       break;
    case 59: ayd(fx,f,cfs,cfns);  /* F1 */
       break;
    case 60: f = 2; nf = 0;break; /* F2 */
    case 61: f = 3; nf = 0;break; /* F3 */
    case 62: f = 4; nf = 0;break; /* F4 */
    case 63: f = 5; nf = 0;break; /* F5 */
    case 64: f = 6; nf = 0;break; /* F6 */
    case 65: f = 7; nf = 0;break; /* F7 */
    case 66: f = 8; nf = 0;break; /* F8 */
     }
     else
      if (out.h.al == 13 && out.h.ah == 28) /* RTN */
     { out.h.al = 0; out.h.ah = 58+f;
       nf = 0;
     }
       else if (out.h.al == 0x1b && out.h.ah == 1)    /* ESC - Regresa Texto */
     { out.h.ah = 0xff;
       nf = 0;
     }
    }
   return(f);
}
```

V.2.3.9 venmen()

Con venmen() se coloca el Menú Principal en pantalla Modo Gráfico.

```
/*****************************************************************************/
/*        Pinta el menú principal en pantalla (Modo Gráfico).        */
/*****************************************************************************/
 venmen(char fx,char f,char cfs,char cfns)
  { char fig[] =
   {0,0,0,42,43,0,2,41,44,0,2,40,45,0,2,39,44,0,38,41,42,43,0,2,37,40,42,
    0,36,39,40,41,0,35,40,0,34,39,0,33,35,36,38,0,32,34,36,37,0,31,33,35,36,
    0,30,32,34,35,0,29,31,33,34,0,28,31,32,33,0,27,32,0,26,31,0,25,30,0,24,29,
    0,23,28,0,22,27,0,21,26,0,20,25,0,19,24,0,18,23,0,17,22,0,16,21,0,15,20,
    0,14,19,0,13,18,0,12,17,0,11,16,0,10,15,0,9,14,0,8,9,13,0,2,8,12,0,7,10,11,
    0,7,9,10,0,6,8,0,5,6,7,0,5,6,1,
    0,49,2,5,29,0,5,6,7,2,28,32,0,5,2,7,11,32,33,0,5,2,11,34,0,5,13,32,34,
    0,5,13,32,34,49,0,3,8,5,13,32,34,0,5,13,2,16,34,0,5,13,16,18,19,21,23,25,27,29,2,31,37,
    0,5,13,16,2,19,22,24,26,28,30,32,34,2,36,39,0,5,13,16,2,22,39,0,3,5,5,13,16,25,39,
    0,49,5,13,16,25,2,31,36,39,0,5,13,16,25,2,31,36,39,0,5,13,16,17,25,39,
    0,5,13,18,19,25,39,0,5,13,20,21,25,39,0,5,13,2,16,25,39,0,5,13,16,2,25,39,
    0,5,13,16,32,34,0,5,13,16,32,34,0,5,13,16,32,34,0,5,13,16,2,22,26,32,34,
    0,5,13,16,2,22,26,32,34,0,5,13,16,2,22,26,32,34,0,5,13,16,32,34,0,5,13,16,32,34,
    0,5,6,13,16,32,34,0,7,8,13,2,16,32,34,0,9,10,11,13,34,0,2,12,34,49,1,
    0,3,4,5,13,32,34,0,5,13,2,27,43,0,5,13,26,2,29,43,0,5,13,25,28,32,43,
    0,5,13,24,27,31,42,0,5,13,2,16,23,26,30,41,0,5,13,16,18,19,25,29,40,
    0,5,13,16,2,19,22,24,28,39,0,5,13,16,2,22,39,0,3,5,5,13,16,25,39,0,1,
    0,0,0,2,29,47,0,29,30,31,2,44,47,0,28,30,44,46,0,28,29,30,2,43,46,
    0,27,29,43,45,0,27,28,29,2,42,45,0,26,28,42,44,0,26,27,28,2,41,44,
    0,25,27,29,31,33,35,37,39,41,43,0,25,26,27,2,40,43,0,24,26,40,42,
```

```
0,24,25,26,2,39,42,0,23,25,39,41,0,23,24,25,2,38,41,0,22,24,37,39,40,
0,2,21,24,26,28,30,32,34,2,36,39,0,2,11,20,22,36,2,38,46,0,10,20,21,22,2,35,38,45,46,
0,9,10,11,19,21,35,2,37,41,44,46,0,8,2,13,16,2,18,21,2,34,37,40,43,46,
0,7,8,9,12,15,16,18,20,34,36,39,41,46,0,6,11,2,14,35,38,2,41,46,0,5,6,7,10,37,2,40,47,
0,4,2,9,36,2,39,44,47,48,0,4,38,2,40,44,47,48,0,4,2,5,37,2,40,48,
0,4,5,15,37,2,41,47,0,4,5,7,10,13,15,18,20,22,23,25,27,28,29,31,32,34,35,37,44,
0,4,5,15,18,20,22,23,25,27,29,2,31,35,37,43,0,4,5,15,18,20,22,24,25,27,28,29,31,33,35,37,42,
0,4,5,7,10,13,15,18,19,20,22,24,25,27,29,31,35,37,41,0,4,5,15,37,40,
0,4,2,5,37,39,0,4,37,38,0,2,4,37,1,
0,2,18,23,2,28,33,0,16,17,23,24,26,27,33,34,35,0,2,6,15,25,2,36,45,0,6,25,45,
0,3,31,5,6,25,45,46,0,5,6,2,17,33,45,46,0,5,15,16,18,20,22,24,25,26,28,30,32,34,35,36,45,46,
0,2,5,15,17,19,21,24,25,26,29,31,33,35,2,37,46,
0,5,6,8,10,12,14,16,18,20,22,24,25,26,28,30,32,34,36,38,40,42,44,45,46,
0,5,7,9,11,13,15,2,17,33,35,37,39,41,43,45,46,0,2,5,16,2,23,27,2,34,46,
0,24,25,26,1,
0,0,2,21,29,0,2,18,21,2,29,32,0,2,16,19,2,31,34,0,15,16,17,33,34,35,
0,14,15,16,34,35,36,0,2,13,16,2,34,37,0,3,5,13,14,15,35,36,37,
0,2,13,16,2,34,37,0,14,15,16,34,35,36,0,15,16,17,33,34,35,0,2,16,19,2,31,34,
0,2,18,21,2,29,32,0,2,21,29,0,22,23,27,28,0,22,28,0,3,2,22,23,27,28,
0,2,21,24,2,26,29,0,21,24,25,26,29,0,3,11,21,29,0,21,24,25,26,29,
0,21,24,26,29,0,21,24,25,26,29,0,21,22,28,29,0,22,23,27,28,0,2,23,27,50);

int x,y,y2=0,i=0,i2,i3,j;
char fg=1,rpt=0;
bar (0,0,319*fx,199);
setcolor(0);
rectangle(1,1,318*fx,186);    /* Ventana de Monitor */
rectangle(105*fx,3,155*fx,53);     /* Ventana Fig.1 */
rectangle(105*fx,65,155*fx,115);   /* Ventana Fig.2 */
rectangle(105*fx,127,155*fx,177);  /* Ventana Fig.3 */
rectangle(265*fx,3,315*fx,53); /* Ventana Fig.4 */
rectangle(265*fx,65,315*fx,115);   /* Ventana Fig.5 */
rectangle(265*fx,127,315*fx,177);  /* Ventana Fig.6 */
rectangle(0,187,319*fx,199);  /* Recuadro, abajo */
outtextxy(30*fx,18,"ESCRIBIR");outtextxy(30*fx,28,"DOCUMENTO");
outtextxy(30*fx,80,"GUARDAR");outtextxy(30*fx,90,"DOCUMENTO");
outtextxy(30*fx,142,"RECUPERAR"); outtextxy(30*fx,152,"DOCUMENTO");
outtextxy(5*fx,177,"<Esc> - Texto");
outtextxy(190*fx,18,"IMPRIMIR");
if (fx == 2)
 outtextxy(190*fx,80,"DICCIONARIO");
else
 {outtextxy(190*fx,80,"DICCIONA-");outtextxy(190*fx,90,"RIO");}
outtextxy(190*fx,142,"BUSQUEDA");
outtextxy(190*fx,177,"SALIR");
setcolor(cfns);outtextxy(5,190,"F1");
setcolor(0);  outtextxy(30,190,"AYUDA");
x = 105*fx+1+(fx-1)*28;
y = 7;
y2 = 131-69;
while (figli) != 50)
 switch (figli)) {
  case 0: ++y;     /* Siguiente línea */
        ++i;break;
  case 1: switch (++fg) {
     case 2: x = 105*fx+1+(fx-1)*28;
                y = 69; break;
     case 3: x = 105*fx+1+(fx-1)*28;
          y = 137; break;
     case 4: x = 263*fx+1+(fx-1)*28;
                y = 7; break;
     case 5: x = 264*fx+1+(fx-1)*28;
```

```
                y = 69; break;
        case 6: x = 264*fx + 1 + (fx-1)*28;
                y = 131; break;
      }
       ++i;break;
    case 2: line(fig[++i]+x,y,fig[++i]+x,y);  /* Linea /2 ptos. */
          if (rpt)
           line(fig[i-1]+x,y+y2,fig[i]+x,y+y2);
          ++i;break;
    case 3: i2 = ++i;     /* Repite Linea */
        for (j=0;j<fig[i2];j++)
           { while (fig[++i] != 0)
             putpixel(fig[i]+x,y,0);
             ++y;
             i3 = i;    /* Ultima pos. de linea */
             i = i2;    /* Pos.inicial-1 de linea*/
           }
          i = ++i3;     /* Pos. inicial de siguiente linea */
        break;
    case 49: ++i;    /* Repetición de figura */
      rpt = (!rpt) ? 1 : 0;
      break;
    case 50: break;
    default: putpixel(fig[i]+x,y,0);
          if (rpt)
           putpixel(fig[i]+x,y+y2,0);
           ++i;break;
  } /*end switch */
  line(160*fx,1,getmaxx()/2+1,getmaxy()-14); /* Linea azul punteada */
  setfillstyle(1,cfs);
  bar(75,187,76,199); /* Linea abajo */
  efes(fx,cfns,2,0);efes(fx,cfns,3,0);efes(fx,cfns,4,0);
  efes(fx,cfns,5,0);efes(fx,cfns,6,0);efes(fx,cfns,7,0);efes(fx,cfns,8,0);
  efes(fx,cfs,f,1);
}
```

Nota: La controversia en el desarrollo de menu() empezó en la elección, distribución, trazo y diseño de figuras en la pantalla. El segundo paso, a la forma de guardar y recuperar figuras. En este caso se optó por un recuadro de 50x50 pixels y 6 figuras, en correspondencia a 15000 puntos o pixels por manejarse. La primera idea fue guardar cada punto en un byte, lo que comprendía cerca de 15 Kbytes (demasiado espacio), después guardar en forma contigua dentro de un byte a 8 puntos que harían decrecer el espacio a 2 Kbytes. Finalmente al analizarse detalladamente, se observó poca velocidad de despliegue. De ello se desprendio la idea de guardar solo la posición de los pixeles encendidos bajo un byte y establecer claves que indicaran el dibujo de líneas completas, repetición de líneas, repetición de secciones de figuras, entre otros. Con esta opción se redujo el software a desarrollar, se aumentó la velocidad y se disminuyó a 1 Kbyte el espacio para guardar las 6 figuras.

V.2.3.10 Subrutina Diccort()

Esta subrutina se encarga de buscar en una lista de términos, la palabra seleccionada. Si la encuentra, mostrará el significado de dicha palabra. De lo contrario, posicionará el cursor en

el rango de términos más cercano a la palabra buscada, con el objeto de que el usuario consulte el rango de términos más cercano al significado de lo que busca.

La información que recibe la subrutina diccort() al momento del llamado será una "**D**" para indicar un llamado desde el menú principal y una "**d**" para indicar un llamado desde el texto. En este último se proporciona también la palabra a consultar.

```
/*************************************************************************/
/****                  diccort ( )                                  ****/
/****  Subrutina que nos permite trabajar en la parte de diccionario  ****/
/****  o la parte de ortografía, segun se seleccione en la llamada.   ****/
/*************************************************************************/
int diccort ( selec, palab, nelem )
char   *palab;
char   selec;  /** Variable que permite seleccionar entre **/
       /** la parte de Diccionario u Ortografía.  **/
       /**   Donde :                              **/
       /**   1-D  -  Palabra tomada de  teclado   **/
       /**   3-O  -    "     "    "    "     "     **/
       /**   2-d  -    "     "    " memoria  "     **/
       /**   4-o  -    "     "    "    "     "     **/
 int          nelem;  /** Contendra el num. de palabras del dicc. leidas **/
{ int         lonpal, i, j, k;
 union REGS    ent, sal;
 struct local  palenc1;

 ent.h.ah = 0;
 s_offcur ();
 s_box ( 0,0,24,39,0,0x17 );  /** se tiene que quitar al acoplarse lod prog.**/
 s_box ( 14,0,24,39,0,0x71 );
 switch ( selec )
  { case '1':  case '2' :  s_ps ( 14,13,"Diccionario",0x71 );
                           break;
  } /** fin  switch(selec) **/
 for ( j = 15; j <= 23; j++ )
   s_ps( ),1,linvac,0x71 );
 if ( locpal [0]->arch [0] != '*' )
  nelem = llena ( );
 if ( nelem > 0 )
  { do
    { s_offcur ();   /* apaga cursor */
      for ( j = 15; j <= 23; j++ )
        s_ps( j,1,linvac,0x71 );
      switch ( selec )
       { case '1':  s_ps( 15,1,"Dame la palabra",0x71 );   /*  escribe string   */
                    s_ps( 15,16,":",0xF1 ); /*  escribe string   */
                    s_posyx ( 15, 17 );
                    s_oncur ();  /*  enciende cursor  */
                    tecla2 ( palab, 15, 17, 2 );
                    s_ps( 15,16,":",0x71 ); /*  escribe string   */
                    break;
         case '2':  s_ps( 15,1,"Palabra seleccionada",0x71 );   /*  escribe string   */
                    s_ps( 15,21,":",0x71 );
                    s_ps( 15,22,palab,0x17 );
                    break;
       } /** fin  switch(selec)  **/
      s_posyx ( 23, 38 );
      lonpal = 0; j = 1;
```

```
do
 { if ( lonpal <= 22 )
    switch ( palab [lonpal] )
     { case 'é': case 'è': case 'í': case 'ó': case 'ú':
       case 'ü': case 'ñ': case 'Ñ': case '-': case 'Ü': break;
       case '\0' :  case '\n' :  j = 0; break;
       default : if ((palab [lonpal] >= 'a') && (palab [lonpal] <= 'z'))
                    break;
                 else if ((palab [lonpal] >= 'A') && (palab [lonpal] <= 'Z'))
                      break;
                      else  { j = -1; lonpal++;
                              palab [0] = '\0';
                            }
     } /** fin switch **/
    else j = 0;
   lonpal++;
 } /** fin while (jl=0) **/
while ( j == 1 );
palab [lonpal - 1] = '\0';
if (( i = strlen ( palab )) != 0 )
  { j = busca ( palab, nelem, j );
    if ( j == 1 )
      { switch ( selec )
         { case '1' : case '2':  sal.h.al = 83; break;
         } /** fin  switch(selec) **/
        if (palenc.sign != 0 )
         { s_offcur ();
           lonpal = palenc.sign;
           i = k = posi - 2;
           do
            { j = compare ( palab, locpal [k++] -> palb, j );
              if ( j == 0 && palenc.sign == 0 )
                i = -1;
            } /** fin  while(i>0&&k<=posi+3)  **/
           while (( i > 0 ) && ( k <= posi + 3 ));
           k--;
           if ( i < 0 )
            { for ( i = 0; locpal[k]->palb [i] != '\0'; i++ )
                 palenc.palb [i]  = locpal[k]->palb [i];
              palenc.palb [i]  = '\0';
              for ( i = 0; i <= 1; i++ )
                palenc.arch [i] = locpal[k]->arch [i];
              for ( i = 0; i <= 3; i++ )
                palenc.pos [i] = locpal[k]->pos [i];
            } /**  fin if_then(i>0)  **/
           else
            { erroracen ()
              delay ( 2000 );
              for ( j = 20; j <= 22; j++ )
                s_ps ( j,1,linvac,0x71 );
              s_ps ( 17,17,palenc.palb,0x17 );
              delay ( 2000 );
              switch ( selec )
               { case '1' : case '2' :
                           sal.h.al = 83;
                           s_oncur ();
                           break;
                 case '3' : case '4' :
                           s_ps ( 23,1,linvac,0x71 );
                           s_ps ( 23,7,"Quieres su significado (s/n) : ?",0x07 );
                           s_posyx ( 23,38 );
                           s_oncur ();
```

```
                int86 ( 0x16, &ent, &sal );
                break;
            } /** fin switch(selec) **/
          } /** fin if_else(i>0) **/
        } /** fin if(palenc.sig!=0) **/
      if (( sal.h.al == 83 ) || ( sal.h.al == 115 )) /** S/s **/
        { s_offcur ( );  /* apaga cursor     */
          s_ps ( 23,1,linvac,0x71 );
          s_ps ( 23,33,"Espera",0x97 );
          obten ( );
          for ( j = 19; j <= 23; j++ )
          s_ps( j,1,linvac,0x71 );
          muestra ( );
          int86 ( 0x16, &ent, &sal );
          switch ( selec )
           { case '1': case '3' : sal.h.al = 13; break;
             case '2': case '4' : sal.h.al = 27; break;
           } /** fin  switch(selec) **/
        } /** fin if_then(S/s) **/
        else  switch ( selec )
              { case '1': case '3' : sal.h.al = 13; break;
                case '2': case '4' : sal.h.al = 27; break;
              } /** fin  switch(selec) **/
    } /** fin if_then(j==1) **/
    else
     { s_offcur ( );
       s_box ( 20,2,22,36,0,0x47 );
       s_ps ( 21,3,"No se encuentra en el diccionario",0xC7 );
       delay ( 3000 );
       for ( j = 20; j <= 22; j++ )
         s_ps ( j,1,linvac,0x71 );
       s_ps ( 23,1,linvac,0x71 );
       s_ps ( 23,9,"Deseas checar el Dicc.(s/n): ?",0x07 );
       s_posyx ( 23, 38 );
       s_oncur ( );
       int86 ( 0x16, &ent, &sal );
       j = 4;
       if (( sal.h.al == 83 ) || ( sal.h.al == 115 )) /** 83/115 - S/s **/
         { s_offcur ();
           s_box ( 14,0,24,39,0,0x71 );
           s_ps ( 14,13,"Diccionario",0x51 );
           if ((palenc.sign - 4) <= 0)
             lonpal = 1;
             else
              if ((palenc.sign + 4) >= nelem)
                lonpal = nelem - 8;
                else lonpal = palenc.sign - 4;
           s_offcur ();
           k = 0;
           do
            { if ( sal.h.al == 13 )
                { s_offcur ( );
                  for ( i = 0; locpal [lonpal+j]->palb [i] != '\0'; i++ )
                    palenc.palb [i] = locpal [lonpal+j]->palb [i];
                  palenc.palb [i]  = '\0';
                  for ( i = 0; i <= 1; i++ ) palenc.arch [i] = locpal [lonpal+j]->arch [i];
                  for ( i = 0; i <= 3; i++ ) palenc.pos  [i] = locpal [lonpal+j]->pos  [i];
                  s_ps ( 23,1,linvac,0x71 );
                  s_ps ( 23,33,"Espera",0x97 );
                  obten ( );
                  muestra ( );
                  int86 ( 0x16, &ent, &sal );
```

```
        sal.h.al = 13;
        k = 0;
      } /** if_then (sal.h.al==ESC) **/
      else
       switch ( sal.h.ah )
        { case 0x50 : if ( j == 7 )
          /* Sube */    if ( lonpal+j < nelem )
                        lonpal += 1;
                        else lonpal = nelem - j;
                      else if ( j <= 6 )
                           { j += 1;
                             s_posyx ( 14+j, 1 );
                           }
                    break;
          case 0x48 : if ( j == 0 )
          /* Baja */    if ( lonpal > 1 )
                        lonpal -= 1;
                        else lonpal = 1;
                      else if ( j >= 1 )
                           { s_posyx ( 14+j, 1 );
                             j -= 1;
                           }
                    break;
          case 0x51 : if ( lonpal+j+7 >= nelem )
          /* Pg baja */ lonpal = nelem - 7;
                      else
                       { lonpal += j;
                         j = 7;
                       }
                    break;
          case 0x49 :  if ( lonpal+j-7 <= 1 )
          /* Pg sube */  lonpal = 1;
                         else
                          { lonpal += j - 7;
                            j = 0;
                          }
                    break;
          default   :  break;
         } /** fin switch(sal.h.al) **/
       s_offcur ();
       if ((j == 7 ) || ( j == 0 ) || ( k == 0 ))
       for ( i = 0; i <= 7; i++ )
         { for ( k = 0; locpal [lonpal+i]->palb [k] != '\0'; k++ )
             palenc.palb [k] = locpal [lonpal+i]->palb [k];
           palenc.palb [k] = '\0';
           s_ps ( 15+i,1,linvac,0x70 );
           s_ps ( 15+i,1,palenc.palb,0x70 );
         }
       if (( sal.h.al == 13 )||( sal.h.al == 83 )||( sal.h.al == 115 ))
         { s_ps ( 23,1,linvac,0x70 );
           s_ps ( 23,1,"ESC salir / ^Y^X moverse / RETURN elegir",0x07 );
         }
       s_posyx ( 15+j, 1 );
       s_oncur ();
       int86 ( 0x16, &ent, &sal );
     } /** fin while(sal.h.al=ESC) **/
    while (sal.h.al != 27);
    s_box ( 14,0,24,39,0,0x71 );
    switch ( selec )
     { case '1' : case '2':  s_ps ( 14,13,"Diccionario",0x71 ); break;
       case '3' : case '4':  s_ps ( 14,14,"Ortografía",0x71 );   break;
     } /** fin  switch(selec) **/
```

```
            for ( j = 15; j <= 23; j++ )
              s_ps( j,1,linvac,0x71 );
            switch ( selec )
             { case '1' :
               case '3' : do
                          { s_ps( 15,1,"Dame la palabra",0x71 );
                            s_ps( 15,16,":",0x71 ); s_ps( 15,17,palab,0x17 );
                            s_ps ( 23,1,linvac,0x71);
                            s_ps ( 23,14,"ESC salir / RETURN seguir",0x07 );
                            s_posyx ( 23, 24 );
                            int86 ( 0x16, &ent, &sal );
                          } /** fin while(sal.h.al=ESC y RETURN) **/
                          while (( sal.h.al != 27 ) && ( sal.h.al != 13 ));
                          break;
               case '2' :
               case '4' : sal.h.al = 27; break;
             } /** fin  switch(selec) **/
           } /** fin if_then(S/s) **/
           else switch ( selec )
                { case '1':  case '3' :  sal.h.al = 13; break;
                  case '2':  case '4' :  sal.h.al = 27; break;
                } /** fin  switch(selec) y if_else(S/s) **/
        } /** fin if_else(j==1) **/
      } /** fin if_then(lonpal!=0) **/
      else
       if ( j >= 0 )
        do
         { s_ps ( 23,1,linvac,0x71);
           s_ps ( 23,14,"ESC salir / RETURN seguir",0x07 );
           s_posyx ( 23, 24 );
           int86 ( 0x16, &ent, &sal );
         } /** fin while(sal.h.al=ESC y RETURN) **/
        while (( sal.h.al != 27 ) && ( sal.h.al != 13 ));
        else
         { s_offcur ();
           s_box ( 17,6,20,32,0,0x47 );
           s_ps ( 17,16,"AVISO",0xC7 );
           s_posyx ( 18,7 ); printf ("%c",palab[lonpal-2] );
           s_ps ( 18,8," Es un caracter NO vali-",0x47 );
           s_ps ( 19,7,"do dentro de la palabra. ",0x47 );
           sound ( 45); delay (400); nosound ();
           delay ( 6000 );
           if (( selec == 2 ) || (selec == 4))
             sal.h.al = 27;
         }
    } /** fin while (sal.h.al=ESC) **/
    while ( sal.h.al != 27 );
  } /**  fin if_then(nelem>0) **/
  else
   { nelem = abs ( nelem );
   } /**  fin if_else(nelem>0) **/
 for ( i = 14; i <= 23; i++ )
   s_ps ( i,1,linvac,atri );
 s_box ( 0,0,24,39,0,0x17 );
 s_oncur ();
 return ( nelem );
} /** fin diccort **/
```

V.2.3.11 Subrutina Llena()

Esta subrutina se encarga de leer el archivo de índices de los términos del diccionario y colocarlos en la memoria principal de la computadora.

```
/*****************************************************************************/
/****                      llena ( )                                  ****/
/****  Subrutina que se encarga de llenar cada uno de los elementos   ****/
/****  que forman el vector donde se realiza la busqueda. Dichos      ****/
/****  elementos estan formados por : Palabra, Archivo y Posición.    ****/
/*****************************************************************************/
int llena ( )
{ int       cont; /** variable que lleva el conteo de palabras leidas **/
  FILE      *ap_dicc; /* apuntador a archivo de entrada --dicctot.dic-- */
  char      cadent [NUMCARBUS],
           *existdicc,
            i;
  int       ban, mini, mfin;

  cont = 0;
  if (( existdicc = searchpath ( "dicctot.dic" )) != NULL )
    { if ( (ap_dicc = fopen ( existdicc, "r" )) != NULL )
      { mini = farcoreleft()/1024;
        if ( mini >= 1 ) /** 200 )   **/
          { s_box ( 18,9,20,30,0,0x17 );
            s_ps ( 19,10,"leyendo: ",0x87 );
            s_ps ( 19,19,"Diccionario",0x07 );
            do
            { ban = 0;
              if (cont <= NUMELEMAR - 2)
                { if (fgets (cadent, NUMCARBUS, ap_dicc) != NULL)
                  { i = strlen ( cadent );
                    if ((locpal [cont] = (struct elem far *) farmalloc (2 + i)) != NULL)
                      { locpal [cont]->arch [0] = cadent [ban++];
                        locpal [cont]->arch [1] = '\0';
                        for ( i = 0; i <= 2; i++)
                        locpal [cont]->pos [i] = cadent [ban++];
                        locpal [cont]->pos [i] = '\0';
                        for ( i = 0; cadent [ban] != '\n'; i++ )
                        locpal [cont]->palb [i] = cadent [ban++];
                        locpal [cont]->palb [i] = '\0';
                        cont++;
                      } /** fin if_then(farmalloc) **/
                      else
                       { s_box ( 18,8,20,30,0,0x47 );
                         s_ps ( 18,16,"AVISO",0xC7 );
                         s_ps ( 19,9,"Memoria insuficiente",0x47 );
                         for ( ban = cont - 1; ban >= 0; ban--)
                           farfree ( locpal [ban] );
                         sound (45); delay (400); nosound ();
                         delay ( 3000 );
                         ban = -1;  cont = 1;
                       } /** fin if_else(farmalloc) **/
                  } /** fin if_then(fgets) **/
                  else { ban = -1; }
                  /**  { s_box ( 18,2,22,34,0,0x47 );                    **/
                  /**    s_ps ( 18,16,"AVISO",0xC7 );                    **/
                  /**    s_ps ( 19,3,"  Existe un error de lectura en",0x47 );  **/
                  /**    s_ps ( 20,3,"el archivo que contiene el dic-",0x47 );  **/
                  /**    s_ps ( 21,3,"cionario.  -  dicctot.dicc  -  ",0x47 );  **/
```

```
        /**   if ( cont > 2 )                                  **/
        /**     { for ( ban = cont - 1; ban >= 0; ban--)       **/
        /**         farfree ( locpal [ban] );                  **/
        /**     }                                            **/
        /**   sound (45); delay (400); nosound ();           **/
        /**   delay ( 3000 );                                **/
        /**   ban = -1; cont = 1;                            **/
        /** } /** fin if_else(fgets)                         **/
       } /** fin if_then(cont<NUMELEMAR) **/
       else
        { s_box ( 18,2,20,37,0,0x47 );
       s_ps ( 18,16,"AVISO",0xC7 );
       s_ps ( 19,3," El Diccionario NO fue cargado en",0x47 );
       s_ps ( 20,27,"  ",0x07 );
       s_ps ( 20,3,"su totalidad, solamente ",0x47 );
       s_posyx ( 20,27 ); printf ("%4d",cont );
       s_ps ( 20,31," pala-",0x47 );
       s_ps ( 21,3,"bras. Capacidad máxima ** 6400 **.",0x47 );
       sound (45); delay (400); nosound ();
       delay ( 6000 );
       ban = -1;
       } /** fin if_else(cont<NUMELEMAR)  **/
     } /** fin while **/
      while ( ban > 0 );
      cont--;
    } /** fin if (mini> =200) **/
    else
     {  s_box ( 18,8,20,30,0,0x47 );
    s_ps ( 18,16,"AVISO",0xC7 );
    s_ps ( 19,9,"Memoria no suficiente",0x47);
    sound (45); delay (400); nosound ();
    delay ( 3000);
    ban = -1; cont = 1;
     }
   fclose ( ap_dicc );
  } /** fin if_then(fopen) **/
  else  { ban = 1; abandonar ( existdicc, ban ); }
 } /** fin if_then(existdicc!= NULL) **/
  else  { ban = 2; abandonar ( "dicctot.dic", ban ); }
for ( i = 18; i <= 22; i++ )
  s_ps ( i,1,linvac,0x71 );
return ( cont );
} /**  fin función llena  **/
```

V.2.3.12 Subrutina Obten()

Esta subrutina se encarga de obtener de los archivos el significado de una palabra, para ello es necesario indicar la siguiente información: a)nombre del archivo donde se encuentra el significado de la palabra b)la posición donde comienza el significado de la palabra c)la palabra misma con el fin de cotejarla.

```
/*************************************************************************/
/****                    obten ( )                                  ****/
/****   Subrutina que obtiene palabra y significado del archivo.    ****/
/*************************************************************************/
obten ( )
{ FILE      *ap_dicc; /* apuntador a archivo de entrada -- diccO?.dic-- */
```

```
char      cadent [NUMCARENT],
          archent [] = "dicc00.dic",
          *existe;
int       i, ban;
unsigned  base;
long int  lugar;

base = 1; lugar = 0;
for ( i = 0; i < = NUMCARSIG; i++ )
  palsig.sig [i]  = '\0';
for ( i = 0; archent [i] != '.'; i++ ) ();
archent [ i - 1 ] = palenc.arch [0];
if (( existe = searchpath ( archent )) == NULL )
  { ban = 2; abandonar ( archent, ban ); }
if (( ap_dicc = fopen ( existe, "r" )) == NULL )
  { ban = 1; abandonar ( existe, ban ); }
for ( i = 2; i > = 0; i-- )
  { lugar += base * ( palenc.pos [i] - 48 );
  base *= 50;
  } /** fin for **/
fseek ( ap_dicc, lugar, 0 );
fread ( cadent, NUMCARENT, 1, ap_dicc );
for ( i = 0; cadent [i] != ' '; i++ )
  palsig.pal [i] = cadent [i];
palsig.pal [i] = '\0';
ban = i + 1;
for ( i = 0; cadent [ban + i] != '\n'; i++ )
  palsig.sig [i] = cadent [ban + i];
palsig.sig [i++] = '\0';
fclose ( ap_dicc );
} /** fin obten **/
```

V.2.3.13 Subrutina Busca()

Esta subrutina se encarga de verificar en el diccionario la existencia de una palabra. En caso de que dicha palabra se encuentre en la lista, devolverá un uno, la palabra, nombre del archivo y posición donde se localiza el significado de dicha palabra. En caso negativo, regresará el valor cero y la posición previa o posterior a la palabra buscada.

```
/*************************************************************************/
/****                    busca ( )                                  ****/
/****  Subrutina que verifica si la palabra se encuentra en el dic-  ****/
/****  cionario, en caso afirmativo regresara un uno y obtendra :    ****/
/****  el archivo y la posición de la palabra dentro de dicho archivo. ****/
/****  En caso negativo regresara un cero.                          ****/
/*************************************************************************/
busca ( palab, fin, tmp )
char      palab [NUMCARPAL]; /** Palabra a buscar en el arreglo **/
int       fin,               /** Num. de elementos del arreglo  **/
          tmp;       /** Variable que regresara un :          **/
                     /** 1 si la palab. se existe en el dicc. **/
                     /** 0 si la palab. no existe en el dicc. **/
{ int     ini, /** Contendra el inicio del arreglo.            **/
          piv, /** Sera el pivote dentro del arreglo.          **/
          ban, /** Contendra el valor de la comparación.       **/
          rep, /** Guardara el num. mayor de caract. semejantes entre  **/
               /** las palabras comparadas, para obtener la palabra    **/
               /** más cercana a la palabra buscada.           **/
```

```
        pal; /** Guarda la posición de la palabra más cercana a la  **/
             /** palabara buscada.                                  **/

 ini = piv = ban = pal = 1;
 rep = tmp = 0;
 while ( ban != 0 )
  { if ( ini > fin )
     { if ( pal != 0)
        { for ( tmp = 0; locpal [pal]->palb [tmp] != '\0'; tmp++)
            palenc.palb [tmp] = locpal [pal]->palb [tmp];
          palenc.palb [tmp]  = '\0';
          for ( tmp = 0; tmp <= 1; tmp++)
            palenc.arch [tmp] = locpal [pal]->arch [tmp];
          for ( tmp = 0; tmp <= 3; tmp++)
            palenc.pos  [tmp] = locpal [pal]->pos [tmp];
          palenc.sign = pal;
        }
        else
        { for ( tmp = 0; locpal [pal]->palb [tmp] != '\0'; tmp++)
             palenc.palb [tmp] = locpal [pal]->palb [tmp];
          palenc.palb [tmp]  = '\0';
          palenc.pos  [0] = '\0';
          palenc.arch [0] = '\0';
          palenc.sign = pal;
        }
       tmp = 0;
       s_oncur ();
       break;
     }
     else  piv = ( ini + fin ) / 2;
    ban = compara ( palab, locpal [piv]->palb, ban );
    if ((tmp = abs (ban)) >= rep )
     { rep = tmp; /** número mayor de caracteres      **/
       pal = piv; /** posición de palabra más cercana **/
     }
    if ( ban < 0 )
     fin = piv - 1;
     else if ( ban > 0 )
          ini = piv + 1;
          else
          { for ( tmp = 0; locpal [piv]->palb [tmp] != '\0'; tmp++)
              palenc.palb [tmp]  = locpal [piv]->palb [tmp];
            palenc.palb [tmp]  = '\0';
            for ( tmp = 0; tmp <= 1; tmp++)
              palenc.arch [tmp] = locpal [piv]->arch [tmp];
            for ( tmp = 0; tmp <= 3; tmp++)
              palenc.pos [tmp] = locpal [piv]->pos [tmp];
            tmp = 1; posi = piv;
          } /** fin if_else(ban>0) **/
 } /** fin while (ban!=0) **/
 return ( tmp );
} /** fin función busca  **/
```

V.2.3.14 Subrutina Compara()

La tarea de esta subrutina consiste en comparar dos palabras que le son proporcionadas como información. Dicha comparación se realiza carácter por carácter. Si la palabra uno es menor que la palabra dos, la subrutina compara() nos regresara un valor negativo. Si la palabra uno es igual

a la palabra dos, dicha subrutina nos regresará un valor cero. Si la palabra uno es mayor que la palabra dos, nos regresará un valor positivo.

```
/*************************************************************************/
/****                    compara ( )                                  ****/
/****    En esta subrutina se realiza la comparación de palabras,     ****/
/****    obteniendose como resultado :                                ****/
/****                                                                 ****/
/****      (-)    si la palabra 1 es menor que la palabra 2           ****/
/****       0     si la palabra 1 es igual que la palabra 2           ****/
/****      (+)    si la palabra 1 es mayor que la palabra 2           ****/
/****                                                                 ****/
/****    y como entrada : las dos palabras a comparar, una variable   ****/
/****    entera que contendra el resultado.                           ****/
/*************************************************************************/
compara ( palab1, palab2, valcomp )
char     palab1 [ NUMCARPAL ]; /* Contiene la palabra 1          */
char far  *palab2;             /* Contiene la palabra 2          */
int      valcomp;              /* Bandera de comparación de palabras. */
{ int    pivpalb,  /* Se usara como pivote dentro de la palabra.  */
         b1,       /* Bandera de comp. de caract. en palabra 1.   */
         b2,       /* Bandera de comp. de caract. en palabra 2.   */
         signo,    /* Bandera que nos indica los signos de ortografía. */
         aux;
  char   palb1 [ NUMCARPAL ],  /* aux. que contendra la palabra 1   */
         palb2 [ NUMCARPAL ]; /* aux. que contendra la palabra 2   */

  strcpy ( palb1, palab1 );
  for ( aux = 0; palab2 [aux] != '\0'; aux++ )
    palb2 [aux] = palab2 [aux];
  palb2 [aux] = '\0';
  valcomp = pivpalb = signo = aux = 0;
  while  (( valcomp == 0 ) && ( aux != 1 ))
   { if (( palb1[pivpalb] == '\0' ) || ( palb2[pivpalb] == '\0') ||
         ( palb1[pivpalb] == '\n' ) || ( palb2[pivpalb] == '\n' ))
     { if ((( palb1[pivpalb] == '\0' ) || ( palb1[pivpalb] == '\n' )) &&
           (( palb2[pivpalb] != '\0' ) && ( palb2[pivpalb] != '\n' )))
         valcomp = -1;
         else if ((( palb2[pivpalb] == '\0' ) || ( palb2[pivpalb] == '\n' )) &&
                  (( palb1[pivpalb] != '\0' ) && ( palb1[pivpalb] != '\n' )))
                 valcomp = 1;
                 else  aux = 1;
     } /** fin if_then **/
     else
      { b1 = b2 = 0;
        if ((palb1[pivpalb] > 'z') || (palb1[pivpalb] < 'a')) b1 = checa ( palb1, b1, pivpalb );
        if ((palb2[pivpalb] > 'z') || (palb2[pivpalb] < 'a')) b2 = checa ( palb2, b2, pivpalb );
        if ((palb1[pivpalb]) > (palb2[pivpalb]))
          valcomp = 1;
          else if (( palb1[pivpalb] ) < ( palb2[pivpalb] ))
                valcomp = -1;
                else /** palb1 == palb2 **/
                 { if ( b1 == b2 )
                     valcomp = 0;
                     else if ( b1 > b2 )
                          if ( signo == 0 )
                           switch ( b1 )
                            { case 1  : valcomp = 0; signo = 1; break;
                              case 2  : valcomp = 0; signo = 2; break;
                              default : valcomp = 1; break;
                            }
```

```
else
 switch ( signo )
  { case 1 : switch ( b1 )
          { case 1  : valcomp = 0; signo = 5; break;
            case 2  : valcomp = 0; signo = 6; break;
            default : valcomp = 1; break;
          } break;
    case 2 : switch ( b1 )
          { case 1  : valcomp = 0; signo = 6; break;
            case 2  : valcomp = 0; signo = 2; break;
            default : valcomp = 1; break;
          } break;
    case 3 : switch ( b1 )
          { case 1  : valcomp = 0; signo =  7; break;
            case 2  : valcomp = 0; signo =  9; break;
            default : valcomp = 1; break;
          } break;
    case 4 : switch ( b1 )
          { case 1  : valcomp = 0; signo =  8; break;
            case 2  : valcomp = 0; signo = 10; break;
            default : valcomp = 1; break;
          } break;
    default : switch ( b1 )
          { case 1  : valcomp = 0; break;
            case 2  : valcomp = 0; break;
            default : valcomp = 1; break;
          } break;
  } /** fin switch(signo) **/
else if ( b1 < b2 )
     if ( signo == 0 )
       switch ( b2 )
        { case 1  : valcomp =  0; signo = 3; break;
          case 2  : valcomp =  0; signo = 4; break;
          default : valcomp = -1; break;
        }
       else switch ( signo )
          { case 1 : switch ( b2 )
                  { case 1  : valcomp =  0; signo = 7; break;
                    case 2  : valcomp =  0; signo = 8; break;
                    default : valcomp = -1; break;
                  } break;
            case 2 : switch ( b2 )
                  { case 1  : valcomp =  0; signo =  9; break;
                    case 2  : valcomp =  0; signo = 10; break;
                    default : valcomp = -1; break;
                  } break;
            case 3 : switch ( b2 )
                  { case 1  : valcomp =  0; signo = 11; break;
                    case 2  : valcomp =  0; signo = 12; break;
                    default : valcomp = -1; break;
                  } break;
            case 4 : switch ( b2 )
                  { case 1  : valcomp =  0; signo = 12; break;
                    case 2  : valcomp =  0; signo =  4; break;
                    default : valcomp = -1; break;
                  } break;
            default : switch ( b2 )
                   { case 1  : valcomp =  0; break;
                     case 2  : valcomp =  0; break;
                     default : valcomp = -1; break;
                   } break;
          } /** fin switch(signo) **/
```

```
            }; /** fin else (palb1 = =palb2) **/
        pivpalb + +;
      } /** fin else **/
  } /** fin while **/
  if (pivpalb - 1) > 0)
    valcomp *= pivpalb - 1;
  palenc.sign = signo;
  return ( valcomp );
} /** fin compara **/
```

V.2.3.15 Subrutina Checa()

El objetivo de esta subrutina es estandarizar caracteres especiales -tales como diéresis, acentos y eñes- a caracteres estándares. Y convierte caracteres mayúsculas a minúsculas.

```
/*****************************************************************************/
/****                    checa ( )                                        ****/
/****  Esta subrutina realiza el cambio de letras mayusculas              ****/
/****  a minusculas, asi como el cambio de caracteres especiales a        ****/
/****  caracteres estandares para facilitar su comparación e indicar      ****/
/****  que tipo de caracter espaecial se trata.                           ****/
/*****************************************************************************/
int checa ( s, b, n )
char     s [ NUMCARPAL ];
int      b, n;
{                                  /* b =  0 -- nada.        */
  b = 0;                             /*     1 -- acento.      */
  if ((s [n] >= 'A') && (s [n] <= 'Z'))    /*     2 -- dieresis.    */
    s [n] = s [n] + 32;                 /*     3 -- Ñ, ñ.        */
    else                             /*     4 -- guion.       */
     { switch  ( s [n] )               /*     5 -- no definido. */
       { case 'á' : s [n] = 'a'; b = 1; break;
         case 'é' : s [n] = 'e'; b = 1; break;
         case 'í' : s [n] = 'i'; b = 1; break;
         case 'ó' : s [n] = 'o'; b = 1; break;
         case 'ú' : s [n] = 'u'; b = 1; break;
         case 'ü' : s [n] = 'u'; b = 2; break;
         case 'Ü' : s [n] = 'u'; b = 2; break;
         case 'ñ' : s [n] = 'n'; b = 3; break;
         case 'Ñ' : s [n] = 'n'; b = 3; break;
         case '-' : b = 4; break;
         default  : b = 5; break;
       }  /* fin switch */
     }  /* fin else */
  return ( b );
}  /* fin checa */
```

V.2.3.16 Subrutina Erracen()

La tarea de esta subrutina consiste en enviar los mensajes de errores de acentuación: falta o sobra de puntos de acentuación. Dichas faltas se detectan en el monento de realizaar la busqueda de la palabra en el acervo del diccionario.

```
/*****************************************************************************/
```

```
/****                    erracen ( )                    ****/
/**** Se encarga de mostrar en una ventana el error de acentuación ****/
/*************************************************************************/
erracen ( )
{  switch ( lonpal)
   { case  1 : s_box ( 20,12,22,25,0,0x47 );
               s_ps ( 21,13,"Sobra acento",0xC7);
               break;
     case  2 : s_box ( 20,11,22,27,0,0x47 );
               s_ps ( 21,12,"Sobran diresis",0xC7);
               break;
     case  3 : s_box ( 20,12,22,25,0,0x47 );
               s_ps ( 21,13,"Falta acento",0xC7);
               break;
     case  4 : s_box ( 20,11,22,27,0,0x47 );
               s_ps ( 21,12,"Faltan diresis",0xC7);
               break;
     case  5 : s_box ( 20,11,22,26,0,0x47 );
               s_ps ( 21,12,"Sobran acentos",0xC7);
               break;
     case  6 : s_box ( 20,6,22,31,0,0x47 );
               s_ps ( 21,7,"Sobran diresis y acento",0xC7);
               break;
     case  7 : s_box ( 20,8,22,28,0,0x47 );
               s_ps ( 21,9,"Acento mal colocado",0xC7);
               break;
     case  8 : s_box ( 20,3,22,34,0,0x47 );
               s_ps ( 21,4,"Sobra acento y faltan diresis",0xC7);
               break;
     case  9 : s_box ( 20,4,22,35,0,0x47 );
               s_ps ( 21,5,"Falta acento y sobran diresis",0xC7);
               break;
     case 10 : s_box ( 20,7,22,30,0,0x47 );
               s_ps ( 21,8,"Diresis mal colocados",0xC7);
               break;
     case 11 : s_box ( 20,11,22,26,0,0x47 );
               s_ps ( 21,12,"Faltan acentos",0xC7);
               break;
     case 12 : s_box ( 20,7,22,31,0,0x47 );
               s_ps ( 21,8,"Falta acento y diresis",0xC7);
               break;
   } /** fin switch(palenc.sig) **/
}
```

V.2.3.17 Subrutina Muestra()

Esta subrutina se encarga de mostrar en la ventana del diccionario la palabra seleccionada junto con su significado, se encarga también de realizar la justificación izquierda y derecha para evitar que las palabras que recibe a través de la subrutina Obten() queden cortadas. Esta justificación se realiza en un espacio de 35 caracteres por línea, y se tienen 8 líneas por ventana.

```
/*************************************************************************/
/****                    muestra ( )                    ****/
/**** Se encarga de mostrar en una ventana el significado de una pa_  ****/
/**** labra que se encuentra en la variable palsig.sig.              ****/
/*************************************************************************/
muestra ( )
{ int psig,   /** Var. que sirve de pivote en el string lin36c     **/
```

```
    ppsig,  /** Var. que sirve de pivote en el string palsig.sig  **/
    lonpal, /** Longitud de palabra                              **/
    lin,    /** Contador de linea                               **/
    i, j;
char   lin36c [ LIMCOL ]; /** Linea de impresión de 36 caracteres     **/

lin = 16; ppsig = 0;
for ( j = 15; j <= 23; j++ )
  s_ps( j,1,linvac,0x71 );
s_ps ( 15,2,palsig.pal,0x17 );
lonpal = strlen ( palsig.pal );
s_ps ( 15,lonpal+3,":",0x17 );
for ( psig = 0; psig <= 2; psig++ ) /** Se coloca una sangria de dos  **/
  lin36c [ psig ] = ' ';          /** lugares en blanco.             **/
lonpal = strlen ( palsig.sig ) + psig;
do
 { j = i = 0;
   if (( lonpal - 35 ) < 0 )
   i += lonpal - psig;
   else  i += 35 - psig;
   if ((palsig.sig [ppsig+i] != ' ')&&(palsig.sig [ppsig+i] != '\0'))
   do
    i--;
   while ((palsig.sig [ppsig+i] != '\0')&&(palsig.sig [ppsig+i] != ' '));
   j = 35 - i - psig;
   lonpal -= i; i--;
   while (i >= 0)
    { lin36c [psig++] = palsig.sig [ppsig];
   if ( palsig.sig [ppsig] == ' ' )
     if ( j > 0 )
       if ((psig >= 30)&&(j >= 2))
         { lin36c [psig++] = ' ';
         lin36c [psig++] = ' ';
         j -= 2;
          } /** fin if_then(psig>=30)&&(j>=2)  **/
          else
         { lin36c [psig++] = ' ';
           j--;
         } /** fin if_else(psig>=30)&&(j>=2)  **/
   ppsig++; i--;
    } /** fin while ((psig)&&(ppsig))  **/
   lin36c [psig++] = '\0';
   if (lin <= 22 )
   { s_ps ( lin,1,linvac,0x71);
     s_ps ( lin++,2,lin36c,0x71 );
   }
    psig = 0;
    ppsig++;
  } /** fin while(fin sig) **/
 while ( lonpal > 0 );
 s_ps ( 23,2,"Presione cualquier tecla para seguir",0x07 );
} /** fin muestra  **/
```

CONCLUSIONES

Al concluir esta tesis es posible corroborar la importancia que tiene un método o regla para el desarrollo de sistemas. En nuestro caso el marco de desarrollo ofreció una gama de posibilidades y retos, con una rica retroalimentación en conocimientos.

La base empieza desde la propia selección del sistema, que debe cubrir un conjunto de necesidades. Además de facilitar, mejorar y solucionar una actividad específica.

En nuestro caso, la elaboración de un procesador de texto para niños, habilita la posibilidad de usar y manejar una herramienta de trabajo en un sector poco atendido.

Como segundo paso está el planteamiento y análisis de la problemática. La cual determina los puntos globales que se deben considerar para diseñar el sistema.

Dentro del mismo entorno está la elección preliminar del lenguaje de programación y el equipo de cómputo adecuado, acorde a las particularidades del sistema. En nuestro caso, predeterminó al lenguaje C y a las computadoras personales (microcomputadoras) como los más viables.

Con estas bases se emprendió su diseño, desarrollo y puesta en marcha. Retroalimentando su esquema en cada una de sus pruebas, con el conjunto de aportaciones que daban niños, maestros y conocedores del área educativa.

Entre ellas se encuentra, un menú con figuras alusivas a la función, letras de mayor tamaño, una sección de ayuda, mensajes que alerten o indiquen algún problema, evasión de

mensajes del sistema operativo, advertir si no se ha guardado un documento, indicar con que documento se trabaja, etc.

Finalmente, el producto pretende no solamente ayudar al niño en la elaboración e impresión de documentos, sino también introducirlo al uso de software. Manejando pocas teclas de funciones y usando estándares semejantes a paquetes que llegará a usar más adelante.

Dentro de sus características el procesador tiene la opción de ejecutar las funciones desde el menú principal (que contiene figuras), desde la ventana de ayuda que proporciona la edición y directamente en edición. Esta característica prevee el dominio que puede llegar a desarrollar el niño.

En principio querrá auxiliarse del menú principal, después de la ayuda en edición y finalmente aprender alguna o la mayoría de funciones y hacerlo con agilidad a través de las teclas de función directa.

Como parte integral del procesador se incluye un diccionario con definición de términos, que representa un rasgo poco común en los procesadores de texto y un beneficio para el niño que lo consulte.

De esta manera, el sistema coadyuvará con el maestro a acrecentar el vocabulario del niño, a mejorar su sintaxis y a fomentar el hábito de consulta.

Cabe mencionar, que el presente procesador de texto se ha enfocado para niños que cursan los últimos años de primaria, principalmente quinto y sexto año.

BIBLIOGRAFIA

KERNIGHAN, Brian y Dennis M. Ritchie, El Lenguaje de Programación C, Prentice-Hall, Englewood Cliffs, México, 1985.

HERBERT, S., Programación en Turbo C, McGraw-Hill, California, 1991.

TANENBAUM, A.S.,Organización de Computadoras: Un enfoque estructurado, Prentice-Hall, Englewood Cliffs, México, 1986.

GARVIN, Paul L., Breve Introducción a la Computación Lingüistica, Universidad Nacional Mayor de San Marcos, Lima, 1969

ROCHKIND, Marc J., Advanced C Programming for Displays, Prentice-Hall, Englewood Cliffs, N. J., 1988.

KNUTH, Donald E., The art of Computer programming, USA, 1973.

ECKEL, Brouce, Using C ++, Osborne/McGraw-Hill, California, 1989.

KRUMM, Rob, Word Perfect 4.2, Mc Graw-Hill, Venezuela, 1988.

HOLTZ, Matthew, Mastering Microsoft Word on PC, SYBEX , 1984.

NEIBAUER, Alan R., The ABC's of Microsoft Word 3°, SYBEX, 1989.

DONOVAN, Jhon J., System Programming, Mc Graw-Hill, Kogakusha Ltd., Tokio, 1972.

GEAR, William C., Computer Organization and Programming, Mc Graw-Hill, USA, 1980.

JEMES, Martin, Computer Data-Base: Organization, Prentice-Hall, New Jersey, 1980.

SCHIDT, Heriberto, Turbo C the Pocket Reference, McGraw-Hill, California, 1988.

PHILLIP, Robinson, **"Word Procesor for PC"**, en BYTE, vol. 12 (17), USA, verano 1987 pp. 55 - 60.

LAMONT, Wool, **"Word Procesor for Desktop Publisher"**, en BYTE, vol. 13 (5), USA, Mayo 1988 pp. 171 - 176.