UNIVERSIDAD AUTONOMA DE NUEVO LEON
Secretaría Académica

REFORMA ACADEMICA DEL NIVEL MEDIO SUPERIOR

M3

Computación

Antología

COMPUTACIÓN, SEGUNDA EDICIÓN 1996

ÍNDICE

II PROGRAMACIÓN EN QuickBASIC

III VARIABLES DIMENSIONADAS

IV MANEJO DE ARCHIVOS

ANEXO

PRESENTACIÓN

Inmersos en el seguimiento de la Reforma Académica que la Universidad Autónoma de Nuevo León ha iniciado en las escuelas preparatorias con la finalidad de ir a la par en el avance cultural y tecnológico de la época actual, hemos elaborado este segundo texto de computación, el cual servirá de base y apoyo para que los alumnos continúen el aprendizaje de esta rama del conocimiento.

Este libro pretende introducir al alumno en el lenguaje llamado QuickBASIC, que es una de las herramientas más sencillas,útiles y rápidas para la programación.

El texto ha sido elaborado de acuerdo al programa aprobado por la Comisión Académica del H. Consejo Universitario de nuestra máxima casa de estudios.

Nuevamente reconocemos las aportaciones que los maestros del área han emitido a través de los coordinadores de la materia. Deseamos que el texto sea un apoyo real y efectivo en el proceso de enseñanza-aprendizaje del curso.

Comité de Computación

Ing. Felipe Rojas Patlán	Preparatoria Nº 2
Prof. Ricardo H. Álvarez Charles	Preparatoria Nº 7
Ing. Raúl Gallegos Cerda	Preparatoria Nº 15
Ing. Araceli Gpe. Álvarez Carvajal	Preparatoria Nº 16

PRÓLOGO

El comité de Computación desea hacer patente el agradecimiento a los maestros:

Profr.	Ricardo H. Álvarez Charles.	Prepa No. 7
Ing.	Sergio E. González González.	Prepa No. 9
Ing.	Raúl Gallegos Cerda.	Prepa No. 15
Ing.	Araceli Gpe. Álvarez Carvajal.	Prepa No. 16

Que hicieron posible la primera edición del libro de computación para el módulo tres desarrollando y explicando los conceptos de un lenguaje de programación (QB), así como los programas que sirvieron de ejemplo a dichos contenidos.

En esta segunda edición se han realizado correcciones y adecuaciones en base a los comentarios y sugerencias que nos han hecho llegar los maestros de las academias de las distintas preparatorias , tanto directamente como a través de los jefes de academia.

Los miembros del comité, Araceli, Raúl y Ricardo han analizado, conformado y adecuado todo este material que hemos recibido y que nos permite presentar esta segunda edición del libro de computación para el módulo tres, el cual es producto de las experiencias de todos y cada uno de los maestros formamos la academia general de computación a nivel preparatoria.

Nuevamente ponemos a la consideración de las academias este material, deseando que les sea útil en el desarrollo del trabajo y esperando todas las sugerencias, que en base a su experiencia, nos hagan llegar para enriquecer este libro.

Atentamente
Comité de Computación

INTRODUCCIÓN

Para el alumno:

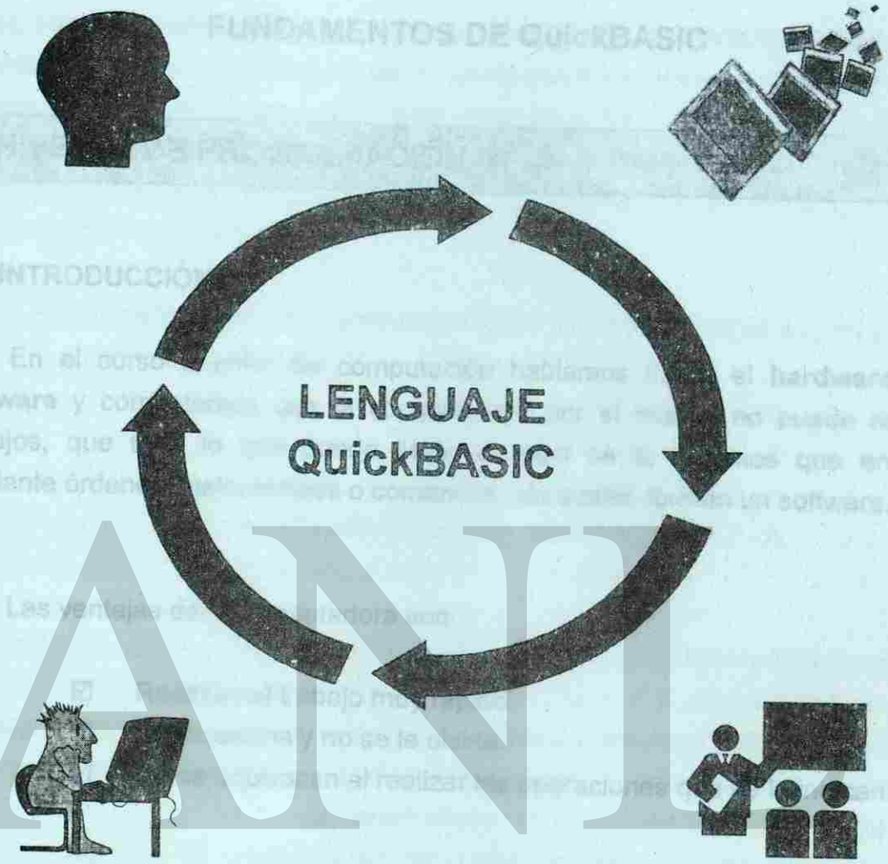
Tienes en tus manos este segundo texto de computación, el cual fue diseñado pensando en tu necesidad de enfrentarte a un mundo donde impera la rapidez y el dinamismo.

Nuestro propósito es darte a conocer, básicamente, el manejo del editor QuickBASIC, así como su aplicación en otras materias que conforman tu currícula escolar: matemáticas, física, etc... El conocimiento de este editor te llevará a manejar la computadora de manera más rápida y eficiente.

El texto está conformado por cinco unidades y una serie de ejercicios para reafirmar lo aprendido, tanto en el curso anterior como en el presente.

Piensa que siempre habrá una bibliografía que ampliará y reafirmará el contenido que ahora te presentamos y que el apoyo de tu maestro será de gran importancia para que alcances el éxito del curso que hoy inicias.

Los autores



FUNDAMENTOS DE QuickBASIC

UNIDAD I

FUNDAMENTOS DE QuickBASIC

1.- NIVELES DE PROGRAMACIÓN.

A.- INTRODUCCIÓN

En el curso anterior de computación hablamos sobre el **hardware** y el **software** y comentamos que la computadora por sí misma no puede realizar trabajos, que todo lo que puede llevar a cabo se lo tenemos que enseñar mediante órdenes, instrucciones o comandos, los cuales forman un software.

Las ventajas de la computadora son:

- ☑ Realizan el trabajo muy rápido.
- ☑ Lo almacena y no se le olvida.
- ☑ No se equivocan al realizar las operaciones que se le indican.

Los programas o software también los dividimos en tres tipos:

- ☑ Software del sistema operativo.
- ☑ Software en lenguaje de programación.
- ☑ Software en paquetes de aplicación.

En el siguiente tema hablaremos sobre los distintos tipos de lenguaje que existen para programar una computadora.

B.- LENGUAJES DE PROGRAMACIÓN.

En el libro anterior hablamos sobre la existencia de diferentes lenguajes para programar como el COBOL, PASCAL, FORTRAN, BASIC y utilizamos éste último, ahora hablaremos de los tipos de lenguajes o formas de instruir a nuestra computadora.

Los lenguajes para programar los clasificaremos en cuatro categorías:

a.- Lenguaje máquina.
b.- Lenguaje de bajo nivel.
c.- Lenguaje de alto nivel.
d.- Lenguaje de alto nivel con compilador

a.- Lenguaje máquina

Cuando las instrucciones están dictadas de manera que el **CPU** las entienda directamente sin necesidad de un traductor, es decir, que están en código binario (0,1) o bits, se les da el nombre de **lenguaje máquina.**

Ejemplos:	**CARACTER**	**BINARIO**
	A	01000001
	a	01100001
	3	00110011
	$	00100100
	+	00101011

Esta forma de representar las instrucciones es la única manera que entiende la computadora, pero para el programador esto es una tarea muy difícil.

b.- Lenguaje de bajo nivel

Por la dificultad que para el programador representa el lenguaje máquina, se buscaron otras formas para elaborar los programas, se necesitaba simplificar el proceso para que el programador pudiera recordar fácilmente cada una de las instrucciones, creándose los lenguajes llamados de **bajo nivel**, que utiliza símbolos alfabéticos o abreviaturas de las órdenes que se pretende que sean ejecutadas por la computadora.

Ejemplos:	**ORDEN**	**INSTRUCCIÓN**
	sumar	ADD
	restar	SUB
	multiplicar	MPY
	dividir	DIV

Estas abreviaturas (nemotecnias) son más fáciles de recordar por el programador que el conjunto de ceros y unos (0,1) o bits; una instrucción en este lenguaje podría ser **ADD x, y, z** lo que significa: sumar el valor de la variable "**x**" con el valor de la variable "**y**" y guarda el resultado en la variable "**z**".

Una vez que el programa se ha escrito en lenguaje de bajo nivel es necesario un traductor que convierta esas órdenes alfabéticas a ceros y unos (0,1), para que la máquina pueda entender y ejecutarlas, a este traductor se le llama **COMPILADOR**. Un compilador siempre escribirá un programa en lenguaje máquina.

c.- Lenguaje de alto nivel

El lenguaje donde las instrucciones, órdenes o comandos que se le dan a la computadora se hace en lenguaje casi cotidiano se le llama **lenguaje de alto nivel**, esto le facilita al programador la tarea de programar la computadora para que realice un trabajo o resuelva un programa.

Ejemplo:

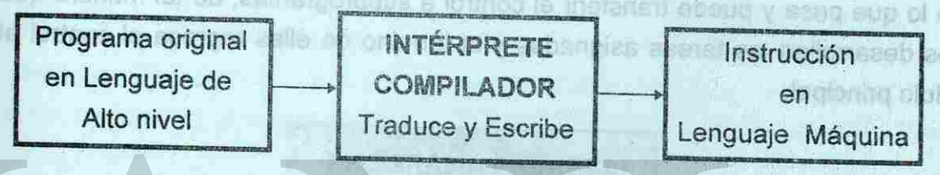
SI PROM ES MAYOR O IGUAL A 70 ENTONCES IMPRIME "APROBADO"
IF PROM >= 70 THEN PRINT "APROBADO"

Los programas en lenguaje de alto nivel no los puede interpretar la computadora, por lo que se requiere de un traductor que pueda hacer la conversión a código binario, o sea un compilador, a menos que el lenguaje provenga de un intérprete. Los intérpretes traducen y ejecutan una por una cada instrucción del programa de alto nivel; esto lo hacen cada vez que se ejecuta el programa y por lo tanto realiza el programa con cierta lentitud; la ventaja es que se puede hacer correcciones en el programa con mucha facilidad; cuando cometemos un error al dictarlo a la computadora, ésta, al ejecutarlo y encontrarlo, nos indica el tipo de error y dónde se ha cometido, como ocurre en el lenguaje BASIC.

La diferencia de un intérptrete y un compilador es que el primero no lo escribe, es decir solo lo lee y traduce cada vez que se corre; mientras el compilador lo lee, traduce y escribe, es decir queda traducido en unos y ceros en forma definitiva.

d.- Lenguaje de alto nivel con compilador

Los lenguajes de alto nivel con compilador son aquellos que tienen integrado un compilador, que al capturarse un programa en lenguaje de alto nivel, si no contiene errores de sintaxis, se traduce escribiendolo a lenguaje máquina (0,1) y, posteriormente, queda listo para ser ejecutado en cualquier momento y cuantas veces se desee.

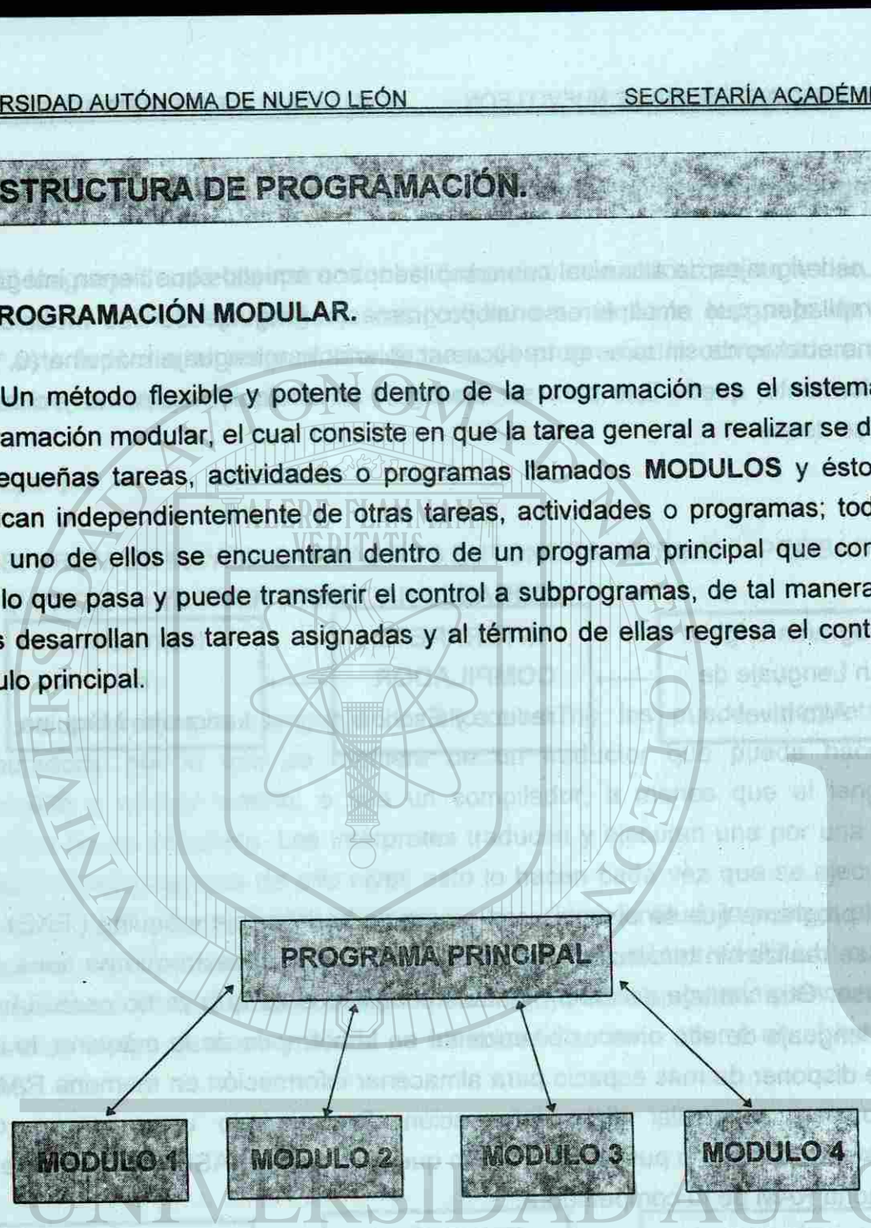


El programa que se ejecuta es el programa en lenguaje máquina (.EXE) por lo que se realiza sin tardanzas, con una rapidez mayor, puesto que no tiene que traducirse. Otra ventaja de los programas compilados es que ya no necesitamos que el lenguaje de alto nivel esté residente en la memoria de la máquina, lo que permite disponer de más espacio para almacenar información en memoria RAM o espacio para manipular dicha información. Por ejemplo un programa que realizaste en BASIC, lo puede ejecutar sin que el lenguaje BASIC se encuentre en la memoria RAM de tu computadora.

2.- ESTRUCTURA DE PROGRAMACIÓN.

A.- PROGRAMACIÓN MODULAR.

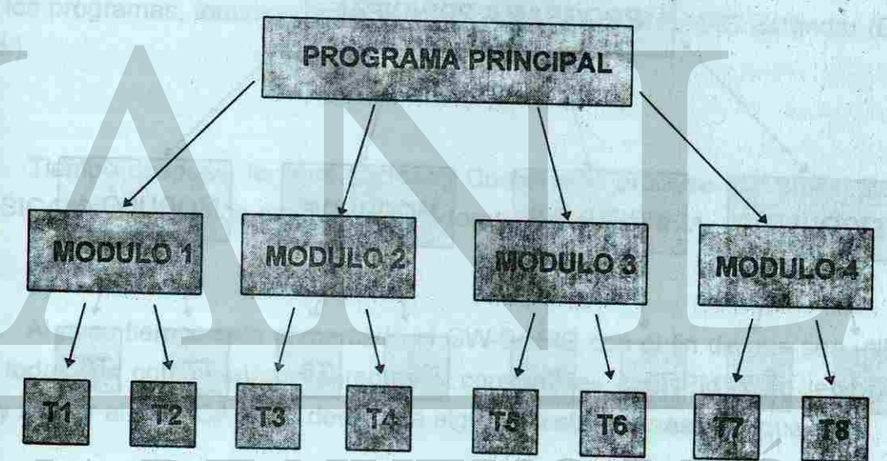
Un método flexible y potente dentro de la programación es el sistema de programación modular, el cual consiste en que la tarea general a realizar se divide en pequeñas tareas, actividades o programas llamados **MODULOS** y éstos se codifican independientemente de otras tareas, actividades o programas; todos y cada uno de ellos se encuentran dentro de un programa principal que controla todo lo que pasa y puede transferir el control a subprogramas, de tal manera que éstos desarrollan las tareas asignadas y al término de ellas regresa el control al módulo principal.



La tarea desarrollada en cada módulo puede ser de entrada, salida o manipulación de datos, o bien control de algunos submódulos.

B.- PROGRAMACIÓN DESCENDENTE.

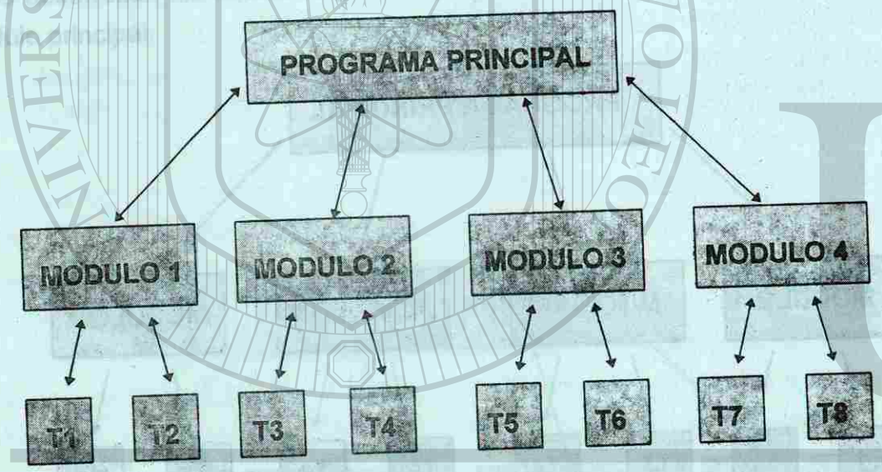
La programación descendente significa que a partir de la tarea general o global, ésta se descomponga en diferentes subtareas o módulos que, en conjunto, resuelvan el problema general y si alguna de estas subtareas o módulos es demasiado compleja, ésta deberá dividirse en otros módulos más simples que tengan tareas muy específicas para ejecutar.



C.- PROGRAMACIÓN ESTRUCTURADA.

En la programación estructurada, la tarea general o total se va a realizar utilizando las siguientes reglas:

- La solución total tiene un diseño modular.
- Los módulos son diseñados en modo descendente.
- En cada módulo sólo se utilizan las tres estructuras básicas de control: secuencial, selección y control (programación sin GOTO).



Los términos programación modular, programación descendente y programación estructurada , creados a mediados de los 60's, son utilizados indiscriminadamente por creer que significan lo mismo, pero como hemos visto cada uno tiene significado diferente. En el transcurso de este libro, iremos encaminando el curso a una programación estructurada, siguiendo los lineamientos que hemos mencionado anteriormente.

3.- CARACTERÍSTICAS DE QuickBASIC.

A.- DE BASIC A QuickBASIC.

📖 En el curso anterior utilizamos el lenguaje BASICA y/o GW-BASIC, los cuales tienen su origen en el BASIC (**B**eginner´s **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) creado en los inicios de los 60's en el Dartmounth College en Hanover, Estados Unidos.

📖 Posteriormente, en 1978, se establecen los requerimientos mínimos que debe tener todo lenguaje de computadora, con el fin de crear un estándar dentro de los programas; tomando estos requerimientos nace el BASIC Estándar (BASIC Std.).

📖 Tiempo después, la firma Microsoft Corporation produce su versión llamada BASICA para ser usada en las computadoras personales IBM.

📖 Al poco tiempo sale al mercado el GW-BASIC con el fin de que sea utilizado en todas las computadoras personales compatibles con IBM; este lenguaje es muy similar al BASICA, pero desarrolla algunas instrucciones más que éste.

📖 A finales de los 70's y principios de los 80's es creado el Quick BASIC con las características del editor intérprete GW-BASIC y las ventajas de un compilador.

B.- VENTAJAS DE QuickBASIC

Como ya mencionamos, existen intérpretes, traductores y compiladores; explicamos la diferencia entre ellos, indicando algunas ventajas y desventajas que tienen entre sí.

- Posee su propio compilador.
- Podemos editar, compilar, ejecutar, depurar y recompilar, si es necesario.
- El editor se ajusta a los estándares para la edición de textos y ofrece la posibilidad de que sea grabado en código ASCII.(Código internacional de letras y símbolos).
- Contiene un editor interactivo, el cual verifica la sintaxis de cada palabra reservada en el momento de teclearla; si es correcta, se traduce(n) la(s) instrucción(es) de la línea al lenguaje máquina; si le falta algún espacio lo corrige automáticamente asi como la sintaxis de una instrucción simple, si se cometió un error que no pueda corregir, entonces nos indica el lugar y tipo de error que se cometió.
- Se pueden elaborar programas con módulos ya elaborados anteriormente con sólo agregar los nombres de los módulos (subrutinas) correspondientes.
- No se necesitan números de línea.
- Ofrece la posibilidad de hacer ejecutable un programa, esto es, que se ejecuten desde el sistema operativo (programas .EXE).
- Los programas tecleados en BASIC, BASICA o GW-BASIC se pueden correr en QuickBASIC siempre y cuando sean grabados en ASCII.
- Las funciones LIST, EDIT, RENUM y AUTO no existen en Quick BASIC, ya que no son necesarias.

Estas son algunas de las ventajas que hacen que Quick BASIC sea rápido de ejecutar, de ahí su nombre.

C.- ELEMENTOS DE QuickBASIC.

Para agilizar y obtener los mejores resultados de la programación es conveniente que nuestro software cuente con :

- Un buen editor que permite elaborar un programa fuente.
- Un manejador de archivos como lo es el sistema DOS 5.0 u otra versión más nueva.
- Un compilador de lenguaje que permite traducir rápida y fácilmente al lenguaje máquina.
- Un depurador de programas que permite hacer las correcciones y/o ajustes a nuestros programas de una manera fácil.

A estas cuatro acciones algunos autores les han llamado entorno de programación .

Podemos decir que el QuickBASIC es un software que cumple todo lo anterior, ya que cuenta con las características antes mencionadas.

Otra ventaja del QuickBASIC es que posee características e instrucciones que permiten estructurar mejor los programas que en BASIC.

NOTA: A partir de este momento cuando hagamos referencia a Quick BASIC lo mencionaremos sólo como QB.

QB nos ofrece, además, adaptarse a nuestras necesidades de programación o a nuestro nivel de conocimientos mediante dos niveles de trabajo o menús, los cuales podemos optar según se requiera. Dichos niveles son:

- **Easy Menus** (menús fáciles). Maneja las instrucciones más simples para el usuario, cuyo programa no necesita de instrucciones muy complicadas o cuando se está desarrollando su capacidad de programación (principiantes).

- **Full Menus** (menús completos). Proporciona las instrucciones para programadores avanzados. Full Menus ofrece más opciones en cada ventana, y una nueva ventana que se llama Calls.

En este libro explicaremos el contenido de las opciones más usadas de los **Full Menus**.

D.- CARGANDO QuickBASIC.

Para cargar el QB tenemos que estar en el subdirectorio en el que fue instalado, teclear qb y **Enter** (↵).

- Si QB está instalado en el drive A
  A:\> qb (↵)

- Si está en el disco duro:
  C:\QB45> qb (↵)

- Si está en RED:
  G:\> ALUMNOS \ EST XX: >qb (↵)

Nota: **La máquina interpreta el nombre de QB tanto en mayúsculas como en minúsculas o mezcladas.**

Al terminar de cargarlo aparecerá una pantalla semejante a la Figura 1, la cual está compuesta por dos cuadros, el de **HOJA DE CONTROL** y uno sobrepuesto llamado Caja de diálogo y Bienvenida (Wellcome), la cual nos permite, pulsando **Enter** (↵), entrar a una guía de ayuda de QB, es decir que aparecerán indicaciones y/o ayuda para el buen uso de cada opción y de las instrucciones, comandos y órdenes que podemos utilizar (Help).

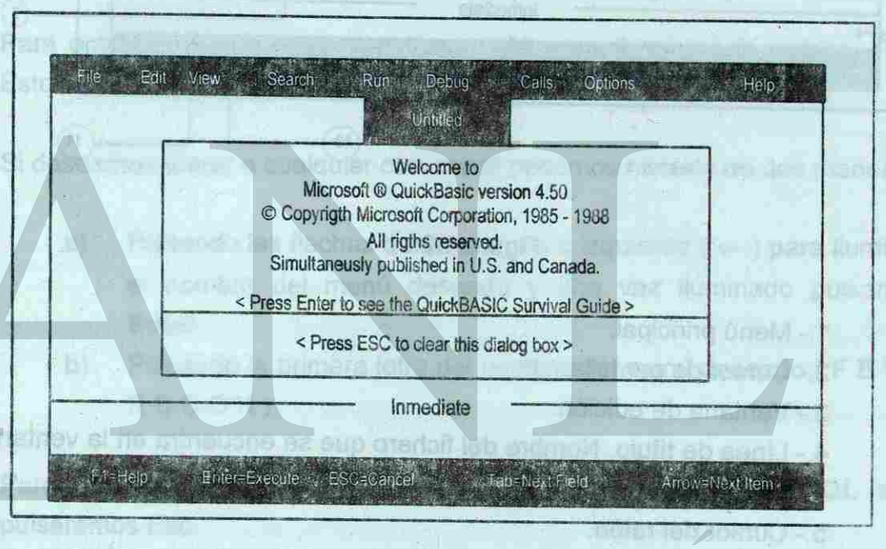


Figura 1.1

Pulsando la tecla ESC borra la caja de Bienvenida y nos deja en la pantalla la **HOJA DE CONTROL** (Figura 1.2), donde podremos accesar las diferentes opciones.

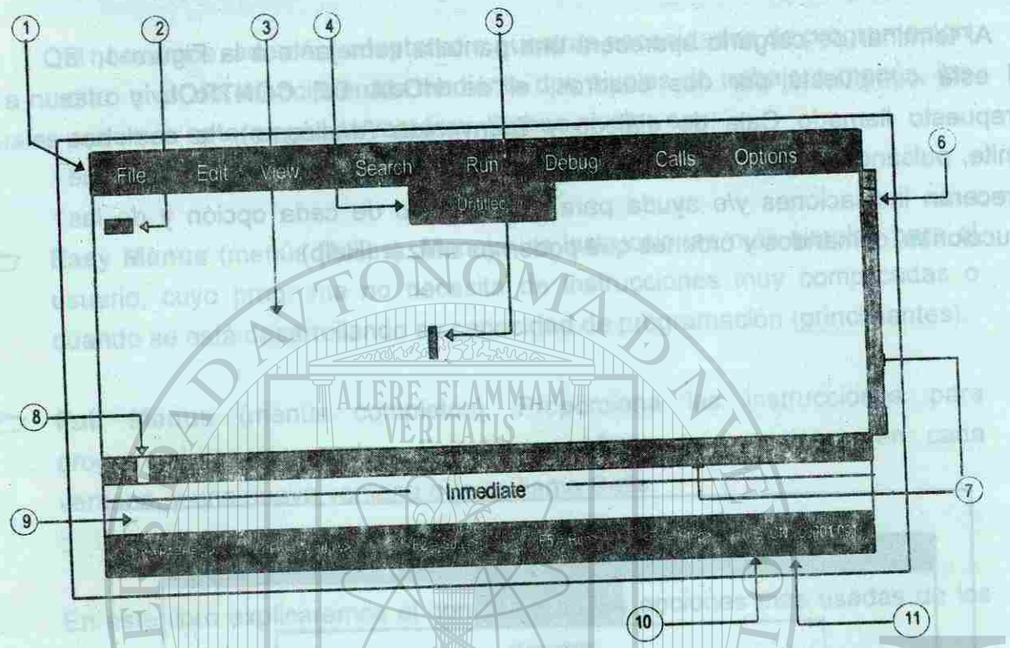


Figura 1.2

1.- Menú principal.
2.- Cursor de pantalla.
3.- Ventana de edición.
4.- Línea de título. Nombre del fichero que se encuentra en la ventana de edición.
5.- Cursor del ratón.
6.- Cursor de scroll vertical. Indica la posición relativa del cursor de pantalla, dentro del texto en edición
7.- Línea de scroll vertical y horizontal. Para utilizarla sólo con el ratón.
8.- Cursor de scroll horizontal.
9.- Ventana en la que se pueden ejecutar sentencias de BASIC directamente.
10.- C Mayúsculas N Teclado numérico
11.- Coordenadas del cursor de pantalla, renglón : columna

E.- MANEJO DE MENÚS.

En la línea superior de la **HOJA DE CONTROL** se encuentra el menú principal, el cual consta de las siguientes opciones: **File, Edit, View, Search, Run, Debug, Calls, Options Help**; éstas contienen una lista de funciones que podemos usar para indicarle a la máquina la instrucción que deseamos realizar. Para estar seguros de que estamos en la **HOJA DE CONTROL** pulsemos Esc. La línea del menú principal estará uniforme, sin estar marcada ninguna palabra.

Ahora veremos el proceso para manejar los menús.

1o.- Para entrar al menú principal pulsemos Alt y se iluminará la palabra **File**. Esto indica que podemos entrar al menú de manejo de archivos: **File**.

2o.- Si deseamos entrar a cualquier otro menú podemos hacerlo de dos maneras:

a) Pulsando las flechas derecha (→) o izquierda (←) para iluminar el nombre del menú deseado y una vez iluminado pulsamos **Enter**.
b) Pulsando la primera letra del nombre del menú deseado (**F E V S R D C O H**).

3o.- Para regresar desde cualquier menú a la **HOJA DE CONTROL** sólo pulsaremos Esc.

En este curso explicaremos las opciones más utilizadas en los menús File, Edit, Run y Options, las restantes opciones se encuentran explicadas en el Anexo, que se encuentra al final del libro.

NOTA: Algunas opciones dentro de los menús nos presentan en la parte derecha de la ventana la forma corta para llamarlos.

Ejem: RUN (SHIFT + F5)

📖 **FILE.-** Cuenta con doce opciones con las que se pueden cargar archivos. Con **Alt+F** llamamos a este menú, el cual contiene:

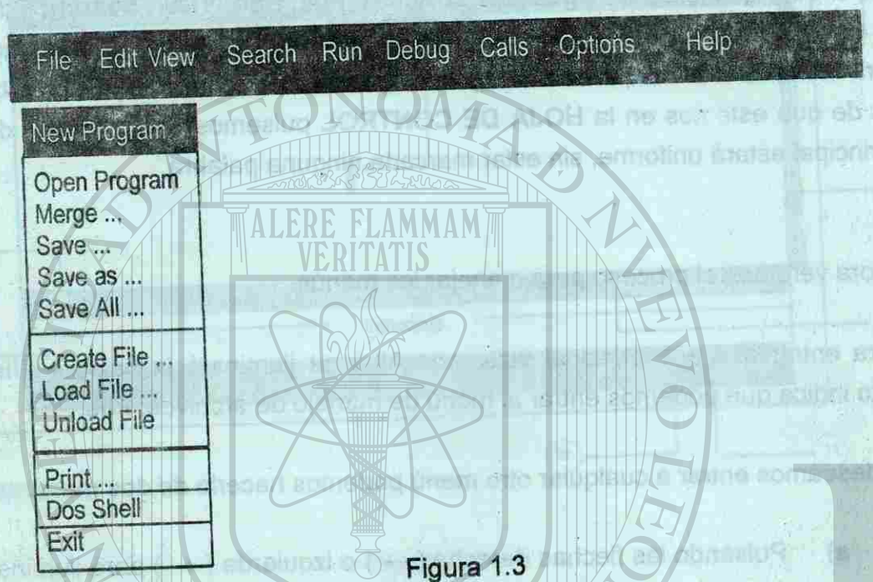


Figura 1.3

New Program.- Borra cualquier programa que se encuentre actualmente en memoria y nos permite crear uno nuevo.

Open Program... Abre en memoria un programa grabado en un disco.

Save. . . Salva o guarda en el disco el archivo que se encuentra en memoria con el nombre dado anteriormente.

Save As... Salva o guarda en disco el archivo que se encuentra en memoria permitiéndonos modificar el nombre, el drive y/o el formato del mismo.

DOS Shell Sale temporalmente de la pantalla de QB y nos permite trabajar en comandos y órdenes del DOS mientras QB está en memoria. Para regresar a la pantalla de QB en que nos salimos, sólo tecleamos la palabra EXIT y **Enter**.

Exit Salida permanente de QuickBASIC al DOS, borrando de memoria el QB. Si en ese momento hay un programa en memoria, al que se le haya hecho una modificación, nos pregunta si queremos salvarlo.

📖 **EDIT.-** Este menú tiene siete opciones, las que se utilizan en la edición de cualquier texto. Para ir directo a este menú: **Alt+E**

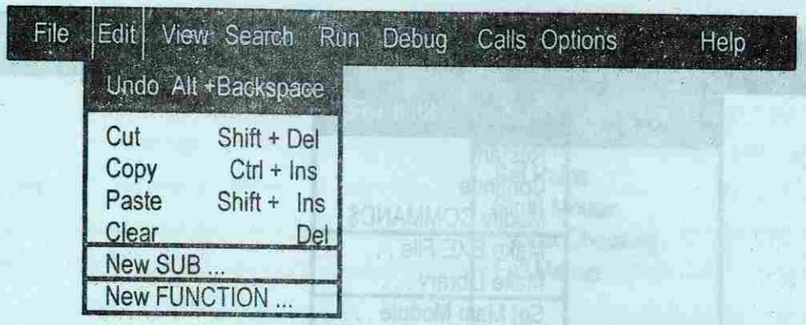


Figura 1.4

Undo Restaura o devuelve el contenido de la línea donde se encuentra el cursor, deshaciendo todos los cambios y correcciones que se hayan hecho en la línea, antes de dar **Enter**.

Cut Recorta un conjunto de caracteres, palabras o líneas seleccionadas previamente y las sitúa en la memoria intermedia.

Copy Copia un conjunto de caracteres, palabras o líneas seleccionadas previamente y sitúa en la memoria intermedia (**Clipboard**), sin alterar el texto original (**Ctrl+Ins**).

Paste Pega o agrega el conjunto de caracteres que está en la memoria intermedia (**Clipboard**), situándolos donde se encuentra el cursor (**Shift+Ins**).

Clear Borra el texto seleccionado, no lo guarda en memoria.(**Del**).

📖 **RUN**.- Este menú contiene siete órdenes. Forma directa **Alt+ R.**

Figura 1.5

Start Ejecuta el programa que está actualmente en memoria (**Shift+F5**).

Restart Se utiliza en el caso de un programa que durante su ejecución es interrumpido y queremos volver a correrlo desde el inicio.

Continue Reanuda la ejecución de un programa en la siguiente instrucción cuando éste se ha interrumpido (**F5**).

Make EXE File... Crea un archivo ejecutable en disco. El archivo tiene que haberse guardado antes con la opción Save del menú Files (**Alt+F+S**).

📖 **OPTIONS**.- Tiene 5 menús. Modo directo (**Alt + O**).

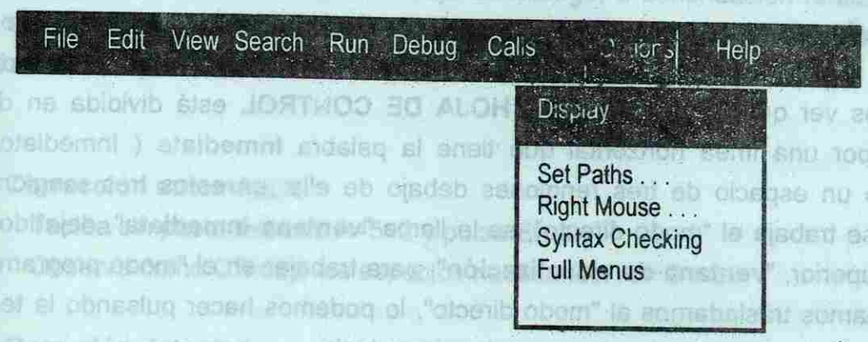


Figura 1.6

Syntax Checking Esta es la función que verifica cada renglón del programa para ver que esté bien escrito; si hay algún error manda un mensaje y el cursor se posiciona en la palabra mal escrita. Esta función se activa o desactiva con la misma orden. El símbolo (•) nos indica que está activada.

Full Menus Activa o desactiva los menús completos (Full Menus); si está marcado con un punto está activo; si no tiene marca es que se está trabajando con los menús simples (Easy Menus).

4.- USO Y MANEJO DE QuickBASIC.

A.- FORMAS DE TRABAJAR EN QB.

De la misma manera como se trabajó en BASIC, modo programa y modo directo, QB también lo permite; si observamos cualquier figura de esta unidad podemos ver que la pantalla de la **HOJA DE CONTROL** está dividida en dos partes por una línea horizontal que tiene la palabra **Inmediate** (Inmediato), dejando un espacio de tres renglones debajo de ella; en estos tres renglones donde se trabaja el "modo directo", se le llama "**ventana inmediata**", dejando la parte superior, "**ventana de visualización**", para trabajar en el "modo programa". Si deseamos trasladarnos al "modo directo", lo podemos hacer pulsando la tecla F6 y para regresarnos lo haremos con la misma tecla.

💻 **MODO PROGRAMA**.- Estando en la pantalla de la **HOJA DE CONTROL** pulsa varias veces la tecla F6 y observa cómo el cursor cambia de ventana; déjalo en la parte superior (**ventana de visualización**).

Teclea:

☞ PRINT "HOLA, BUENOS DIAS" (↵)

La máquina no ejecutó la instrucción que le dimos, para que la ejecute pulsemos la teclas **SHIFT + F5**. Pulsando **Enter** regresaremos a la pantalla de la **HOJA DE CONTROL**.

💻 **MODO DIRECTO**.- Pulsando F6 el cursor pasará a la parte de abajo **(ventana inmediata)**

Teclea:

☞ CLS : PRINT "HOY ES DIA DE TRABAJO EN ESTA PREPARATORIA"

Observa que cuando pulsamos **Enter** inmediatamente borra la pantalla y despliega el mensaje que le dimos. A esto se le llama modo directo y sólo puede ejecutarse en la **ventana inmediata**.

B.- CORRECCIÓN DE SINTAXIS.

Al detectar errores de escritura en la elaboración de un programa original, QB corrige el error o despliega una caja de diálogo, a continuación realizaremos un ejemplo para explicarlo mejor. Realiza los pasos necesarios para entrar a QB, esto nos dará una pantalla de la **HOJA DE CONTROL** limpia.

a.- Corrección automática

☞ Teclea sin pasar espacio A=5+2 y pulsa **Enter**.

☞ Observa como QB dejo los espacios necesarios: A = 5 + 2

b.- Correción detectada

☞ Teclea la siguiente instrucción, conscientes de que está equivocada:

☞ PRONT "YO SOY UNA COMPUTADORA"

☞ Pulsa las teclas **Shift + F5**.

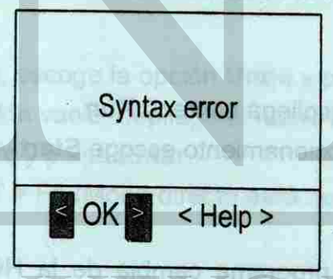


Figura 1.7

Aparecerá la Figura 1.7 en donde aparecerá la palabra PRONT con mayor intensidad de luz; si escogemos **< OK >** regresa al programa y el cursor estará bajo la palabra **PRONT**; si escogemos **< Help >** aparecerá en la pantalla la explicación para las posibles causas del error.

☞ Para corregir el error escoge la opción **<OK>** y corrige la palabra PRONT por PRINT.

☞ Pulsa las teclas **Shift + F5**.

☞ Borra el programa anterior sin salirnos de QB.

C.- MANEJO DE QB

En el siguiente ejemplo utilizaremos los menús explicados en las páginas anteriores; realiza los pasos necesarios para estar en QB, pulsa varias veces la tecla ESC para desactivar lo que tengas en pantalla.

a.- Edición y Ejecución de un programa.

En la **ventana de visualización** teclea el siguiente programa, en QB al final de cada línea se puede pulsar **Enter** o con las **flechas de direccionamiento** te puedes cambiar de renglón para que la línea sea aceptada .

```
CLS
PRINT "PREPARATORIA No.15"
PRINT "EJEMPLO DE QB"
PRINT "UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN"
END
```

- Con las teclas **Alt, R** despliega el menú **Run**.
- Con las flechas de direccionamiento escoge **Start** y pulsa **Enter**.

Cuando se ejecuta el programa cambia de la **HOJA DE CONTROL** a la **HOJA DE EJECUCIÓN**, no se necesitan números de línea para trabajar en QB por lo tanto la secuencia de ejecución del programa va de acuerdo a las lineas de las instrucciones.

Como se observa, los letreros de las instrucciones se encuentran en desorden, utilizaremos el menú Edit para arreglarlo; para regresar a la **HOJA DE CONTROL** pulsa cualquier tecla como lo indica el letrero de la parte inferior **Press any key to continue**. Recuerda que para mover el cursor se utilizan las flechas de direccionamiento.

- Coloca el cursor debajo de la letra "**P**" de la linea PRINT " UNIVER...
- Pulsa la tecla SHIFT y sin soltarla pulsa la flecha de direccionamiento hacia la derecha, hasta llevar el cursor al final del renglón (al ponerse en video inverso significa que se encuentra seleccionado).
- Despliega el Menú **Edit**, escoge la opción de **Cut** y pulsa **Enter**.

Si observastes la línea desapareció al pulsar **Enter**, esto sucede porque el texto seleccionado se guardo en la memoria intermedia (**Clipboard**) de la computadora listo para pegarse en otra parte del programa.

- Coloca el cursor debajo de la letra "**P**" de la línea PRINT " PREPA...
- Depliega el menú **Edit**, escoge la opción **Paste** y pulsa **Enter**.

En el mismo renglón aparece aparecen las dos líneas, vamos a utilizar otra opción del menú **Edit** para arreglarlo.

- Depliega el menú **Edit**, escoge la opción **Undo** y pulsa **Enter**.
- Sube el cursor al renglón vacio, depliega el menú **Edit**.
- Escoge la opción **Paste** y pulsa **Enter**.
- Pulsa las teclas SHIFT + F5 (Modo directo para correr un programa)

Al pulsar **Enter** cuando el cursor se encuentra en la parte inical de una línea se abre un renglón o si se pulsa la tecla BACKSPACE se quita un renglón vacio.

- Selecciona el renglón PRINT " UNIVER.... ,
- Despliega el menú **Edit**, escoge la opción **Copy** y pulsa **Enter**.
- Coloca el cursor en el siguiente renglón de la instrucción END.
- Despliega el menú **Edit**, escoge la opción **Paste** y pulsa **Enter**.

Como se observa se manda copiar la línea en la parte final del programa, para borralo sigue las siguientes instrucciones:

- Selecciona la última línea, despliega el menú **Edit**.
- Escoge la opción **Clear** y pulsa **Enter**.

b.- Guardar un programa

Para grabar el ejemplo anterior utilizaremos el menú **File**, sigue los siguientes pasos:

- Despliega el menú **File**, escoge la opción **Save as ...** y pulsa **Enter**.
- Aparecerá en la pantalla de tu monitor la siguiente caja de diálogos:

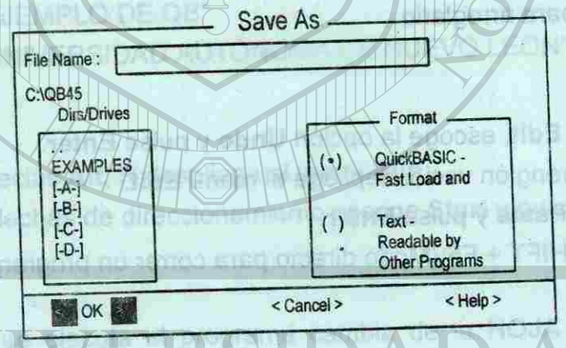


Figura 1.8

Para mover el cursor dentro de cualquier caja de diálogos se utiliza la tecla **TAB**. En el recuadro de **Dirs/drive** se selecciona el Drive donde se encuentra el disco de trabajo. En **File Name** se escribe el nombre del programa, máximo ocho caracteres sin pasar espacios en blanco, podemos teclear la extensión .BAS al programa o si se nos olvida QB lo pone de manera automática. En **Format** se puede grabar el programa realizado como **QuickBASIC...** o **Text..**, éste segundo sirve para grabarlo en ASCII o sea como texto.

Primero direccionaremos el Drive donde se encuentra nuestro disco de trabajo.

- Con la tecla TAB posiciona el cursor en el recuadro de **Dirs/Drive**.
- Con las flechas de direccionamiento selecciona el Drive y pulsa **Enter**.
- Observa como se cambia la segunda línea de la caja de diálogos.
- En la linea **File Name** escribe **Ejemplo1** y pulsa **Enter**.
- Observa en la **HOJA DE CONTROL** se pone el nombre del programa.
- Despliega el menú **File**, escoge la opción **New Program** y pulsa **Enter**.

Para terminar cualquier sesión de trabajo en QB, despliega el menú File, escoge la opción **Exit** y pulsa **Enter**.

Si realizastes algún cambio en el programa desde la última vez que lo grabastes y quieres salir QB, al momento de salir se desplegará un letrero que te indicará si quieres grabar los cambios realizados, con la tecla **TAB** escoge la opción más adecuada a tus cambios.

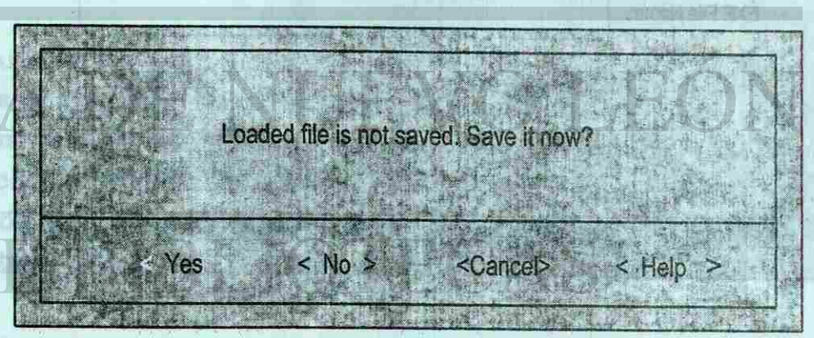


Figura 1.9

D.- PROGRAMA EJECUTABLE .

Una vez guardado el programa podemos transformarlo en otro programa que tenga .EXE como extensión, para que sea ejecutable desde DOS, sin que esté presente el QB o el GW-BASIC o cualquier otro de los conocidos.

Realizaremos un ejemplo para explicarlo, esto lo haremos usando el menú **Run** y la opción **Make EXE File...** ,.

- Realiza los pasos necearios para entrar a QB.
- Despliega el menú **File**, escoge la opción **Open Program ...** y pulsa **Enter**.
- Con la tecla **TAB** direcciona el drive donde está tu disco de trabajo.
- Selecciona el programa **Ejemplo1.BAS** y pulsa **Enter**.
- Pulsa las teclas **SHIFT + F5** y observa la ejecución del programa.
- Pulsa cualquier tecla
- Pulsa las teclas **Alt,R** para desplegar el menú **Run**.
- Escoge la opción **Make EXE File ...** y pulsa **Enter**.

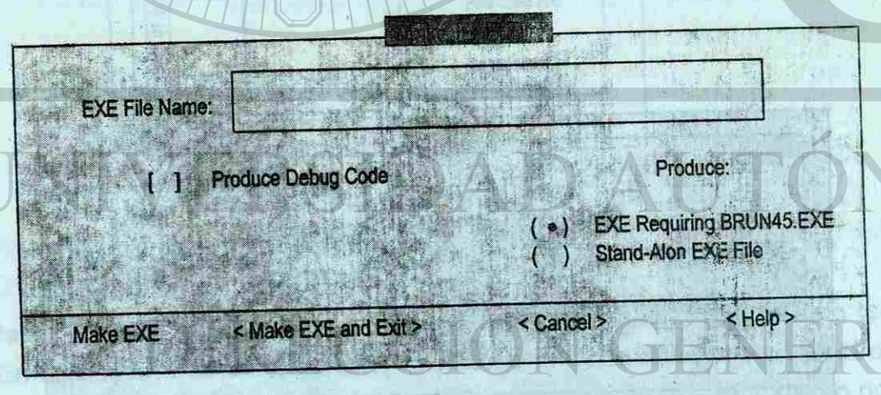


Figura 1.10

a) Con BRUN 45

En forma automática se encuentra seleccionada la opción **EXE Requering BRUN45.EXE**, si pulsamos **Enter** se crearía un programa que se puede ejecutar en cualquier máquina compatible, sin necesidad de que QB, BASICA o GW-BASIC esté en memoria, sólo necesitamos que el archivo **BRUN45.EXE** se encuentre en el disco.

- En el recuadro de **EXE File name** escribe A: o B:EJEMPLO1 (según donde se encuentre tu disco de trabajo) y pulsa **Enter**.

b) Con Stand - Alone

Con la tecla **TAB** y las flechas de direccionamiento podemos seleccionar en **Make EXE File** la opción de **Stand - Alone EXE File**, con esto se crearía un programa ejecutable sin la necesidad de ningún archivo.

- Pulsa las teclas **Alt,R** para desplegar el menú **Run**.
- Escoge la opción **Make EXE File ...** y pulsa **Enter**.
- En el recuadro de **EXE File name** escribe A: o B:EJEMPLO2 (según donde se encuentre tu disco de trabajo).
- Con la tecla TAB y las flechas de direccionamiento escoge la opción **Stand - Alone EXE File** y pulsa **Enter**.

Para comprobar lo anterior utilizaremos el **DOS Shell** de QB para salir en forma momentánea al sistema operativo, recuerda que para regresar sólo hay que escribir **Exit** y pulsar **Enter**.

- Despliega el menú **File** , escoge la opción **DOS Shell** y pulsa **Enter**.
- En el Prompt **A:\>** o **B:\>** escribe EJEMPLO1.EXE y pulsa **Enter**.
- Observa como se ejecuta el programa.
- En el Prompt **A:\>** o **B:\>** escribe EJEMPLO2.EXE y pulsa **Enter**.
- Observa como se ejecuta el programa.
- En el Prompt **A:\>** o **B:\>** escribe DIR y pulsa **Enter.**
- Observa las caracteristicas de EJEMPLO1.BAS, EJEMPLO1.EXE y EJEMPLO2.EXE
- Teclea **Exit** y al regresar a QB, realiza los pasos necesarios para salir de QB en forma definitiva.

E.- CONVERSIÓN DE BASIC A QB.

En párrafos anteriores tratamos el tema sobre las ventajas de QB y se dijo que los programas guardados en BASIC se pueden ejecutar en QB si se guardaron en código ACII, ahora incluiremos la manera de convertir un programa de BASIC a QB.

En el curso anterior se elaboraron varios programas, de los cuales tomaremos el ejemplo de la página 112 de tu libro de COMPUTACIÓN I , el cual dice programa que suma dos números cualesquiera, los multiplica e imprime sus resultados.

Si no tienes este programa guardado lo puedes teclear nuevamente en BASIC o también puedes optar por otro programa. Antes de convertirlo a QB veamos como fue guardado por BASIC.

En el sistema operativo DOS, existe una instrucción que permite leer del disco el contenido de un programa, archivo, etc. tal y como fue guardado, esta instrucción es **TYPE**; si el nombre del archivo a trabajar se llama **SUMPOR.BAS**, veamos como funciona esta instrucción:

- Enciende tu computadora y espera a que aparezca el prompt.
- Introduce tu disco de trabajo, donde está el programa, al drive.
- Teclea **TYPE B:SUMPOR.BAS** y pulsa **Enter**.
- ¿ Que sucedió ?

_____

_____

La instrucción **TYPE** muestra en pantalla algunos caracteres no entendibles para nosotros, junto con algunos sonidos y sólo los letreros que tenemos entre comillas en el programa aparecerán de un modo que sí se pueda leer.

Realiza los siguientes pasos para convertir de BASIC el programa SUMPOR.BAS a QB.

- Realiza los pasos necearios para entrar a BASIC.
- Escribe **LOAD"B:SUMPOR.BAS"** y pulsa **Enter**.
- Escribe **SAVE"B:SUMPOR.BAS",A** y pulsa **Enter**.
- Escribe **SYSTEM** y pulsa **Enter.**

Ahora observaremos en el sistema operativo como fue guardado el programa SUMPOR.BAS en el disco.

- Teclea **TYPE B:SUMPOR.BAS** y pulsa **Enter**.
- ¿ Que sucedió ?

_____

_____

De esta manera al guardar un archivo en BASIC con **SAVE"SUMPOR",A** le indicamos que se grabe en modo texto o ASCII; con la instrucción TYPE podemos leer, en el sistema oparativo, el programa con todos los números de línea, las instrucciones y los letreros; también podemos correrlo en QB.

Ahora lo veremos el programa en QB, lleva a cabo los siguientes pasos

- Realiza los pasos necesarios para entrar a QB.
- Pulsa las teclas Alt,F para desplegar el menú **File**.
- Escoge la opción **Open program ...** y pulsa **Enter**.
- Con la tecla TAB y las flechas de direccionamiento selecciona el drive de trabajo y el programa SUMPOR.BAS y pulsa **Enter**.
- Con las teclas **Alt, R** despliega el menú **Run**.
- Con las flechas de direccionamiento escoge **Start** y pulsa **Enter**.
- ¿ Que sucedió ?

_____

_____

Con los siguientes pasos veremos que sin números de líneas podemos correr el programa en QB.

- Pulsa cualquier tecla (para regresar a la pantalla de programación).
- Quitale los números de línea al programa.
- Con las teclas **Alt, R** despliega el menú **Run**.
- Con las flechas de direccionamiento escoge **Start** y pulsa **Enter**.
- ¿ Que sucedió ?

_____

_____

- Realiza los pasos necesarios para guardar el programa con el nombre **CONVER.BAS** en tu disco de trabajo.

☺ **EJERCICIO GENERAL:**

1.- Entra a BASIC y carga en memoria un programa que hayas realizado en el curso anterior.

2.- Guardalo en tu disco de trabajo en modo texto o código ASCII y sal de BASIC.

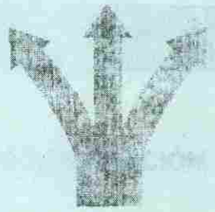
3.- En el sistema operativo utiliza la instrucción TYPE para observarlo.

4.- Entra a QB, carga el programa, quitale los números de línea y corre el programa.

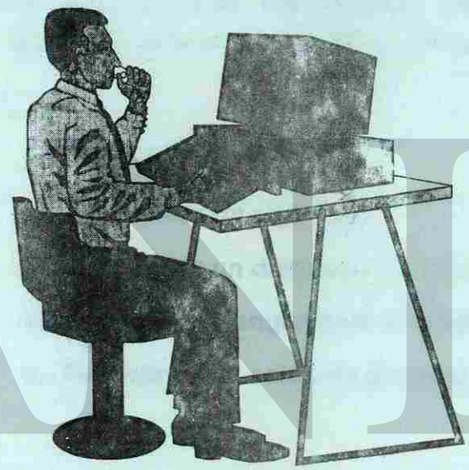
5.- Guarda el archivo con el nombre EJEGEN y sal de QB.

CONDICIONAL

SECUENCIAL



CÍCLICA



PROGRAMACIÓN ®
EN
QuickBASIC

UNIDAD II

PROGRAMACIÓN EN QuickBASIC

1.- INTRODUCCIÓN.

A.- METODOLOGÍA DE LA PROGRAMACIÓN

En el libro del curso anterior, explicamos que para resolver problemas o tareas por medio de la computadora, era necesario seguir una metodología para la programación, la cual consiste en los siguientes pasos:

a.- Analizar el problema.

b.- Desarrollar un algoritmo.

c.- Desarrollar un diagrama de flujo.

d.- Codificar en lenguaje de programación.

e.- Depurar (corrección de errores).

Mencionamos, además, que estos pasos hay que realizarlos independientemente del lenguaje que estemos utilizando: BASIC, PASCAL, FORTRAN,COBOL, etc.

En ésta unidad, nos dedicaremos a trabajar la programación en QB de manera semejante al libro anterior, en donde estudiamos un conjunto de instrucciones cuyo formato es igual para el QB. Nuestro aprendizaje se dividió en algunas formas de programación como son las secuenciales, condicionales y cíclicas o bucles.

2.- TIPOS DE PROGRAMACIÓN EN QB.

A.- PROGRAMACIÓN SECUENCIAL.

En el curso anterior de computación se utilizaron en la programació secuencial algunas instrucciones elementales como:

- **CLS**.- Limpia pantalla.
- **LET**.- Indica una operación arimética o asigna un valor determinado a u variable.
- **PRINT**.- Imprime el contenido de las variables y/o expresiones en el monito
- **LPRINT**.- Imprime el contenido de las variables y/o expresiones en impresora.
- **INPUT**.- Acepta datos de entrada por el teclado.
- **END**.- Termina la ejecución de un programa.

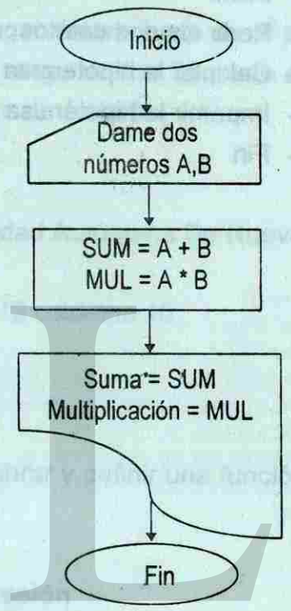
a.- Ejemplos

Ejemplo No.1.- Elabora el **algoritmo**, diagrama de flujo y la **codificación** en QB de un programa que, **con dos números cualesquiera, calcule la suma y la multiplicación e imprima sus resultados**.

ALGORITMO

1.- Inicio
2.- Pedir dos números
3.- Sumar los números
4.- Multiplicar los números
5.- Imprimir resultados
6.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

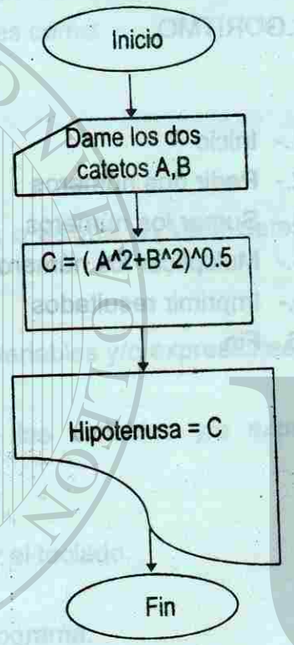
```
CLS
INPUT "Dame el valor del primer número: ", A
INPUT "Dame el valor del segundo número: ", B
SUM = A + B
MUL = A * B
PRINT " La suma es: "; SUM
PRINT " La multiplicación es:" ; MUL
END
```

Ejemplo No.2.- Elabora el algoritmo, diagrama de flujo y la codificación en QB de un programa que obtenga la hipotenusa de un triángulo a partir de sus catetos: $C^2 = A^2 + B^2$

ALGORITMO

1.- Inicio
2.- Pedir los dos catetos
3.- Calcular la hipotenusa
5.- Imprimir la hipotenusa
6.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
INPUT "Dame el valor del primer cateto: ", A
INPUT "Dame el valor del segundo cateto: ", B
C = (A^2 + B^2)^0.5
PRINT " El valor de la hipotenusa es: "; C
END
```


b.- Nuevas Instrucciones

Ahora en el estudio de este módulo, incrementaremos algunas instrucciones, con las cuales, después de ver su definición, formato y ejemplo, realizaremos algunos programas para explicarlas mejor.

- **LOCATE**.- Permite posicionar el cursor en lugar determinado de la pantalla, sobre un renglón, el cual debe de estar en el rango de 1 al 24 y una columna especificada, la cual debe de estar en el rango de 1 al 80.

  Formato: LOCATE r , c
  Ejemplo: LOCATE 12 , 10 : PRINT "Universidad Autónoma De Nuevo León"

  En el ejemplo el letrero iniciará en el renglón 12 columna 10.

- **DEF FN**.- Con esta instrucción podemos nombrar y definir una función o una fórmula y así obtener el resultado de ella.

  Formato: DEF FN nombre (variable) = expresión
  Ejemplo: DEF FNy(x) = x + 5

  El ejemplo indica que y se encuentra en funcion de **x** y su valor es **x** más 5.

- **SPC**.- Instrucción que sirve para especificar **n** cantidad de espacios en blanco.

  Formato: SPC (n)
  Ejemplo: PRINT SPC (50)

  La instrucción del ejemplo indica que imprimirá 50 espacios en blanco.

SQR.- Es una función numérica que proporciona la raíz cuadrada de una expresión aritmética y es equivalente a obtener la potencia 0.5 de la base.

Formato: SQR(expresión numérica)
Ejemplo: B = SQR (A)

El valor de B en el ejemplo, será igual a la raiz cuadrada del valor de la variable A.

TECLEA:

```
CLS
DEF FNy(x) = x^ 3+ 7
LOCATE 4, 10 : INPUT " Dame el valor de x:  ", x
A = SQR ( x ) : B = SQR (FN y(x) )
LOCATE 4, 10 : PRINT SPC ( 50 ) ;
LOCATE 10, 20 : PRINT " El valor de y es:  ", FNy(x)
LOCATE 12, 20 : PRINT " La raiz cuadrada de x es:  " ; A
LOCATE 14, 20 : PRINT " La raíz cuadrada de y es: " ; B
END
```

- Corre el programa (SHIFT + F5).
- Dale un valor de 10
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

Desde la tercera hasta la séptima línea del programa, los dos puntos sirven para realizar dos operaciones y/o instrucciones en la misma línea.

FUNCIONES TRIGONOMETRICAS.- Los ángulos que se utilizan en las funciones trigonométricas que se trabaja tanto en BASIC como en QB, deben de estar en radianes, por lo tanto si se realiza un programa y los datos se proporcionan en grados, se tendrá que hacer la conversión de grados a radianes con la siguiente fórmula: R = G (3.1416)/180

Formato: SIN(X)	**COS(X)**	**TAN(X)**
Ejemplos: SIN (20)	COS (B)	TAN (N)

En los ejemplos de las fuciones trigonométricas, podemos darle el ángulo en forma directa o en función de una variable.

TECLEA:

```
CLS
DEF FNCOT (R) = 1 / TAN(R)
DEF FNSEC (R) = 1 / COS(R)
DEF FNCSC (R) = 1 / SIN(R)
LOCATE 6, 20: INPUT "Dame el valor del Ángulo en grados: ", G
R = G * 3.1416 / 180
LOCATE 10, 20: PRINT "El valor del Seno es: "; SIN(R)
LOCATE 12, 20: PRINT "El valor del Coseno es: "; COS(R)
LOCATE 14, 20: PRINT "El valor de la Tangente es: "; TAN(R)
LOCATE 16, 20: PRINT "El valor de la Cotangente es: "; FNCOT(R)
LOCATE 18, 20: PRINT "El valor de la Secante es: "; FNSEC(R)
LOCATE 20, 20: PRINT "El valor de la Cosecante es: "; FNCSC(R)
END
```

- Corre el programa (SHIFT + F5).
- Dale un valor de 30
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

c.- Ejercicios

Desarrolla el algoritmo, el diagrama de flujo y la codificación QB para los siguientes programas.

1.- Realiza un programa que dé la solución a un sistema de ecuaciones por el método de determinantes.

Ecuación 1: $a_1x + b_1y = c_1$

Ecuación 2: $a_2x + b_2y = c_2$

NOTA: Para resolver este problema consulta tu libro de matemáticas

2.- Elabora programa que determine la raíz cuadrada de la suma de tres números dados.

3.- Realiza el programa que encuentre el valor de la pendiente de la recta que pasa a través de dos puntos dados P1, P2, cuyas coordenadas son: (X_1 , Y_1) y (X_2 , Y_2).

$m = (Y_2 - Y_1) / (X_2 - X_1)$

B.- PROGRAMACIÓN CONDICIONAL.

La instrucción IF-THEN-ELSE, que se utilizó en BASIC para establecer la selección de una de las dos opciones en el proceso de un programa, se puede usar en QB, pero si en el programa se presentan más de dos opciones se puede utilizar un nueva instrucción que sólo trabaja en QB: SELECT CASE.

- **SELECT CASE**.- Permite establecer dos o más opciones en un mismo programa.

Formato: SELECT CASE expresión
CASE condición 1
CASE condición 2
CASE . . . . .
CASE . . . . .
CASE ELSE (opcional)
END SELECT

Ejemplos de formatos que se utilizan en las condiciones:

CASE IS < x	cuando la condición es menor que x
CASE IS =15	cuando la condición es igual a 15
CASE 10 to 100	cuando la condición está en el rango de 10 a 100
CASE IS > =100	cuando la condición es mayor o igual que 100
CASE " A "	cuando la condición es igual a la constante A
CASE 7	cuando la condición es igual a 7
CASE 3,9	cuando la condición es igual a 3 ó 9

Al igual que el IF-THEN-ELSE, el SELECT CASE acepta los mismos operadores relacionales y lógicos: **=, > ,< , >= , <= , AND, OR, NOT**

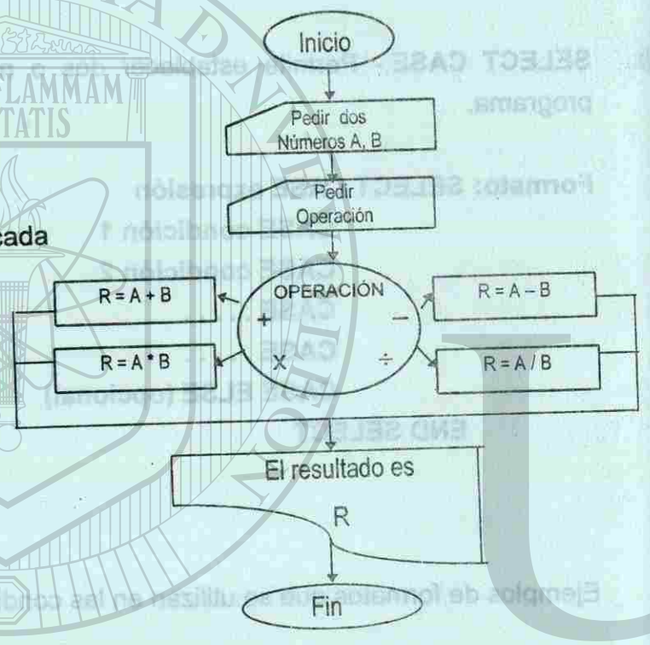
a.- Ejemplos

Ejemplo No.1.- Realiza el algoritmo, diagrama de flujo, codificación en QB de un programa que, con dos números cualesquiera, pueda efectuar una de las operaciones fundamentales de la aritmética.

ALGORITMO

1.- Inicio
2.- Pedir dos números
3.- Pedir la operación
4.- Seleccionar opción
5.- Efectuar operación indicada
6.- Imprimir resultado
7.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

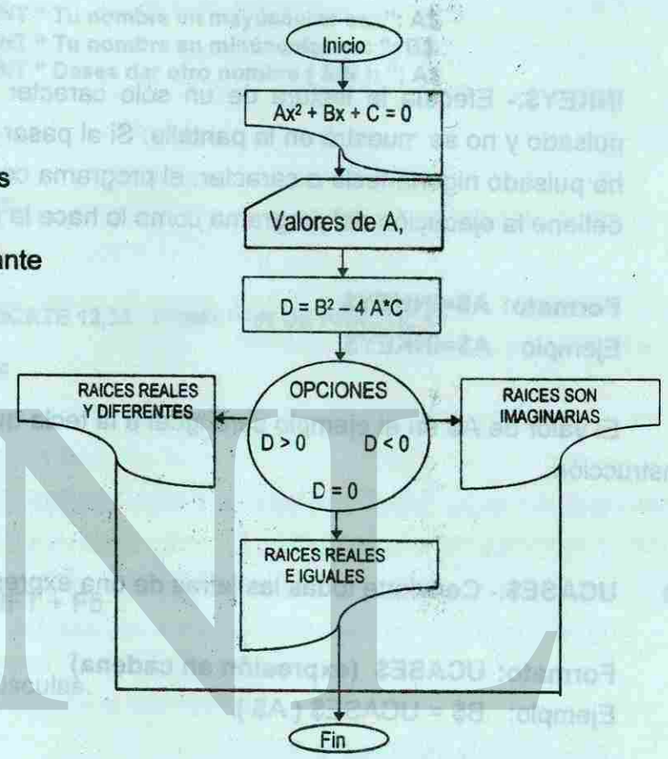
```
CLS
INPUT "Dame el valor del primer número: ",A
INPUT "Dame el valor del segundo número (Diferente a cero): ",B
INPUT "Operación a ejecutar: +,-,*,/: ",OP$

SELECT CASE OP$
      CASE "+"
            R=A+B
      CASE "-"
            R=A-B
      CASE "*"
            R=A*B
      CASE "/"
            R=A/B
END SELECT

PRINT "La operación tiene como respuesta: ";R
END
```


Ejemplo No. 2.- Realiza el algoritmo, diagrama de flujo y codificación QB de un programa que determine si la ecuación de 2do. grado $Ax^2+Bx+C=0$ tiene raíces: reales y diferentes, reales e iguales o imaginarias.

ALGORITMO

1 Inicio
2.- Imprime ecuación cuadrática general
3.- Pregunta los valores de los coeficientes
4.- Evalúa el discriminante
5.- Selecciona opción
6.- Imprime resultado
7.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
LOCATE 3,10 : PRINT"La Ecuación a trabajar es Ax^2+Bx+C=0"
LOCATE 5,10 : INPUT"Dame los valores de A,B,C: ",A,B,C
D=B^2-4*A*C

SELECT CASE D
     CASE IS = 0
          LOCATE 5,15 : PRINT"Las raíces son reales e iguales"
     CASE IS > 0
          LOCATE 5,15 : PRINT"Las raíces son reales y diferentes"
     CASE IS < 0
          LOCATE 5,15 : PRINT"Las raíces son imaginarias"
END SELECT

END
```

b.- Nuevas Instrucciones

A continuación veremos nuevas instrucciones que se pueden aplicar en QB, después de ver su definición, formato y ejemplo, realizaremos un programa para explicarlas mejor.

- **INKEY$.-** Efectúa la lectura de un sólo caracter del teclado que **se** haya pulsado y no se muestra en la pantalla. Si al pasar por ésta instrucción no se ha pulsado niguna tecla o caracter, el programa continúa; ésta instrucción no detiene la ejecución del programa como lo hace la instrucción INPUT.

  Formato: A$=INKEY$
  Ejemplo : A$=INKEY$

El valor de A$ en el ejemplo será igual a la tecla que se pulse al llegar a esta instrucción.

- **UCASE$.-** Convierte todas las letras de una expresión en mayúsculas.

  Formato: UCASE$ (expresión en cadena)
  Ejemplo: B$ = UCASE$ (A$)

En el ejemplo el contenido de B$ será guardado con letras mayúsculas, independientemente si A$ se encuentre con mayúsculas o minúsculas.

- **LCASE$.-** Convierte todas las letras de una expresión en minúsculas.

  Formato: LCASE$ (expresión en cadena)
  Ejemplo: B$ = LCASE$ (B$)

En el ejemplo el contenido de B$ será guardado con letras minúsculas, independientemente si B$ se encuentre con mayúsculas o minúsculas.

TECLEA:

```
20    CLS
      LOCATE 4,10: INPUT " Dame tu nombre completo:  ", A$
      A$ = UCASE$ ( A$ )
      B$ = LCASE$ ( A$ )
      LOCATE 10,15 : PRINT " Tu nombre en mayúsculas es: "; A$
      LOCATE 12,15 : PRINT " Tu nombre en minúsculas es: "; B$
      LOCATE 20,28 : PRINT " Desea dar otro nombre ( S/N ): "; A$
25    C$ = INKEY$

      SELECT CASE C$
            CASE ""
                  GOTO 25
            CASE "S","s"
                  GOTO 20
            CASE "N"
                  CLS : LOCATE 12,35 : PRINT "FIN DE PRÁCTICA"
            CASE ELSE
                  GOTO 25

      END SELECT
      END
```

- Corre el programa (SHIFT + F5).
- Dale tu nombre en minúsculas.
- ¿Que sucedió? _____
- Daie tu nombre en mayúsculas.
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

NOTA: Como se observa el programa no está validado o sea que puede tener errores en la captura de información.

Ahora veremos otras instrucciones que aplicaremos en el programa que se encuentra despúes de la explicación.

- **SOUND**.- Da un sonido en una frecuencia desde 37 hasta 32767 Hertz y una duración en la que va a permanecer el sonido durante el tiempo que nosotros indiquemos, cada unidad de duración equivale a 1/18 de segundo.

  Formato: SOUND frecuencia, duración
  Ejemplos: SOUND 1000, 9

  El sonido que producirá la instrucción en el ejemplo tendrá una frecuencia de 1000 Hertz y una duración de medio segundo.

- **RND.-** Sirve para semejar números aleatorios de 0 a 0.999999, con RANDOMIZE TIMER la función RND obtendrá un valor de semilla diferente cada vez que se ejecute, pero dependiendo de la escala de números que se quiera trabajar, se hacen los arreglos necesarios.

Formato: RANDOMIZE TIMER
RND

Ejemplo:	RND	genera un número de 0 a 0.999999
	100*RND	genera un número de 0 a 99.9999
	1000*RND	genera un número de 0 a 999.999

En los ejemplos anteriores es importante recalcar el uso del RANDOMIZE TIMER, porque si éste no se coloca al antes del RND, el valor será el mismo cuantas veces se genere.

- **INT**.- Sirve para eliminar la parte decimal del número, dejando sólo la parte entera.

Formato: INT(expresión numérica)

Ejemplos:	INT(75.32)	obtiene el número 75
	INT(0.54)	obtiene el 0
	INT(1000*RND)	obtiene un número de 0 a 999
	INT (100*RND)+1	obtiene un número de 1 a 100

En el último ejemplo se puede observar que se tuvo que hacer un ajuste aritmetico para obtener un rango apropiado de trabajo.

- **GOSUB**.- Se utiliza en programas cuando se tienen operaciones repetitivas; se puede crear una subrutina en forma interna o externa del programa principal. Al final de la subrutina se concluye con RETURN para que regrese a la línea del programa después del GOSUB; cuando una subrutina es grabada como externa es necesario que sea en modo texto.

Formato: GOSUB número de subrutina
GOSUB nombre de la subrutina

Ejemplo :	GOSUB 5000
	GOSUB SONIDO

Como se observa en los dos ejemplos el nombre de la subrutina puede ser un número, por lo regular es una cantidad grande para no interferir con algunas etiquetas de un programa, o un nombre que vaya de acuerdo a la operación de la subrutina.

TECLEA:

```
20    CLS
      RANDOMIZE TIMER
      LOCATE 4,10: INPUT " Dame tu nombre:  ", A$
      A$ = UCASE$ ( A$ )
      A = INT ( 10 * RND ) + 1
      LOCATE 6, 10: INPUT " Adivina un número del 1 al 10:  ", B

      SELECT CASE  A
           CASE B
                GOSUB 50
                LOCATE 12,28 : PRINT " Felicidades lo adivinaste "
           CASE ELSE
                SOUND 55,36
                LOCATE 12,28 : PRINT " Lo siento no es el número "
      END SELECT

      LOCATE 14,28 : PRINT " El número que pensé es: " ; A
      LOCATE 16,28 : PRINT " El número  de  " ; A$ ; " es: " ; B
25    LOCATE 22,25 : PRINT SPC (50);
      LOCATE 22,25 : PRINT INPUT " Quieres probar otra vez (S/N): " , B$

      SELECT CASE UCASE$(B$)
           CASE "S"
                GOTO 20
           CASE "N"
                CLS: LOCATE 12,35 : PRINT " FIN DE JUEGO "
           CASE ELSE
                SOUND 55,36 : GOTO 25
      END SELECT

      END

50    SOUND 300,3 : SOUND 600,2 : SOUND 900,1
      SOUND 300,1 : SOUND 600,2 : SOUND 900,3
      SOUND 300,3 : SOUND 600,2 : SOUND 900,1
```

☞ Corre el programa (SHIFT + F5).

✎ ¿Que sucedió? _____

☞ Guarda el programa en tu disco de trabajo.

NOTA: Como se observa el programa está validado o sea que puede detectar errores en la captura de información.

c.- Ejercicios

Desarrolla el algoritmo, el diagrama de flujo y la codificación QB para los siguientes programas.

1.- Realiza un programa que determine los valores de las raíces de una ecuación cuadrática $Ax^2+Bx+C=0$ y diga la naturaleza de las raíces.

2.- Realiza un programa que cree un menú de materias para introducir una calificación en una sola opción e imprima la materia con su calificación.

MATERIAS: 1.- Matemáticas
2.- Computación
3.- Español
4.- Educación Física

3.- Realiza un programa que, a partir de un listado de 10 alumnos, escoja uno al azar y se le asigne como APROBADO e imprima el nombre.

1.- Aracely
2.- Felipe
3.- Raúl
4.- Ricardo
5.- Sergio
6.- Sigifredo
7.- Tomás
8.- Ulises
9.- Verónica
10.- Yazmín

C.- PROGRAMACIÓN CÍCLICA.

El concepto de bucle o ciclo es la repetición de instrucciones para lograr realizar una tarea especifica; en programación BASIC se utilizó FOR - NEXT para estudiarlo; ahora en QB usaremos DO / UNTIL, DO / WHILE y el DO - LOOP. En los siguientes ejemplos de las tres instrucciones, realizaremos el mismo programa, imprimir la serie del 1 al 10, con el fin de ver las diferentes formas en que funcionan los ciclos en QB.

- **DO / LOOP**.- Esta instrucción se le considera como bucle infinito, para salir de él es necesario que en una parte del ciclo se utilice EXIT DO.

Formato:

```
DO
--------
EXIT DO
--------
LOOP
```

Ejemplo:

```
CLS
K = 0
DO
    K = K + 1
    PRINT K
    IF K =10 THEN EXIT DO
LOOP
END
```

En el ejemplo el programa entrará en forma directa al DO, como no hay condición de entrada o salida, el proceso del programa estará en el ciclo en forma infinita, pero con la ayuda de las instrucciónes IF y EXIT DO saldrá del bucle.

- **DO / UNTIL**.- El bucle se ejecuta mientras la condición sea falsa, es decir, HASTA que la condición sea verdadera. La condición puede ir al principio o al final del bucle. Si la condición va al principio se utiliza la opción DO UNTIL / LOOP y si la condición va al final se convierte en DO / LOOP UNTIL.

Formato:

```
DO UNTIL (Condición)          DO
---------------------         ---------------------
---------------------         ---------------------
---------------------         ---------------------
---------------------         ---------------------
LOOP                          LOOP UNTIL (Condición)
```

Ejemplo 1:

```
CLS
K = 0
DO  UNTIL K = 10
      K = K + 1
      PRINT K
LOOP
END
```

Ejemplo 2:

```
CLS
H = 0
DO
      H = H + 1
      PRINT H
LOOP UNTIL H = 10
END
```

En el ejemplo 1 la condición va al inicio, por lo tanto el proceso del programa entrará al ciclo porque al iniciar el programa K = 0, **hasta** que K = 10 no entrará al DO y dará un salto a la instrucción END.

En el ejemplo 2 la condición va al final, por lo tanto el proceso del programa entrará al ciclo en forma directa y saldrá del DO **hasta** que H = 10.

DO / WHILE.- El bucle se ejecuta MIENTRAS la condición sea vedadera, es decir, hasta que la condición sea falsa. Permite tener la condición al principio o al final del bucle, cuando la condición va al principio del bucle, la instrucción es DO WHILE / LOOP y si la condición va al final se convierte en DO / LOOP WHILE.

Formatos:

DO WHILE (Condición)	**DO**
----------	----------
----------	----------
----------	----------
----------	----------
LOOP	**LOOP WHILE (Condición)**

Ejemplo 1:

```
CLS
K = 0

DO  WHILE K < 10
      K = K + 1
      PRINT K
LOOP

END
```

Ejemplo 2:

```
CLS
H = 0

DO
      H = H + 1
      PRINT H
LOOP WHILE H < 10

END
```

En el ejemplo 1 la condición va al inicio, por lo tanto el proceso del programa entrará al DO porque al iniciar el programa K = 0, **mientras** K < 10 estará en el bucle y cuando K no cumpla con la condición dará un salto a la instrucción END.

En el ejemplo 2 la condición va al final, por lo tanto el proceso del programa entrará al ciclo en forma directa, **mientras** que H < 10 permanecerá en el bucle, cuando H no cumpla con la condición saldrá del DO.

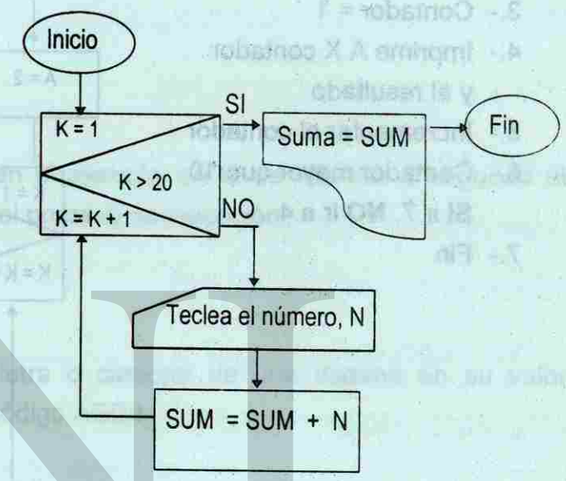
a.- Ejemplos

Ejemplo No.1.- Elabora el diagrama de flujo, la codificación QB de un programa que capture 20 números y obtenga la sumatoria de ellos

ALGORITMO

1.- Inicio
2.- Contador = 1
3.- Pregunta número
4.- Suma el nuevo número
5.- Incrementa el contador
6.- Contador mayor que 20
SI 1r 7, **NO** ir 3
7.- Imprimir suma
8.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

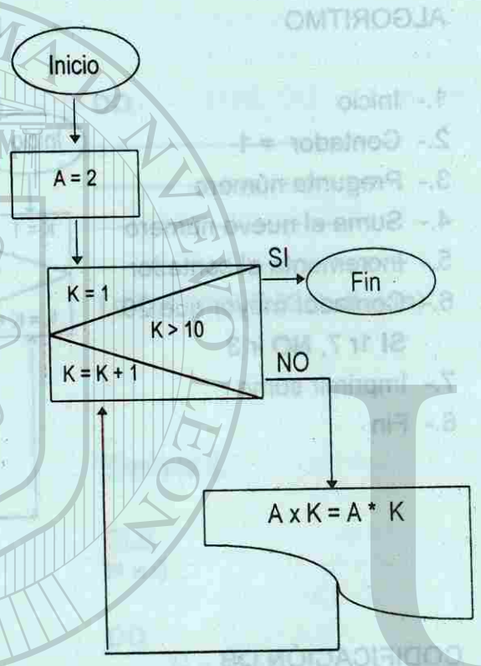
```
CLS
ACUM=0
J=0

DO WHILE J < 20
      J=J+1
      LOCATE 5, 10: PRINT J
      LOCATE 5, 20 : PRINT SPC(50);
      LOCATE 5,20 : INPUT " El  número a sumar es:  ", NUM
      ACUM=ACUM+NUM
LOOP

LOCATE 7,10 : PRINT " La suma total es: " , ACUM
END
```

Ejemplo No. 2.- Elabora el diagrama de flujo, la codificación QB de un programa que muestre en la pantalla de tu monitor la tabla de 2 .

ALGORITMO

1.- Inicio
2.- A = 2
3.- Contador = 1
4.- Imprime A X contador y el resultado
5.- Incrementar el contador
6.- Contador mayor que 10 **SI** ir 7, **NO** ir a 4
7.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
D = 0 : A = 2
LOCATE 5,34  : PRINT "La tabla del 2 "

DO
     D = D + 1
     LOCATE 7+D, 33 : PRINT A ; "  X  " ; D ; "  =  "; A*D
LOOP UNTIL D = 10

END
```


b.- Nuevas Instrucciones.

Ahora veremos nuevas instrucciones que se pueden aplicar en QB, con las cuales, después ver su definición, formato y ejemplo, realizaremos algunos programas para explicarlas mejor.

- **SLEEP**.- Sirve para retardar el tiempo en QB.

  Formato: SLEEP(expr)
  Ejemplo: SLEEP (5)

La función de la instrucción en el ejemplo, es la de retardar 5 segundo el programa y despues continuar con el proceso de ejecución.

- **ASC**.- Convierte la primera letra o caracter de una cadena en su valor numérico correspondiente al código ASCII.

  Formato: ASC (string o variable cadena)
  Ejemplo: X$ = "Examen"
  B = ASC(X$)

En el ejemplo el valor de B sera igual a 69, porque toma el valor numérico del código ACII de la primera letra de la palabra Examen.

- **CHR$**.- Es la función inversa de ASC, permite generar caracteres alfanuméricos correspondientes a un código dado de 0 a 255.

  Formato: CHR$ (X)
  Ejemplos: CHR$(7) Activa el timbre interno de la computadora.
  CHR$(12) Borra la pantalla.
  CHR$(13) Produce un retorno de carro (Enter).

TECLEA:

```
CLS
H=0

DO
      H = H + 1
      PRINT ASC(CHR$(H)) ; ".- ", CHR$(H)
LOOP UNTIL H = 255

END
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Agrega la instrucción **SLEEP (1)** antes de **LOOP UNTIL H = 255**
- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

En la siguiente página elaboraremos otro programa aplicando las instrucciones explicadas. Antes de teclearlo realiza los pasos necesarios para quitar el programa anterior de la memoria y pantalla de QB.

TECLEA:

```
CLS
I = 0
J = 0

DO
      I = I + 1
      LOCATE I, 1: PRINT CHR$(177)
      LOCATE I, 80: PRINT CHR$(177)
LOOP UNTIL I = 23

DO
      J = J + 1
      LOCATE 1, J: PRINT CHR$(177)
      LOCATE 23, J: PRINT CHR$(177)
LOOP UNTIL J = 80

END
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____

El programa anterior lo utilizaremos más adelante por lo tanto lo grabaremos como **subrutina externa**, realiza los siguientes pasos:

- Agrega antes de la primera línea la etiqueta **10000**
- Cambia la instrucción **END** por **RETURN**
- Presionar las teclas **ALT + F**.
- Seleccionar la opción **Save As ....**
- Con la tecla TAB y la flechas de direccionamiento selecciona tu drive de trabajo y pulsa **Enter**.
- En el recuadro de **File Name** escribe el nombre de MARCO
- Con la tecla TAB y la flechas de direccionamiento selecciona el cuadro **Text - Readable by other programs**.
- Pulsa la tecla TAB, selecciona **< OK >** y pulsa Enter.

TECLEA:

```
CLS
H=0

DO
      H = H + 1
      PRINT ASC(CHR$(H)) ; ".- ", CHR$(H)
LOOP UNTIL H = 255

END
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Agrega la instrucción **SLEEP (1)** antes de **LOOP UNTIL H = 255**
- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

En la siguiente página elaboraremos otro programa aplicando las instrucciones explicadas. Antes de teclearlo realiza los pasos necesarios para quitar el programa anterior de la memoria y pantalla de QB.

TECLEA:

```
CLS
I = 0
J = 0

DO
      I = I + 1
      LOCATE I, 1: PRINT CHR$(177)
      LOCATE I, 80: PRINT CHR$(177)
LOOP UNTIL I = 23

DO
      J = J + 1
      LOCATE 1, J: PRINT CHR$(177)
      LOCATE 23, J: PRINT CHR$(177)
LOOP UNTIL J = 80

END
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____

El programa anterior lo utilizaremos más adelante por lo tanto lo grabaremos como **subrutina externa**, realiza los siguientes pasos:

- Agrega antes de la primera línea la etiqueta **10000**
- Cambia la instrucción **END** por **RETURN**
- Presionar las teclas **ALT + F**.
- Seleccionar la opción **Save As ....**
- Con la tecla TAB y la flechas de direccionamiento selecciona tu drive de trabajo y pulsa **Enter**.
- En el recuadro de **File Name** escribe el nombre de MARCO
- Con la tecla TAB y la flechas de direccionamiento selecciona el cuadro **Text - Readable by other programs**.
- Pulsa la tecla TAB, selecciona **< OK >** y pulsa Enter.

- **LEN**.- Proporciona la longitud de una expresión cadena, que va de 0 hasta 32,767 caracteres.

Formato: LEN (expresión cadena)
Ejemplo: A$ = "Universidad Autónoma De Nuevo León"
B = LEN (A$)

El valor de B en el ejemplo será de 34 porque a parte del número de letras que contiene el letrero, se toman en cuenta los espacios en blanco.

- **LEFT$**.- Es una función subcadena que proporciona los n primeros caracteres de una expresión cadena, contando desde la izquierda.

Formato : LEFT$ (expresión de cadena, n)
Ejemplo: A$="Universidad Autónoma De Nuevo León"
B$ = LEFT$(A$,11)

En el ejemplo el contenido de B$ es "Universidad", porque toma las primeras 11 letras a partir de la izquierda de A$.

- **RIGHT$**.- Devuelve **n** caracteres que se encuentran a la derecha de la cadena.

Formato: RIGHT$ (expresión de cadena, n).
Ejemplo: A$ = "Universidad Autónoma De Nuevo León"
B$ = RIGHT$ (A$,5)

El contenido de B$ en el ejemplo es "León", porque toma las primeras 5 letras a partir de la derecha de A$.

- **MID$**.- Es una función subcadena que extrae una parte del interior de una expresión alfanumérica; n es el primer caracter que va a tomar a partir de la izquierda; m es el primer caracter que va a tomar a partir de la izquierda.

Formato: MID$ (expresión de cadena, n, [m]).
Ejemplo: A$="Universidad Autónoma De Nuevo León"
B$=MID$ (A$, 13, 8)

En el ejemplo el contenido de B$ es "Autónoma", porque toma a partir del caracter 13 de la derecha de A$ las siguientes 8 letras.

- **INSTR.**- Localiza un string dentro de otro y si lo encuentra, regresa al lugar que ocupa en el string secundario, si no lo encuentra regresa un cero (0), r es el lugar donde se iniciará la búsqueda, si se omite se toma como uno.

Formato: INSTR(string principal, string secundaria, r).
Ejemplo: A$="Universidad Autónoma De Nuevo León"
B$="De"
X = INSTR(A$,B$)
Y = INSTR (A$, 13,"o")

En el ejemplo el valor de X es igual a 22, porque a partir del primer caracter, B$ ocupa el lugar 22; el valor de Y es igual a 6, porque a partir del caracter 13 la letra "o" ocupa el lugar 19 (la letra "ó" es diferente a "o" para la computadora)

- **LINE INPUT.**- Instrucción para recibir datos, en cual se puede incluir la coma (,); la inclusión de la coma no es posible con el INPUT

Formato: LINE INPUT; variable
Ejemplo: Variable = Universidad, Autónoma, De, Nuevo, León

En el ejemplo, la información que recibe la variable, contiene una o más comas y se toma como un solo dato.

TECLEA

```
CLS
DO
      CLS
      LOCATE 1, 15: PRINT "Escribe tu nombre separandolo con comas"
      LOCATE 2, 22: PRINT "una antes de cada apellido"
      LOCATE 4, 15: PRINT "Ejemplo: JOSE LUIS,GARZA,MARTINEZ"
      LOCATE 6, 20: A$ = "": LINE INPUT "Cuál es tu nombre: "; A$
      A = LEN(A$)
      LOCATE 10, 10: PRINT "El nombre que me diste tiene ";  A; " caracteres"
      B1 = INSTR(A$, ",")
      N$ = LEFT$(A$, B1 - 1)
      LOCATE 12, 10: PRINT N$; " tiene"; B1 - 1; "caracteres"
      AP$ = RIGHT$(A$, A - B1)
      B2 = INSTR(AP$, ",")
      M$ = MID$(A$, B1 + 1, B2 - 1)
      LOCATE 14, 10: PRINT M$; " tiene"; B2 - 1; "caracteres"
      AP = A - (B1 + B2)
      P$ = RIGHT$(A$, AP)
      LOCATE 16, 10: PRINT P$; " tiene "; AP
      LOCATE 18, 10: PRINT "y dos comas (,)"
      LOCATE 21, 25: INPUT " Deseas dar otro nombre (S/N) "; D$
      IF UCASE$(D$) = "N" THEN EXIT DO
LOOP
END
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

$INCLUDE.- Sirve para incluir un programa o subrutinas que están fuera de un programa, es decir, programas o subrutinas externas.

Formato: REM $INCLUDE: 'nombre de archivo '
´$INCLUDE: 'nombre del archivo '
Ejemplo: REM $INCLUDE: 'B:MARCO.BAS'

En el ejemplo la instrucción dice que hay que incluir en el programa principal un programa o subrutina que se encuentra en el drive B que se llama MARCO.BAS

TECLEA:

```
CLS
A = 0: B=0

DO
      CLS : GOSUB 10000
      A = A + 1
      LOCATE 5, 34: PRINT "La tabla del "; A

      DO
            B = B + 1
            LOCATE 7 + B, 34: PRINT A; " X "; B; " = "; A * B
            SLEEP (1)
      LOOP WHILE B < 10

LOOP WHILE A < 10

END

REM $INCLUDE: 'B:MARCO.BAS'
```

- Corre el programa (SHIFT + F5).
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

c.- Ejercicios

Desarrolla el algoritmo, el diagrama de flujo y la codificación QB para los siguientes programas.

1.- Realiza un programa que capture **N** números y obtenga la sumatoria de ellos

2.- Realiza un programa que muestre la lista del código ASCII desde el término **N** hasta el término **M**.

3.- **Realiza un programa que muestre en la pantalla del monitor un marco con el asterisco (*) y ponga en el centro tus datos.**

MATRICULA:
NOMBRE:
GRUPO:
CANTANTE FAVORITO:
CANCIÓN PREFERIDA:

☺ EJERCICIO GENERAL:

1.- Elabora un programa que determine el valor de las funciones Seno, Coseno, Tangente, Cotangente, Secante y Cosecante e imprima sus resultados, el valor del ángulo estará en grados y radianes.

GRADOS	RADIANES
Seno =	Seno =
Coseno =	Coseno =
Tangente =	Tangente =
Cotangente =	Cotangente =
Secante =	Secante =
Cosecante =	Cosecante =

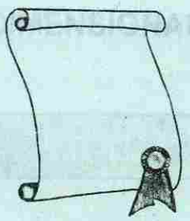
2.- Realiza un programa que a partir del nombre de un alumno y una calificación determine e imprima el nombre y si su calificación es:

EXCELENTE: 100	BUENA: 70 a 84
MUY BUENA: 85 a 99	MALA: menos de 70

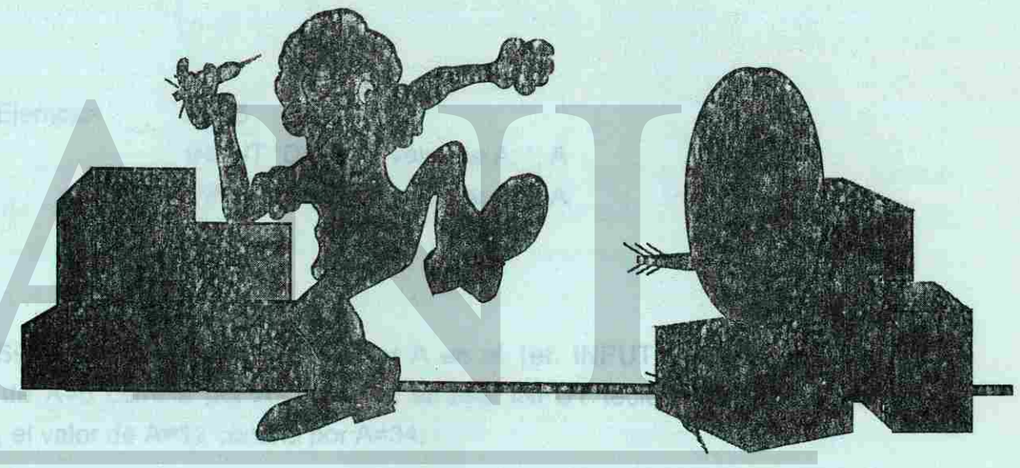
3.- Realiza los pasos necesarios para que en el ejercicio anterior se pueda capturar N nombres, pero que sólo pueda trabajar con uno a la vez (hacerlo cíclico).

UNIDAD III

VARIABLES DIMENSIONADAS



Can$



VARIABLES DIMENSIONADAS

UNIDAD III

VARIABLES DIMENSIONADAS

1.-DIMENSIONAMIENTO.

A.- CONCEPTO DE VECTOR

Durante un programa se utilizan variables numéricas o alfanuméricas, las cuales durante el proceso o ejecución del programa, pueden tomar varios valores, pero almacenan un sólo valor.

Ejemplo:

```
A = 5
INPUT "Dame el valor de A: ", A
INPUT "Dame el valor de A: ", A
```

Si tecleamos 12 al pedir el valor A en el 1er. INPUT y pulsamos Enter, el valor de A=5 cambia por A=12; si en el 2do. INPUT tecleamos 34 y pulsamos Enter, el valor de A=12 cambia por A=34;

Con lo anterior podemos observar que una variable numérica o alfanumérica tendrá en memoria un sólo valor.

Un **vector** es un arreglo unidimensional, en el cual podemos almacenar diferentes valores en una misma variable numérica o alfanumérica, que durante el proceso del programa se pueden utilizar.

Hay que señalar que durante el proceso o ejecución del programa el vector puede almacenar diferentes valores, pero al momento de reiniciar el programa, la información que se utilizó se perderá.

B. -PARTES DE UN VECTOR

Las partes del vector son:

a.- Nombre.
b.- Número de la posiciones.
c.- Datos almacenados (éstos pueden ser numéricos o alfanuméricos).

Ejemplo 1

NUM (5)

1	15
2	30
3	12
4	4
5	16

Ejemplo 2

NOM$ (6)

1	Ricardo
2	Raul
3	Araceli
4	Nelly
5	Yani
6	Denisse

	Ejemplo 1	Ejemplo 2
Nombre del vector:	NUM	NOM$
Número de posiciones:	5	6
Datos almacenados:	NUM(1)=15	NOM$(1)="Ricardo"
	NUM(2)=30	NOM$(2)="Raul"
	NUM(3)=12	NOM$(3)="Araceli"
	NUM(4)=4	NOM$(4)="Nelly"
	NUM(5)=16	NOM$(5)="Yani"
		NOM$(6)="Denisse"

C.- INSTRUCCIÓN PARA UN VECTOR.

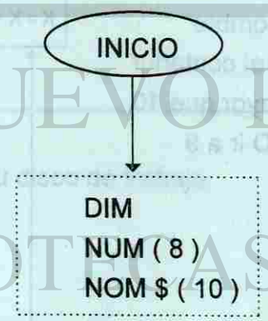
En QB al crear un vector es necesario declararlo y esto se logra mediante la instrucción DIM, donde se especificará el nombre del vector y el total de posiciones que tendrá.

- **DIM**.- Esta instrucción permite fijar el tamaño de la memoria que hay que reservar

Formato: DIM variable1 (n) , variable2 (m), variable3 (p)
Ejemplo: DIM NUM (8), NOM$ (10)

Para representar el diagrama de flujo para los dos vectores, uno llamado NUM con **8** posiciones de memoria para **8** elementos numéricos, y otro llamado NOM$ con **10** posiciones de memoria para **10** elementos alfanuméricos quedaría:

DIAGRAMA DE FLUJO



2.- FORMAS DE LLENAR UN VECTOR.

A.- POR LECTURA.

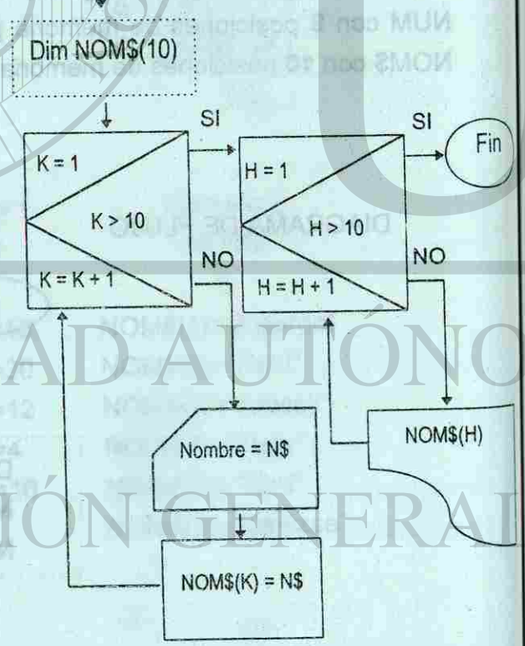
Al proceso de asignarle información a un vector mediante la captura de datos desde el teclado y/o de un disco se le llama llenado por lectura.

Ejemplo No.1.- Programa que a partir del primer nombre de diez personas, se guardarán de uno por uno en un vector llamado NOM$ (éste se llenará mediante un control de ciclos) y por último dará el reporte total de los nombres.

ALGORITMO

1.- Inicio
2.- Especificar vector
3.- Iniciar contador
4.- Preguntar nombre
5.- Guardar en el vector
6.- Incrementar el contador
7.- Contador mayor que 10
SI ir a 8, **NO** ir a 4
8.- Iniciar contador
9.- Imprimir el nombre
10.- Incrementar el contador
11.- Contador mayor que 10
SI ir a 12, **NO** ir a 9
12.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
DIM NOM$(10)
K = 0

DO
      K = K + 1
      LOCATE 5, 25: PRINT SPC(50);
      LOCATE 5, 20: PRINT K
      LOCATE 5, 25: INPUT "Dame el nombre: ", NOM$(K)
      NOM$(K) = UCASE$(NOM$(K))
LOOP UNTIL K = 10

LOCATE 20, 15: PRINT "Pulsa cualquier tecla para observar los nombres"

5     X$ = INKEY$
      IF X$ = "" THEN 5 ELSE 10

10    CLS : H = 0
      DO
            H = H + 1
            LOCATE 4 + H, 30: PRINT H; ".- "; NOM$(H)
      LOOP WHILE H < 10

      END
```

- Corre el programa (SHIFT + F5).
- Dale los datos que pide.
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

B.- POR OPERACIÓN.

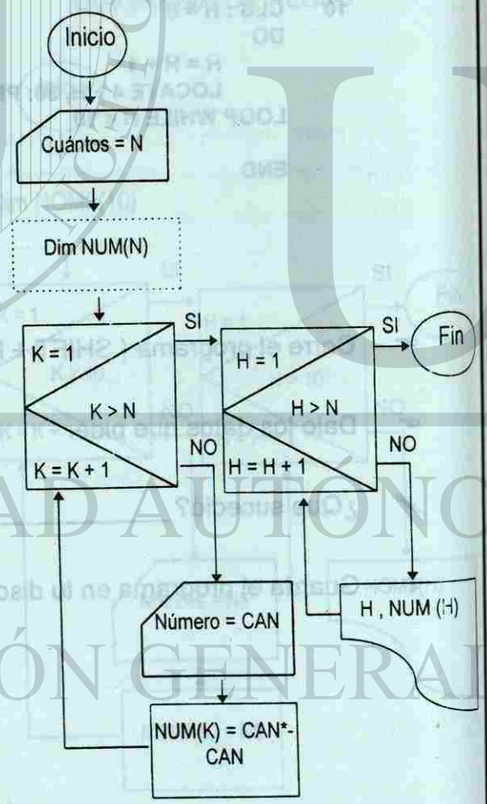
También es posible guardar información en un vector mediante la realización de una operación aritmética; una vez calculado el resultado de la operación durante el proceso, se deberá indicar la posición del vector en donde se almacenará dicho resultado.

Ejemplo No.1.- Programa que pida **N** números, de los cuales encontraremos el cuadrado de cada uno de ellos, utilizaremos un vector para guardar los resultados y al final dará el reporte del vector .

ALGORITMO

1.- Inicio
2.- Pregunta cuantos números, **N**
3.- Especificar vector
4.- Iniciar contador (K)
5.- Pregunta número
6.- Guardar en el vector el cuadrado del número
7.- Incrementar el contador
8.- Contador mayor que **N** **SI** ir a 9, **NO** ir a 5
9.- Iniciar contador (H)
10.- Imprimir el cuadrado
11 Incrementar el contador
12.- Contador mayor que **N** **SI** ir a 13, **NO** ir a 10
13.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
LOCATE 5, 20: INPUT "Dame el total de números a trabajar: ", N
DIM NUM(N)
CLS : K = 0

DO
      K = K + 1
      LOCATE 5, 35: PRINT SPC(50);
      LOCATE 5, 20: PRINT K
      LOCATE 5, 25: INPUT "Dame el número: ", CAN
      CLS
      NUM(K) = CAN * CAN
LOOP UNTIL K = N

      LOCATE 20, 15: PRINT "Pulsa cualquier tecla para observar los números"

5     X$ = INKEY$
      IF X$ = "" THEN 5 ELSE 10

10    CLS : H = 0
      DO
            H = H + 1
            LOCATE 4 + H, 30: PRINT H; ".- "; NUM(H)
      LOOP WHILE H < N

      END
```

- Corre el programa (SHIFT + F5).
- Dale los datos que pide.
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

B.- POR OPERACIÓN.

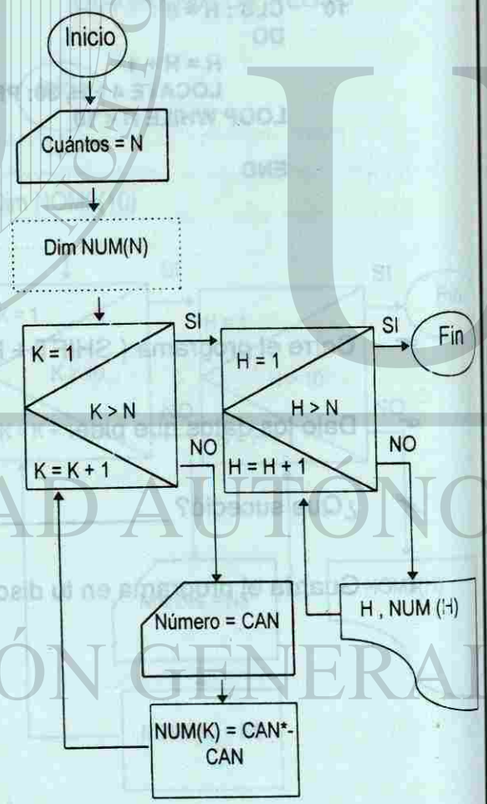
También es posible guardar información en un vector mediante la realización de una operación aritmética; una vez calculado el resultado de la operación durante el proceso, se deberá indicar la posición del vector en donde se almacenará dicho resultado.

Ejemplo No.1.- Programa que pida **N** números, de los cuales encontraremos el cuadrado de cada uno de ellos, utilizaremos un vector para guardar los resultados y al final dará el reporte del vector .

ALGORITMO

1.- Inicio
2.- Pregunta cuantos números, **N**
3.- Especificar vector
4.- Iniciar contador (K)
5.- Pregunta número
6.- Guardar en el vector el cuadrado del número
7.- Incrementar el contador
8.- Contador mayor que **N** **SI** ir a 9, **NO** ir a 5
9.- Iniciar contador (H)
10.- Imprimir el cuadrado
11 Incrementar el contador
12.- Contador mayor que **N** **SI** ir a 13, **NO** ir a 10
13.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
LOCATE 5, 20: INPUT "Dame el total de números a trabajar: ", N
DIM NUM(N)
CLS : K = 0

DO
      K = K + 1
      LOCATE 5, 35: PRINT SPC(50);
      LOCATE 5, 20: PRINT K
      LOCATE 5, 25: INPUT "Dame el número: ", CAN
      CLS
      NUM(K) = CAN * CAN
LOOP UNTIL K = N

      LOCATE 20, 15: PRINT "Pulsa cualquier tecla para observar los números"

5     X$ = INKEY$
      IF X$ = "" THEN 5 ELSE 10

10    CLS : H = 0
      DO
            H = H + 1
            LOCATE 4 + H, 30: PRINT H; ".- "; NUM(H)
      LOOP WHILE H < N

      END
```

- Corre el programa (SHIFT + F5).
- Dale los datos que pide.
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

C.- POR LECTURA Y OPERACIÓN.

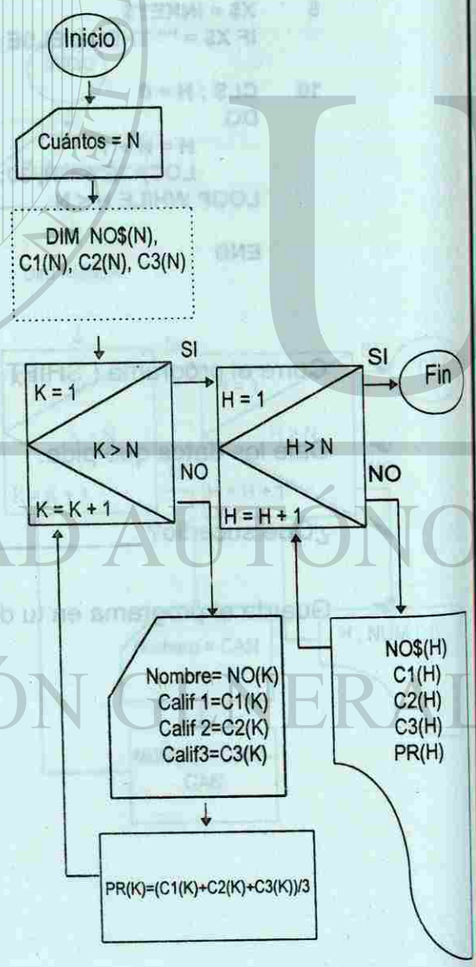
Ahora en los programas ha realizar, se almacenarán la información en vectores por medio de lectura y operación.

Ejemplo No.1.- Programa en donde a partir de **N** alumnos tomaremos el primer nombre y las calificaciones de 3 examenes parciales, los cuales guardaremos en varios vectores y al final daremos un reporte con sus nombres, calificaciones parciales y su promedio final.

ALGORITMO

1.- Inicio
2.- Pregunta cuantos alumnos, **N**
3.- Especificar los vectores
4.- Iniciar contador (K)
5.- Pregunta nombre guardándolo en el vector NO$(K)
6.- Pregunta calificaciones guardándolas en los vectores C1(K), C2(K), C3(K)
7.- Guarda el promedio en el vector PR(K)
8.- Incrementar el contador (K)
9.- Contador mayor que **N** **SI** ir a 10, **NO** ir a 5
10.- Iniciar contador (H)
11.- Imprimir nombre, tres calificaciones y el promedio
12 Incrementar el contador (H)
13.- Contador mayor que **N** **SI** ir a 14, **NO** ir a 11
14.- Fin

DIAGRAMA DE FLUJO



CODIFICACIÓN QB

```
CLS
LOCATE 5, 20: INPUT "Dame el total de alumnos: ", N
DIM NO$(N), C1(N), C2(N), C3(N), PR(N)
CLS : K = 0

DO
      K = K + 1
      LOCATE 5, 35: PRINT SPC(50);
      LOCATE 5, 20: PRINT K
      LOCATE 5, 25: INPUT "Dame el nombre del alumno: ", NO$(K)
      LOCATE 7, 25: INPUT "Dame la 1a. calificación: ", C1(K)
      LOCATE 9, 25: INPUT "Dame la 2a. calificación: ", C2(K)
      LOCATE 11, 25: INPUT "Dame la 3a. calificación: ", C3(K)
      CLS
      NO$(K) = UCASE$(NO$(K))
      PR(K) = (C1(K) + C2(K) + C3(K)) / 3
LOOP UNTIL K = N

LOCATE 20, 15: PRINT "Pulsa cualquier tecla para observar el reporte"

5     X$ = INKEY$
      IF X$ = "" THEN 5 ELSE 10

10    CLS : H = 0
      LOCATE 3, 8: PRINT " NOMBRE "
      LOCATE 3, 35: PRINT "1a.CAL" : LOCATE 3, 45: PRINT "2da.CAL"
      LOCATE 3, 55: PRINT "3a.CAL" : LOCATE 3, 66: PRINT "PROMEDIO"

      DO
            H = H + 1
            LOCATE 4 + H, 2: PRINT H; ".- "; NO$(H)
            LOCATE 4 + H, 36: PRINT C1(H) : LOCATE 4 + H, 46: PRINT C2(H)
            LOCATE 4 + H, 56: PRINT C3(H) : LOCATE 4 + H, 65: PRINT PR(H)
      LOOP WHILE H < N

      END
```

- Corre el programa (SHIFT + F5).
- Dale los datos que pide.
- ¿Que sucedió? _____
- Guarda el programa en tu disco de trabajo.

D.- EJERCICIOS

En los siguientes programas elabora el algoritmo, el diagrama de flujo y la codificación en QB.

1.- Una línea aérea cuenta con N clientes y vende boletos a cinco ciudades. Por políticas de esta compañía a cada cliente se le puede vender la cantidad de boletos que él desee, pero siempre y cuando sean a un sólo destino. Elabora un programa que permita capturar el nombre del cliente, la cantidad de boletos y la ciudad para dar el reporte del nombre y el pago total de cada uno de los clientes

CIUDAD	PRECIO
Can - Cún	$ 750.00
Mazatlán	$ 530.00
Acapulco	$ 585.00
Manzanillo	$ 470.00
Huatulco	$ 630.00

2.- Una agencia de detectives cuenta con una cierta cantidad de clientes, a los cuales les han resuelto sus casos, cobra dependiendo de la cantidad de horas trabajadas y además hace un descuento del 50% si la persona trae credencial de estudiante. Elabora un programa que capture el nombre del cliente, las horas trabajadas y si utiliza credencial para dar el reporte del nombre y el total a pagar.

1 Hora trabajada = $50.00

3.- Una tienda de discos tiene ofertas en dos departamentos, en cada uno de ellos se venden Compact Disc al mismo precio, a los clientes se les vende la cantidad de articulos que desee pero de un solo departamento, elabora un programa que a partir del N clientes, departamento de compra y cantidad de articulos; para dar un reporte del nombre, total a pagar de cada uno de los clientes y el total recaudado en cada departamento .

Departamento de Discos	Precio del Compact Disc
ESPAÑOL	$ 40.00
INGLES	$ 50.00

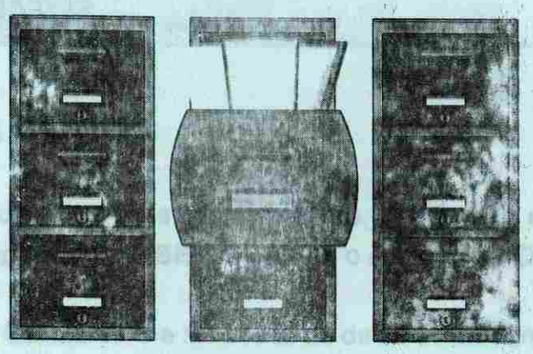
☺ EJERCICIO GENERAL

1.- Elabora en forma manual una lista de los nombres de tus campañeros de clase.

2.- Pideles la calificación que obtuvieron en computación en el curso anterior y anotala en la lista

3.- Elabora un programa utilizando vectores para capturar N alumnos y la calificación de cada uno de ello; el reporte del programa imprimirá de cinco en cinco los nombre de tus compañeros y en la parte final que imprima el promedio final del grupo.

BASE DE DATOS



MENÚ

BAJAS

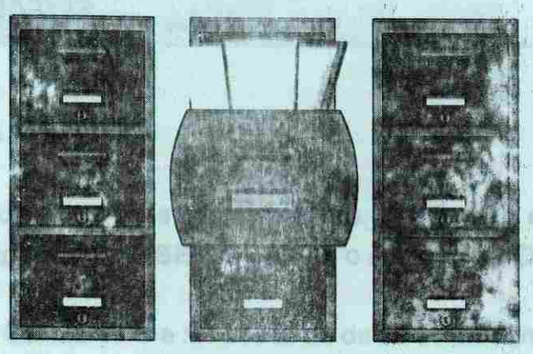
CONSULTA

ALTAS

CAMBIOS

MANEJO DE ARCHIVOS

BASE DE DATOS



MENÚ

BAJAS

CONSULTA

ALTAS

CAMBIOS

MANEJO DE ARCHIVOS

UNIDAD IV

MANEJO DE ARCHIVOS

1.- BASE DE DATOS.

A.- CONCEPTOS BÁSICOS

Al conjunto de datos de la misma especie, clasificados de acuerdo a ciertas características, se le llama **BASE DE DATOS** o **ARCHIVOS DE DATOS**.

En BASIC y QuickBASIC a las bases de datos se les llama archivos de datos y por lo general llevan como extensión **.DAT**. En este capítulo llamaremos archivos de datos a las bases de datos, para no confundirnos.

En general las acciones que se pueden dar con un archivo de datos son básicamente siete:

- ☑ Abrir.
- ☑ Agregar información.
- ☑ Dar de baja (borrar).
- ☑ Manipular (cambiar, modificar).
- ☑ Leer.
- ☑ Consultar
- ☑ Cerrar.

Un ejemplo de BASE DE DATOS sería el directorio telefónico, pues contiene datos clasificados de la misma especie (nombre, dirección, colonia, código postal y teléfono) de los abonados a TEL-MEX; el más común de los directorios, sección blanca, tiene clasificados a los abonados mediante un orden alfabético de apellido. A cada abonado le llamaremos **REGISTRO** y a cada uno de los datos que tiene cada registro se le llama CAMPO; así en este caso cada registro tiene cinco datos o cinco CAMPOS; 1) nombre, 2) dirección, 3) colonia, 4) código postal, 5) teléfono.

B.- TIPOS DE ARCHIVOS DE DATOS

QB manejan dos tipos de archivos, los secuenciales y los aleatorios.

a.- Secuenciales

Los archivos secuenciales guardan la información en forma consecutiva es decir un dato seguido de otro separados por un caracter especial, que indica el fin de un dato; lo que permite no dejar espacios en blanco entre uno y otro con un ahorro de espacio en disco. Tienen una **desventaja**, para leer un dato hay que leer todos los anteriores, por ejemplo si busco los datos del registro 35 tiene que leer los 34 primeros, si deseo el registro 479 necesito leer del 1 al 478 para poder encontrar el 479. Por ejemplo:

	Monterrey	,	Guadalajara	,	México	,	Veracruz	,	Monclova	,	Saltillo
Caracteres:	9	1	11	1	6	1	8	1	8	1	8

Total de caracteres: **55**

b.- Aleatorios

Los archivos aleatorios o directos o RANDOM guardan todos los registros de igual tamaño, se utilicen o no todos los lugares; para leer un registro podemos leer cualquiera de ellos directamente sin pasar o leer los anteriores ahorrando tiempo de lectura. Tienen **desventajas** producen lugares vacíos o datos incompletos, además genera archivo de mayor tamaño que los secuenciales. Por ejemplo:

	Monterrey	
	Guadalajar	
	México	
	Veracruz	
	Monclova	
	Saltillo	
Caracteres	10	Total de caracteres: **60**

C.- INSTRUCCIONES PARA MANEJO DE ARCHIVOS

- **OPEN**.- Esta instrucción sirve para abrir un archivo de datos ya existente o crear uno nuevo.

  Formato : OPEN "nombre del archivo" FOR RANDOM AS n LEN = m
  Ejemplo: OPEN "B:DATOS.DAT" FOR RANDOM AS 3 LEN = 32

En el ejemplo la computadora abre, del drive B, el archivo "DATOS.DAT" de tipo RANDOM por la línea de comunicación 3 mediante la cual establecemos comunicación entre la computadora y el disco; como se pueden abrir varias bases de datos de un mismo disco es necesario que la línea de comunicación sea diferente en cada una para evitar una interferencia o choque de información; LEN indica la longitud de cada registro, 32 bytes.

- **FIELD**.- Esta instrucción sirve para especificar cuántos campos tiene cada registro, cuales son y de qué longitud es cada uno de ellos.

  Formato : FIELD n, m AS var1$, p AS var2$, q AS var3$,...
  Ejemplo: FIELD 3, 5 AS C1$, 20 AS C2$, 7 AS C3$

En el ejemplo el "3" es el mismo número de línea de comunicación que se utilizó en la instrucción OPEN; 5 caracteres es el tamaño del campo C1$, 20 caracteres del campo C2$ y 7 caracteres del campo C3$; la suma de los caracteres en los campos C1$,C2$ y C3$ no debe pasar del tamaño de LEN = 32, ya que marcaría un error de desbordamiento del buffer.

Al abrir un archivo, BASICA y/o QB separan en memoria una cantidad de bytes igual al tamaño del registro, a este lugar de memoria temporal se le llama **buffer**, el cual sirve para acomodar los campos de nuestro registro, ya sea para guardar en el disco o al leer del disco. Existen tantos buffers activos como archivos estén abiertos, uno cada archivo y pueden ser cada uno de distinto tamaño.

LOF.- Esta instrucción indica la cantidad de bytes que se han utilizado en el disco para el archivo de datos que abrimos.

Formato: LOF(n)
Ejemplo: A = LOF(3) Cantidad total de bytes en el archivo de datos

LOF (3) dará la cantidad de caracteres del archivo de datos abierto por la línea "3".

Ejemplo: W = A / 32 Cantidad de registros en el archivo de datos

Si esta cantidad la dividimos entre 32, la cantidad de bytes que tiene un registro, obtenemos la cantidad de registros que tiene nuestro archivo de datos.

LSET - RSET.- Instrucciones para acomodar la información en cada campo del buffer, justificados a la derecha o a la izquierda.

Formato :LSET Var1$ = datos a guardar
RSET Var2$ = datos a guardar

Ejemplo: LSET C1$ = " DENISSE "
RSET C2$ = " DENISSE "

En el campo **C1$** la información se guarda a partir de la izquierda.

D	E	N	I	S	S	E			

En el campo **C2$** la información se guarda a partir de la derecha.

			D	E	N	I	S	S	E

PUT.- Instrucción para guardar en el disco, los datos del buffer del registro.

Formato: PUT n, r
Ejemplo: PUT 3,12

En el ejemplo el "3" es el número de línea de comunicación con que abrimos el archivo y "12" es el número de registro o renglón en que lo vamos a guardar.

GET.- Instrucción para leer y traer un registro especifico, coloca en la memoria temporal (buffer) toda la información del registro, para poder utilizarla o manipularla según nuestras necesidades.

Formato: GET n, r
Ejermplo: GET 3, 8

En el ejemplo el "3" es el número de línea de comunicación con que abrimos el archivo y "8" es el registro o renglón que vamos a leer.

CLOSE.- Esta instrucción sirve para cerrar nuestros archivos; es muy conveniente cerrarlo, pues si se nos olvida hacerlo puede perderse información o llenarse de basura.

Formato: CLOSE n
Ejemplos: CLOSE Cierra todos los archivos
CLOSE 1 Cierra el archivo abierto por la línea 1.
CLOSE 2,5 Cierra los archivos abiertos por la 2 y 5.

2.- APLICACIONES.

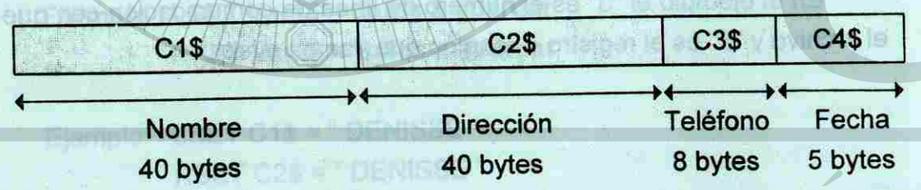
Ahora con todas las instrucciones que hemos visto, realizaremos un ejercicio para elaborar un programa que genere un archivo de datos para dar de alta, al cual lo llamaremos AGENDA.DAT y en él guardarmos cuatro datos en cada registro: nombre, dirección, teléfono y fecha de cumpleaños.

La cantidad de bytes que usaremos en cada campo será:

- ☑ 40 bytes para el nombre.
- ☑ 40 bytes para la dirección.
- ☑ 8 para el teléfono (nnn-nnnn).
- ☑ 5 para la fecha de cumpleaños.

93 bytes en total.

REGISTRO:



Ejemplo:

Nombre: PRISCILA ALEMAN ÁLVAREZ
Dirección: PASEO DE SANTA LUCÍA # 27
Teléfono: 345-6789
Fecha de cumpleaños: 07NOV

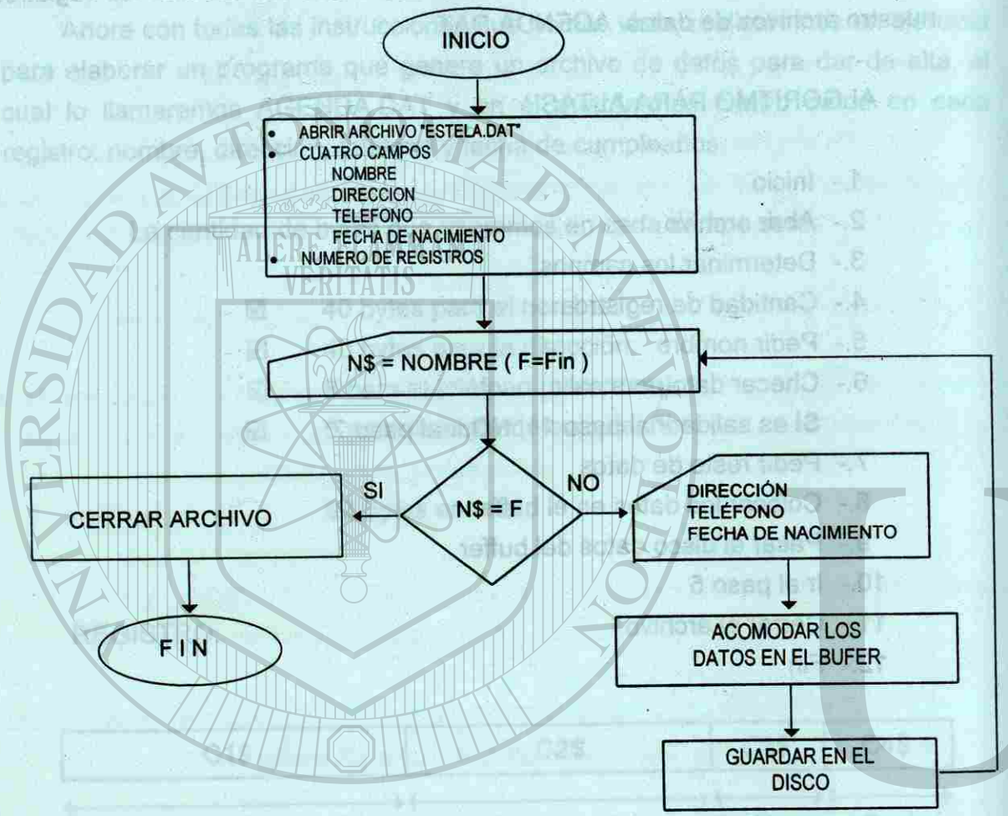
A.- PROGRAMA PARA ALTAS.

El programa para altas nos va a permitir dar de alta los registros en nuestro archivos de datos AGENDA.DAT

ALGORITMO PARA ALTAS

1.- Inicio
2.- Abrir archivo
3.- Determinar los campos
4.- Cantidad de registros
5.- Pedir nombre
6.- Checar dato para salir
SI es salida ir al paso 11, **NO** ir al paso 7
7.- Pedir resto de datos
8.- Colocar los datos en el buffer
9.- Pasar al disco datos del buffer
10.- Ir al paso 5
11.- Cerrar el archivo
12.- Fin

DIAGRAMA DE FLUJO PARA ALTAS



CODIFICACIÓN QB PARA ALTAS

```
OPEN "B:AGENDA.DAT" FOR RANDOM AS 1 LEN = 93
FIELD 1, 40 AS C1$, 40 AS C2$, 8 AS C3$, 5 AS C4$
W = LOF(1) / 93

DO
        CLS : GOSUB 10000
        LOCATE 3, 26: PRINT "ALTAS en la agenda de MARIA ESTELA"
        LOCATE 21, 31: PRINT "TOTAL DE REGISTROS: "; W
        LOCATE 8, 5: NO$ = "": INPUT " Nombre (F = Fin): ", NO$
        NO.- $ = UCASE$(NO$)
        IF NO$ = "F" THEN EXIT DO ELSE 5

5       LOCATE 10, 5: DI$ = "": INPUT " Dirección: ", DI$
        LOCATE 12, 5: TE$ = "": INPUT " Teléfono (nnn-nnnn): ", TE$
        LOCATE 14, 5: FE$ = "": INPUT " Fecha (ddmmm): ", FE$
        K = 1: R$ = ""

        DO
                GET 1, K
                IF MID$(C1$, 1, 5) = "     " THEN GOSUB 50 ELSE 10
                PUT 1, K: R$ = "1": EXIT DO
10              K = K + 1
        LOOP UNTIL K > W

        IF R$ = "1" THEN 20 ELSE GOSUB 50
        W = W + 1: PUT 1, W
20      CLS
LOOP

CLOSE
END

50  LSET C1$ = NO$
    LSET C2$ = DI$
    LSET C3$ = TE$
    LSET C4$ = FE$
    RETURN

    REM  $INCLUDE: 'B:MARCO.BAS'
```

Guárdalo en tu disco de trabajo como ALTAS.BAS

Ejercicio: Da de alta en la agenda, cuando menos, a veinte de tus amigos.

B.- PROGRAMA PARA BAJAS.

El programa de bajas nos va a permitir borrar definitivamente algún registro de nuestra AGENDA.

ALGORITMO PARA BAJAS

1.- Inicio
2.- Abrir archivo
3.- Determinar los campos
4.- Cantidad de registros = W
5.- Pedir nombre
6.- Checar dato para salir
SI es salida ir al paso 19, **NO** ir a 8
7.- Incremetar el contador
8.- Contador mayor que W
SI ir al paso 9, **NO** ir al paso 11
9.- Imprime "no encontré a " NO$
10.- Ir al paso 5
11.- Leer el registro del disco
12.- Revisar si el registro contiene el nombre
SI contiene el nombre ir a 13, **NO** lo contiene ir a 7
13.- Presentar los datos.
14.- Preguntar si es la persona que se busca.
SI es la persona ir a 15, **NO** es ir a 7
15.- Asegurar si es baja (S/N)
SI es BAJA ir a 16, **NO** es ir a 5
16.- Borrar los campos
17.- Guardar el registo
18.- Ir a 5
19.- Cerrar el archivo
20.- Fin

CODIFICACIÓN QB PARA ALTAS

```
OPEN "B:AGENDA.DAT" FOR RANDOM AS 1 LEN = 93
FIELD 1, 40 AS C1$, 40 AS C2$, 8 AS C3$, 5 AS C4$
W = LOF(1) / 93

DO
      CLS : GOSUB 10000
      LOCATE 3, 26: PRINT "ALTAS en la agenda de MARIA ESTELA"
      LOCATE 21, 31: PRINT "TOTAL DE REGISTROS: "; W
      LOCATE 8, 5: NO$ = "": INPUT " Nombre (F = Fin): ", NO$
      NO.- $ = UCASE$(NO$)
      IF NO$ = "F" THEN EXIT DO ELSE 5

5     LOCATE 10, 5: DI$ = "": INPUT " Dirección: ", DI$
      LOCATE 12, 5: TE$ = "": INPUT " Teléfono (nnn-nnnn): ", TE$
      LOCATE 14, 5: FE$ = "": INPUT " Fecha (ddmmm): ", FE$
      K = 1: R$ = ""

      DO
            GET 1, K
            IF MID$(C1$, 1, 5) = "     " THEN GOSUB 50 ELSE 10
            PUT 1, K: R$ = "1": EXIT DO
10          K = K + 1
      LOOP UNTIL K > W

      IF R$ = "1" THEN 20 ELSE GOSUB 50
       W = W + 1: PUT 1, W
20    CLS
LOOP

CLOSE
END

50    LSET C1$ = NO$
      LSET C2$ = DI$
      LSET C3$ = TE$
      LSET C4$ = FE$
      RETURN

REM  $INCLUDE: 'B:MARCO.BAS'
```

Guárdalo en tu disco de trabajo como ALTAS.BAS

Ejercicio: Da de alta en la agenda, cuando menos, a veinte de tus amigos.

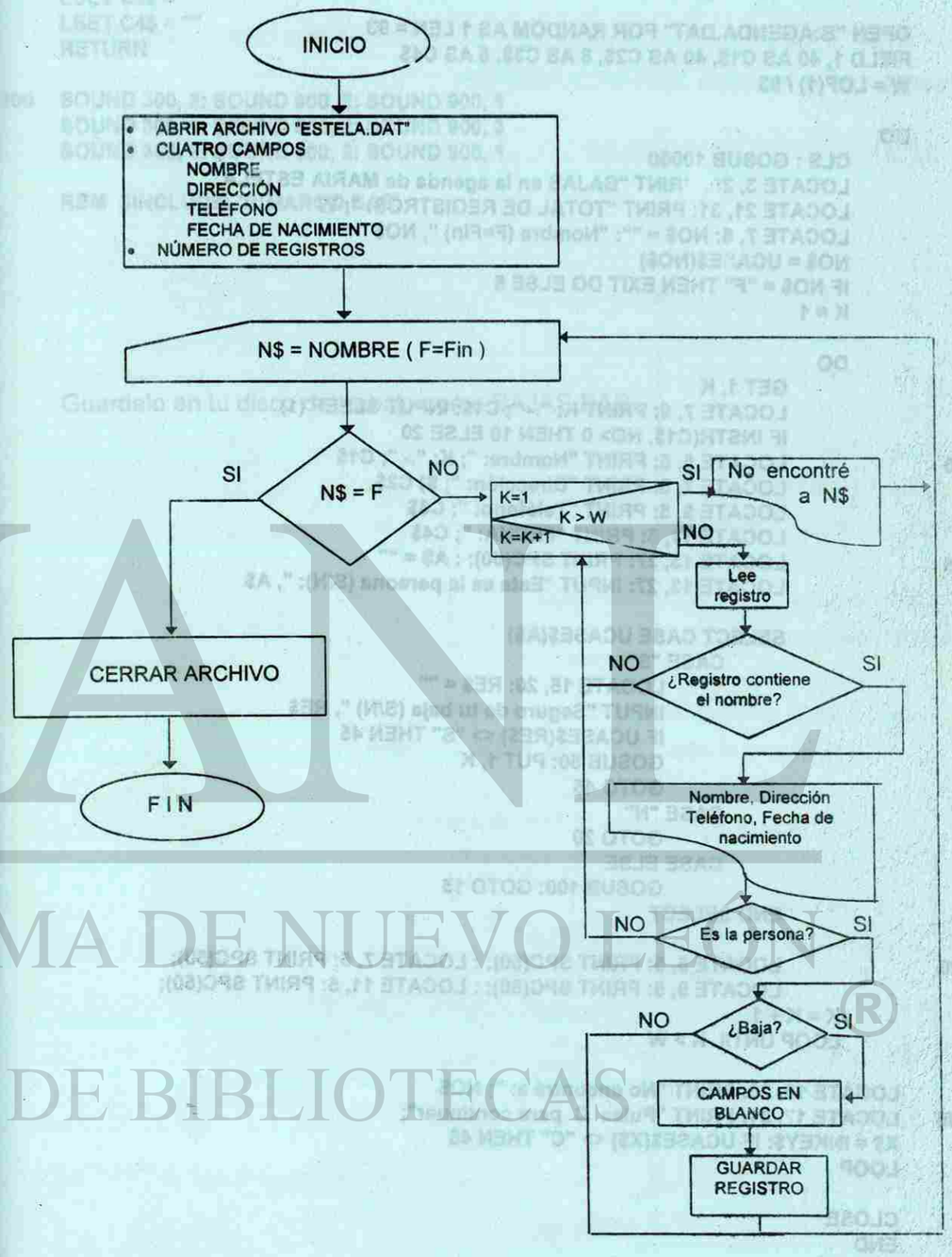
B.- PROGRAMA PARA BAJAS.

El programa de bajas nos va a permitir borrar definitivamente algún registro de nuestra AGENDA.

ALGORITMO PARA BAJAS

1.- Inicio
2.- Abrir archivo
3.- Determinar los campos
4.- Cantidad de registros = W
5.- Pedir nombre
6.- Checar dato para salir
    SI es salida ir al paso 19, **NO** ir a 8
7.- Incremetar el contador
8.- Contador mayor que W
    SI ir al paso 9, **NO** ir al paso 11
9.- Imprime "no encontré a " NO$
10.- Ir al paso 5
11.- Leer el registro del disco
12.- Revisar si el registro contiene el nombre
    SI contiene el nombre ir a 13, **NO** lo contiene ir a 7
13.- Presentar los datos.
14.- Preguntar si es la persona que se busca.
    SI es la persona ir a 15, **NO** es ir a 7
15.- Asegurar si es baja (S/N)
    SI es BAJA ir a 16, **NO** es ir a 5
16.- Borrar los campos
17.- Guardar el registo
18.- Ir a 5
19.- Cerrar el archivo
20.- Fin

DIAGRAMA DE FLUJO PARA BAJAS



CODIFICACIÓN QB PARA BAJAS

```
OPEN "B:AGENDA.DAT" FOR RANDOM AS 1 LEN = 93
FIELD 1, 40 AS C1$, 40 AS C2$, 8 AS C3$, 5 AS C4$
W = LOF(1) / 93

DO
    CLS : GOSUB 10000
    LOCATE 3, 2[illegible]RINT "BAJAS en la agenda de MARIA ESTELA."
    LOCATE 21, 31: PRINT "TOTAL DE REGISTROS: "; W
    LOCATE 7, 6: NO$ = "": "Nombre (F=Fin) ", NO$
    NO$ = UCASE$(NO$)
    IF NO$ = "F" THEN EXIT DO ELSE 5
5   K = 1

    DO
        GET 1, K
        LOCATE 7, 6: PRINT K; ".- "; C1$: INPUT SLEEP (1)
        IF INSTR(C1$, NO> 0 THEN 10 ELSE 20
10      LOCATE 5, 5: PRINT "Nombre: "; K; ".- "; C1$
        LOCATE 7, 5: PRINT "Dirección: "; $) C2$
        LOCATE 9, 5: PRINT "Teléfono: "; C3$
        LOCATE 11, 5: PRINT "FECHA: "; C4$
15      LOCATE 13, 27: PRINT SPC(50); : A$ = ""
        LOCATE 13, 27: INPUT "Esta es la persona (S/N): ", A$

        SELECT CASE UCASE$(A$)
            CASE "S"
                LOCATE 15, 20: RE$ = ""
                INPUT "Seguro de tu baja (S/N) ", RE$
                IF UCASE$(RE$) <> "S" THEN 45
                GOSUB 50: PUT 1, K
                GOTO 45
            CASE "N"
                GOTO 20
            CASE ELSE
                GOSUB 100: GOTO 15
        END SELECT

20      LOCATE 5, 5: PRINT SPC(50); : LOCATE 7, 5: PRINT SPC(50);
        LOCATE 9, 5: PRINT SPC(50); : LOCATE 11, 5: PRINT SPC(50);
    K = K + 1
    LOOP UNTIL K > W

LOCATE 10, 20: PRINT "No encontré a: "; NO$
45  LOCATE 17, 31: PRINT "Pulsa  C  para continuar";
X$ = INKEY$: IF UCASE$(X$) <> "C" THEN 45
LOOP

CLOSE
END
```



```
50  LSET C1$ = ""
    LSET C2$ = ""
    LSET C3$ = ""
    LSET C4$ = ""
    RETURN

100 SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    SOUND 300, 1: SOUND 600, 2: SOUND 900, 3
    SOUND 300, 3: SOUND 600, 2: SOUND 900, 1

    REM  $INCLUDE: 'B:MARCO.BAS'
```

Guardalo en tu disco de trabajo como BAJAS.BAS

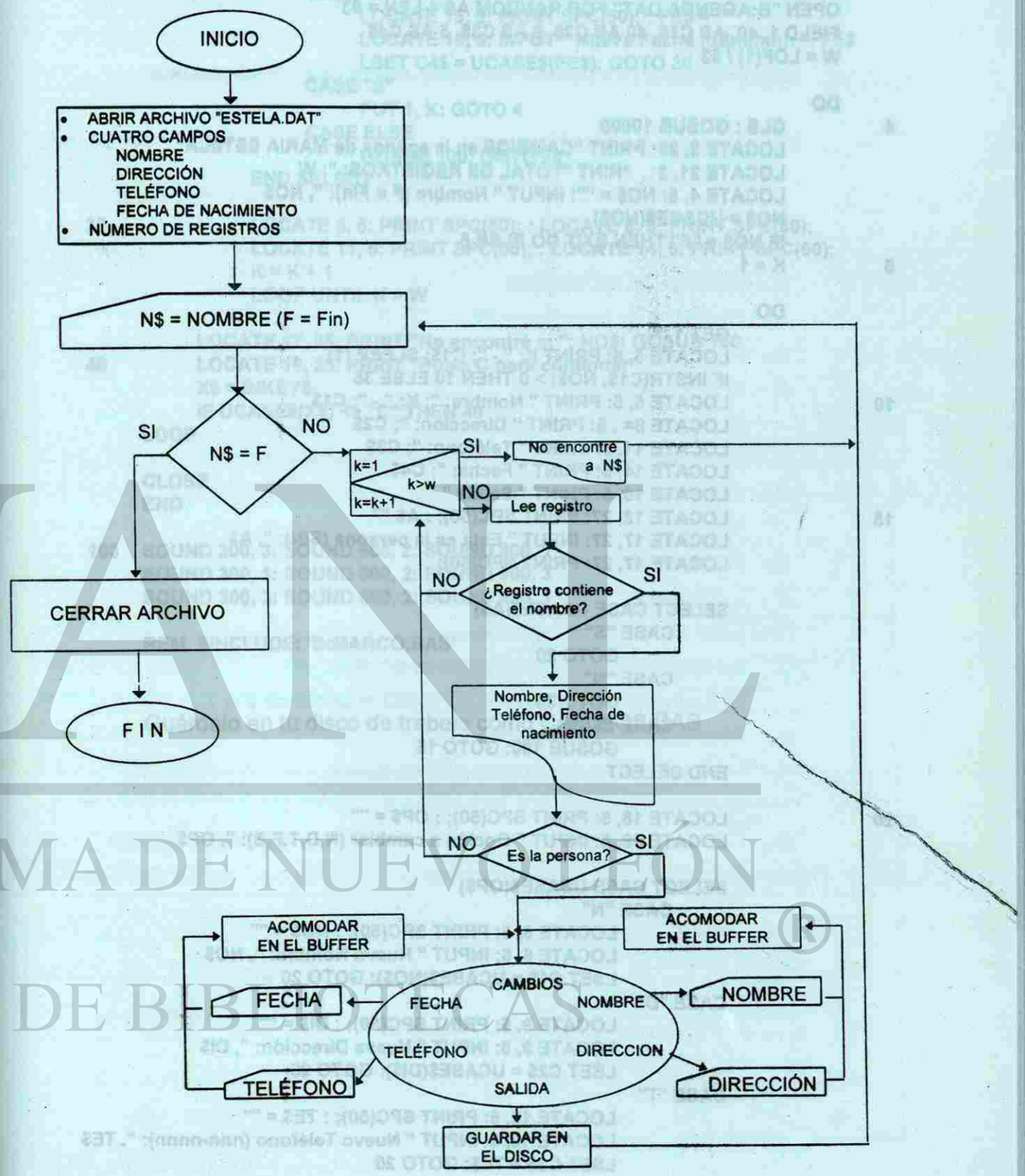
C.- PROGRAMA PARA CAMBIOS.

El programa para cambios nos permite modificar algún campo de cualquiera de los registros de nuestra AGENDA.

ALGORITMO PARA CAMBIOS

1.- Inicio
2.- Abrir archivo
3.- Determinar los campos
4.- Cantidad de registros = W
5.- Pedir nombre
6.- Checar dato para salir
    SI es salida ir al paso 21, **NO** ir a 8
7.- Incremetar el contador
8.- Contador mayor que W
    SI ir al paso 9, **NO** ir al paso 11
9.- Imprime "no encontré a " NO$
10.- Ir al paso 5
11.- Leer del disco el registro
12.- Revisar si el registro contiene el nombre
    SI contiene el nombre ir a 13, **NO** lo contiene ir a 7
13.- Presentar todos los datos.
14.- Preguntar si es la persona que se busca.
    SI es la persona ir a 15, **NO** es ir a 7
15.- Pedir dato a cambiar
    SI es salida ir a 19, **NO** ir a 16
16.- Pedir nuevo dato
17.- Acomodar el dato en el buffer
18.- Ir a 15
19.- Guardar datos en el disco
20.- Ir al paso 5
21.- Cerrar archivo.
22.- Fin.

DIAGRAMA DE FLUJO PARA CAMBIOS



CODIFICACIÓN QB PARA CAMBIOS

```
    OPEN "B:AGENDA.DAT" FOR RANDOM AS 1 LEN = 93
    FIELD 1, 40 AS C1$, 40 AS C2$, 8 AS C3$, 5 AS C4$
    W = LOF(1) / 93
    DO
4       CLS : GOSUB 10000
        LOCATE 2, 28: PRINT "CAMBIOS en la agenda de MARIA ESTELA"
        LOCATE 21, 3: PRINT "TOTAL DE REGISTROS: "; W
        LOCATE 4, 5: NO$ = "": INPUT " Nombre (F = Fin): ", NO$
        NO$ = UCASE$(NO$)
        IF NO$ = "F" THEN EXIT DO ELSE 5
5       K = 1

        DO
            GET 1, K
            LOCATE 5, 6: PRINT K; ".- "; C1$: SLEEP (1)
            IF INSTR(C1$, NO$) > 0 THEN 10 ELSE 35
10          LOCATE 5, 5: PRINT " Nombre: "; K; ".- "; C1$
            LOCATE 8= , 5: PRINT " Dirección: "; C2$
            LOCATE 11, 5: PRINT " Teléfono: "; C3$
            LOCATE 14, 5: PRINT " Fecha: "; C4$
            LOCATE 16, 5: PRINT " Salida "
15          LOCATE 18, 27: PRINT SPC(50); : A$ ""
            LOCATE 17, 27: INPUT " Esta es la persona (S/N): ", A$
            LOCATE 17, 27: PRINT SPC(50);

            SELECT CASE UCASE$(A$)
                CASE "S"
                    GOTO 20
                CASE "N"
                    GOTO 35
                CASE ELSE
                    GOSUB 100: GOTO 15
            END SELECT

20          LOCATE 18, 5: PRINT SPC(50); : OP$ = ""
            LOCATE 18, 5: INPUT " Opción a cambiar (N,D,T,F,S): ", OP$

            SELECT CASE UCASE$(OP$)
                CASE "N"
                    LOCATE 6, 5: PRINT SPC(50); : NO$ = ""
                    LOCATE 6, 5: INPUT " Nuevo Nombre: ", NO$
                    LSET C1$ = UCASE$(NO$): GOTO 20
                CASE "D"
                    LOCATE 9, 5: PRINT SPC(50); : DI$ = ""
                    LOCATE 9, 5: INPUT " Nueva Dirección: ", DI$
                    LSET C2$ = UCASE$(DI$): GOTO 20
                CASE "T"
                    LOCATE 12, 5: PRINT SPC(50); : TE$ = ""
                    LOCATE 12, 5: INPUT " Nuevo Teléfono (nnn-nnnn): ", TE$
                    LSET C3$ = TE$: GOTO 20
```



```
                CASE "F"
                    LOCATE 15, 5: PRINT SPC(50); : FE$ = ""
                    LOCATE 15, 5: INPUT " Nueva Fecha (ddmmm): ", FE$
                    LSET C4$ = UCASE$(FE$): GOTO 20
                CASE "S"
                    PUT 1, K: GOTO 4
                CASE ELSE
                    GOSUB 100: GOTO 20
            END SELECT

35          LOCATE 5, 5: PRINT SPC(50); : LOCATE 8, 5: PRINT SPC(50);
            LOCATE 11, 5: PRINT SPC(50); : LOCATE 14, 5: PRINT SPC(50);
            K = K + 1
            LOOP UNTIL K > W

        LOCATE 17, 25: PRINT "No encontré a: "; NO$: GOSUB 100
40      LOCATE 18, 25: PRINT "Pulsa C para continuar"
        X$ = INKEY$
        IF UCASE$(X$) <> "C" THEN 40
    LOOP

    CLOSE
    END

100 SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    SOUND 300, 1: SOUND 600, 2: SOUND 900, 3
    SOUND 300, 3: SOUND 600, 2: SOUND 900, 1

    REM $INCLUDE: 'B:MARCO.BAS'
```

Guárdalo en tu disco de trabajo como CAMBIOS.BAS

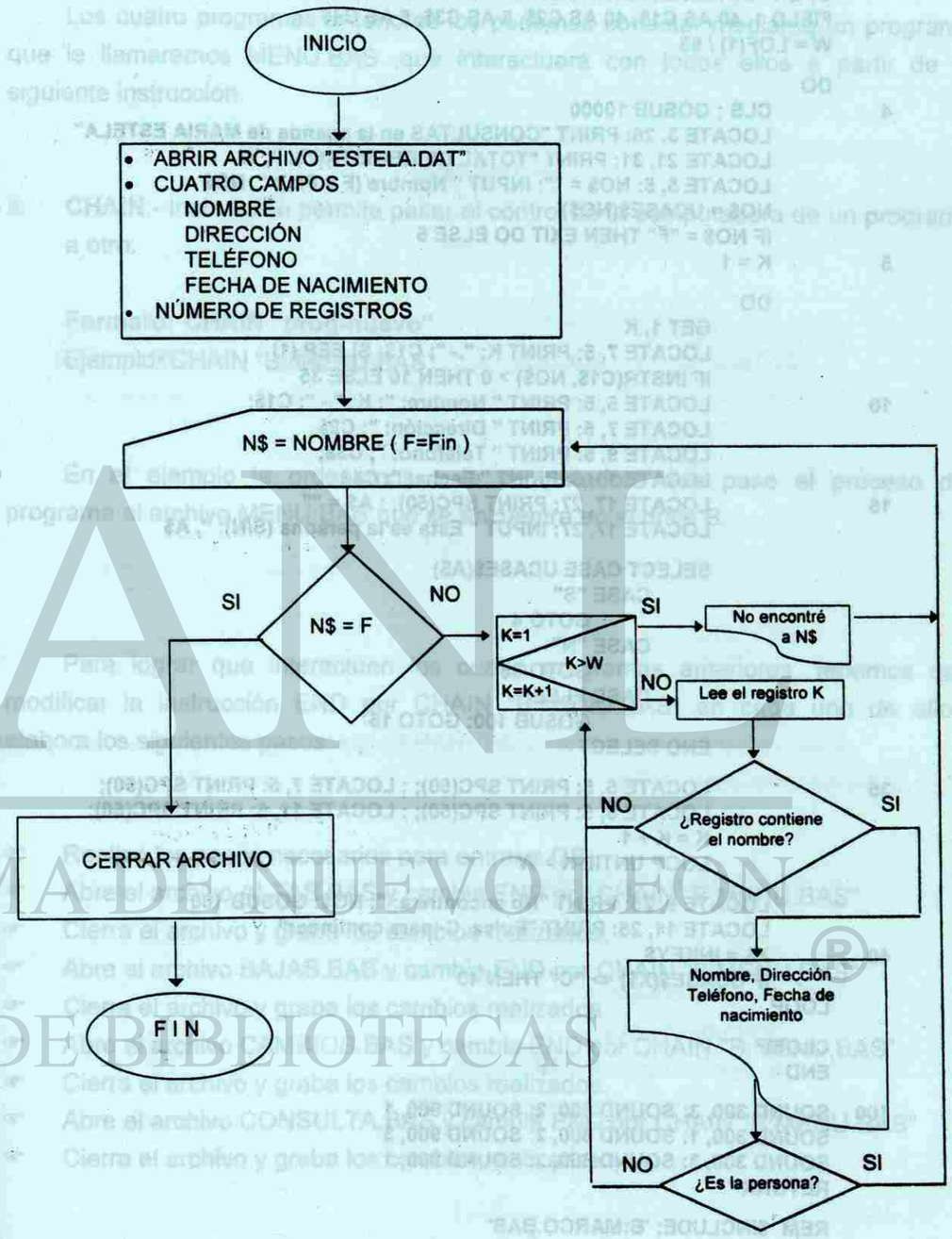
D.- PROGRAMA PARA CONSULTA.

Este programa nos permite consultar los datos de una persona.

ALGORITMO PARA CONSULTA

1.- Inicio
2.- Abrir archivo
3.- Determinar los campos
4.- Cantidad de registros = W
5.- Pedir nombre
6.- Checar dato para salir
SI es salida ir al paso 15, **NO** ir a 7
7.- Incremetar el contador
8.- Contador mayor que W
SI ir al paso 9, **NO** ir al paso 11
9.- Imprime "no encontré a " NO$
10.- Ir al paso 5
11.- Leer del disco el registro
12.- Revisar si el registro contiene el nombre
SI contiene el nombre ir a 13 si **NO** lo contiene ir a 7
13.- Presentar todos los datos.
14.- Preguntar si es la persona que se busca.
SI es la persona ir a 5, **NO** es ir a 7
15.- Cerrar archivo.
16.- Fin.

DIAGRAMA DE FLUJO PARA CONSULTA



CODIFICACIÓN QB PARA CONSULTAS

```
OPEN "B:AGENDA.DAT" FOR RANDOM AS 1 LEN = 93
FIELD 1, 40 AS C1$, 40 AS C2$, 8 AS C3$, 5 AS C4$
W = LOF(1) / 93

DO
4       CLS : GOSUB 10000
        LOCATE 3, 26: PRINT "CONSULTAS en la agenda de MARIA ESTELA"
        LOCATE 21, 31: PRINT "TOTAL DE REGISTROS: "; W
        LOCATE 5, 5: NO$ = "": INPUT " Nombre (F = Fin): ", NO$
        NO$ = UCASE$(NO$)
        IF NO$ = "F" THEN EXIT DO ELSE 5
5       K = 1

        DO
                GET 1, K
                LOCATE 7, 5: PRINT K; ".- "; C1$: SLEEP (1)
                IF INSTR(C1$, NO$) > 0 THEN 10 ELSE 35
10              LOCATE 5, 5: PRINT " Nombre: "; K; ".- "; C1$;
                LOCATE 7, 5: PRINT " Dirección: "; C2$
                LOCATE 9, 5: PRINT " Teléfono: "; C3$;
                LOCATE 11, 5: PRINT " Fecha: "; C4$
15              LOCATE 17, 27: PRINT SPC(50); : A$ = ""
                LOCATE 17, 27: INPUT " Esta es la persona (S/N): ", A$

                SELECT CASE UCASE$(A$)
                        CASE "S"
                                GOTO 4
                        CASE "N"
                                GOTO 35
                        CASE ELSE
                                GOSUB 100: GOTO 15
                END SELECT

35              LOCATE 5, 5: PRINT SPC(50); : LOCATE 7, 5: PRINT SPC(50);
                LOCATE 9, 5: PRINT SPC(50); : LOCATE 11, 5: PRINT SPC(50);
                K = K + 1
                LOOP UNTIL K > W

        LOCATE 9, 25: PRINT "No encontré a: "; NO$: GOSUB 100
        LOCATE 11, 25: PRINT "Pulsa  C  para continuar"
40      X$ = INKEY$
        IF UCASE$(X$) <> "C" THEN 40
LOOP

CLOSE
END

100 SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    SOUND 300, 1: SOUND 600, 2: SOUND 900, 3
    SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    RETURN

    REM  $INCLUDE: 'B:MARCO.BAS'
```

Guárdalo en tu disco de trabajo como CONSULTA.BAS

E.- PROGRAMA PARA MENÚ.

Los cuatro programas anteriores los podemos conectar mediante un programa que le llamaremos MENU.BAS ,que interactuará con todos ellos a partir de la siguiente instrucción.

- **CHAIN**.- Instrucción permite pasar el control de la computadora de un programa a otro.

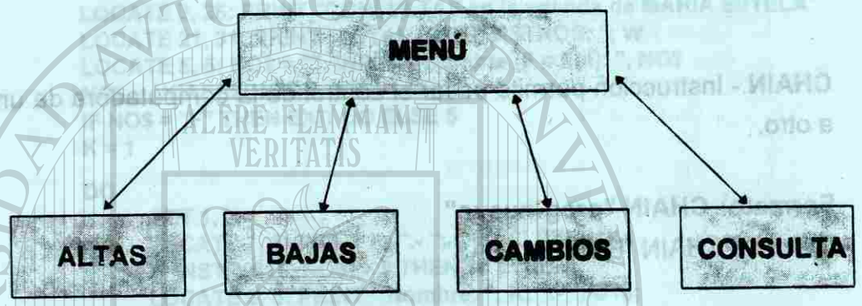
Formato: CHAIN "prog-nuevo"
Ejemplo: CHAIN "B:MENU.BAS"

En el ejemplo le ordenamos a la computadora que pase el proceso del programa al archivo MENU:BAS que se encuentra en el drive B.

Para lograr que interactúen los cuatro programas anteriores, tenemos que modificar la instrucción END por CHAIN "B:MENU.BAS" en cada uno de ellos; elabora los siguientes pasos:

- Realiza los pasos necesarios para entrar a QB.
- Abre el archivo ALTAS.BAS y cambia END por CHAIN "B:MENU.BAS"
- Cierra el archivo y graba los cambios realizados.
- Abre el archivo BAJAS.BAS y cambia END por CHAIN "B:MENU.BAS"
- Cierra el archivo y graba los cambios realizados.
- Abre el archivo CAMBIOS.BAS y cambia END por CHAIN "B:MENU.BAS"
- Cierra el archivo y graba los cambios realizados.
- Abre el archivo CONSULTA.BAS y cambia END por CHAIN "B:MENU.BAS"
- Cierra el archivo y graba los cambios realizados.

La representación gráfica de los programas enlazados quedaría de la siguiente manera.

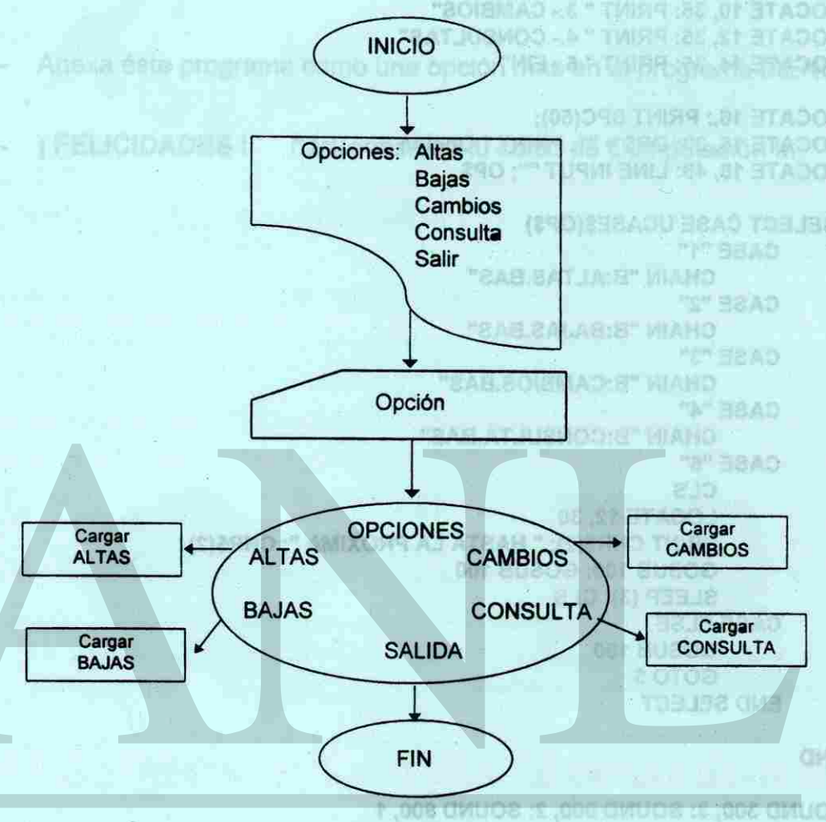


Elaboraremos el programa **MENU.BAS** para lograr conectar los programas realizados en esta unidad.

ALGORITMO PARA MENÚ

1.- Inicio
2.- Imprime opciones
3.- Pregunta opción
4.- **SI** opción es fin ir 6, **NO** ir a 5
5.- Carga el programa según opción
6.- Fin.

DIAGRAMA DE FLUJO PARA MENÚ



CODIFICACIÓN QB PARA MENÚ

```
CLS : GOSUB 10000
LOCATE 3, 26: PRINT "MENÚ de la agenda de MARIA ESTELA"
LOCATE 6, 35: PRINT " 1.- ALTAS"
LOCATE 8, 35: PRINT " 2.- BAJAS"
LOCATE 10, 35: PRINT " 3.- CAMBIOS"
LOCATE 12, 35: PRINT " 4.- CONSULTAS"
LOCATE 14, 35: PRINT " 5.- FIN"

5   LOCATE 16,: PRINT SPC(50);
    LOCATE 16, 30: OP$ = "": PRINT " Opción .....[ ] "
    LOCATE 16, 48: LINE INPUT ""; OP$

    SELECT CASE UCASE$(OP$)
        CASE "1"
            CHAIN "B:ALTAS.BAS"
        CASE "2"
            CHAIN "B:BAJAS.BAS"
        CASE "3"
            CHAIN "B:CAMBIOS.BAS"
        CASE "4"
            CHAIN "B:CONSULTA.BAS"
        CASE "5"
            CLS
            LOCATE 12, 30
            PRINT CHR$(2); " HASTA LA PRÓXIMA "; CHR$(2)
            GOSUB 100: GOSUB 100
            SLEEP (3): CLS
        CASE ELSE
            GOSUB 100
            GOTO 5
    END SELECT

    END

100 SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    SOUND 300, 1: SOUND 600, 2: SOUND 900, 3
    SOUND 300, 3: SOUND 600, 2: SOUND 900, 1
    RETURN

    REM $INCLUDE: 'B:MARCO.BAS'
```

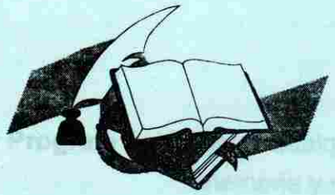
Guárdalo en tu disco de trabajo como MENU.BAS

☺ EJERCICIO GENERAL

1.- Elabora un programa que enliste todos los nombres guardados en el archivo AGENDA.DAT

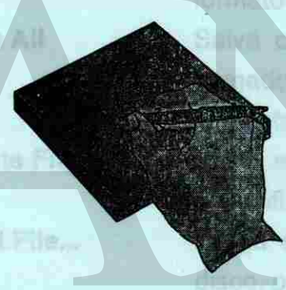
2.- Anexa éste programa como una opción más en el programa MENU.BAS

3.- **¡ FELICIDADES !** has terminado tu curso de Computación II.

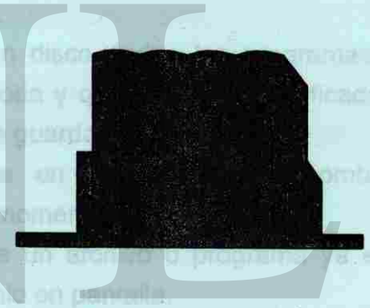


MENÚS

INSTRUCCIONES



ASCII



BIBLIOGRAFÍA

ANEXO

1.- MENÚS EN QuickBASIC

MENÚ FILE

New Program	Borra cualquier programa que se encuentre actualmente en memoria y nos permite crear uno nuevo.
Open Program...	Abre en memoria un programa grabado en un disco.
Merge...	Con esta opción podemos pegar archivos al insertarlos en el módulo actual.
Save. . .	Salva o guarda en el disco el archivo que se encuentra en memoria con el nombre dado anteriormente.
Save As...	Salva o guarda en disco el archivo que se encuentra en memoria permitiéndonos modificar el nombre, el drive y/o el formato del mismo.
Save All	Salva o guarda en disco, todos los programas que están cargados en memoria y que han sido modificados desde la vez anterior que se guardaron.
Create File...	Crea en memoria un archivo con el nombre que se especifica en ese momento.
Load File...	Carga en memoria un archivo o programa ya existente en disco, presentándolo en pantalla.
Unload file...	Borra de la memoria un archivo o programa; no tiene efecto sobre el disco.
Print...	Manda imprimir el texto seleccionado, el texto de la ventana activa, el módulo actual asociado con la ventana activa o todos los módulos que componen el programa actual.
DOS Shell	Sale temporalmente de la pantalla de QB y nos permite trabajar en comandos y órdenes del DOS mientras QB está en memoria. Para regresar a la pantalla de QB en que nos salimos, sólo tecleamos la palabra EXIT y **Enter**
Exit	Salida permanente de QuickBASIC al DOS, borrando de memoria el QB. Si en ese momento hay un programa en memoria, al que se le haya hecho una modificación, nos pregunta si queremos salvarlo.

MENÚ EDIT

Undo Restaura o devuelve el contenido de la línea donde se encuentra el cursor, deshaciendo todos los cambios y correcciones que se hayan hecho en la línea, antes de dar **Enter**.

Cut Recorta un conjunto de caracteres, palabras o líneas seleccionadas previamente y las sitúa en la memoria intermedia.

Copy Copia un conjunto de caracteres, palabras o líneas seleccionadas previamente y sitúa en la memoria intermedia (**Clipboard**), sin alterar el texto original (**Ctrl+Ins**).

Paste Pega o agrega el conjunto de caracteres que está en la memoria intermedia (**Clipboard**), situándolos donde se encuentra el cursor (**Shift+Ins**).

New SUB... Nos permite crear una nueva subrutina o procedimiento dentro de nuestro módulo actual.

New FUNCTION... Mediante este comando podemos programar una nueva función dentro de nuestro módulo de programación.

MENÚ VIEW

SUBs... Despliega una caja de diálogos que contiene, además, una lista de todos los módulos del programa actual, así como subprogramas de procedimientos y funciones contenidos en cada módulo.

Next SUB Muestra el siguiente procedimiento del módulo actual en la ventana activa.

Split Divide la pantalla en dos ventanas; para regresar a una sola ventana se da la misma orden:Split. (F6 para cambiar de ventana).

Next Statement Al suspender la ejecución, el cursor se posiciona en la orden siguiente. Esta orden es muy útil en la depuración de los programas.

Output Screen Presenta la ventana de salida de datos en un programa y retorna a la pantalla de edición tecleando la misma orden o con F4.

Include File Presenta el archivo de inclusión, dando la oportunidad de reeditarlo y poder grabar una nueva versión.

Include Lines Presenta el archivo de inclusión (INCLUDE), pero no permite hacer ninguna corrección.

MENÚ SEARCH

Find... Localiza un caracteren , palabra o palabras dentro de un programa.

Selected Text Esta orden permite buscar un texto seleccionado previamente (**Ctrl+**).

Repeat Last Find Repite la última búsqueda, cuantas veces se desee, mediante la tecla F3.

Change ... Nos permite seleccionar un texto y cambiarlo por otro.

Label... Esta orden se utiliza para buscar etiquetas. Al teclear la palabra a buscar, añade por default : .

MENÚ RUN

Start Ejecuta el programa que está actualmente en memoria (**Shift+F5**).

Restart Se utiliza en el caso de un programa que durante su ejecución es interrumpido y queremos volver a correrlo desde el inicio.

Continue Reanuda la ejecución de un programa la siguiente instrucción cuando éste se ha interrumpido (F5).

Modify COMMAND$... Establece la cadena devuelta por la función **COMMAND$.**

Make EXE File... Crea un archivo ejecutable en disco. El archivo tiene que haberse guardado antes con la opción Save del menú Files (**Alt+F+S**).

Make Library... Crea una biblioteca(library) Quick en el disco.

Set Main Module... Cambia o establece el módulo principal de entre todos los que componen el programa.

MENÚ DEBUG

Add Watch.. Añade una marca en la expresión que se desea observar en la ventana Watch, a la hora de ejecutar el programa, es decir, los valores correspondientes serán visualizados en la parte superior de la pantalla.

Instant Watch... Visualiza el valor de una variable, de una expresión aritmética o de una expresión de relación (falso o verdadero), para ello se debe posicionar el cursor sobre la variable.

Watchpoint... Permite introducir en un programa puntos de parada condicionales, añadir expresiones o variables que detendrán la ejecución del programa, cuando al evaluarlos den un resultado diferente de 0.

Delete Watch... Borra la marca o punto en una expresión que ya no se desea observar, dad con Add Watch o Watch Point.

Delete All Watch Borra todas las marcas o puntos de las expresiones que se han estado observando.

Trace On Rutina para ir observando el programa durante toda su ejecución; nos presenta en pantalla el programa e ilumina más intensamente la orden que se va ejecutando.

History ON Al cometer un error en el programa, podemos repetir paso a paso las ultimas veinte instrucciones mediante **SHIFT+F8**, o seguir paso a paso las siguientes veinte instrucciones mediante **SHIFT+F10.**

Toggle Breackpoint Esta orden permite poner o quitar una pausa. Para hacerlo, se coloca el cursor en la línea y se pulsa F9.

Clear All Breackpoint Borra todas las marcas de los puntos de interrupción.

Break on Error Se utiliza cuando en un programa incluimos sentencias ON ERROR Break on Errors; pone implícitamente un punto de parada después de la etiqueta de la rutina de manejo de errores.

Set Next Statement Sitúa el cursor en la siguiente sentencia a ejecutarse, después del punto de interrupción (Breackpoint), el efecto es similar al GOTO.

MENÚ CALLS

MENÚ OPTIONS

Display... Para modificar el entorno visual de la pantalla.

Set Paths... Establece ruta de acceso.

Right Mouse... Permite cambiar el efecto del botón derecho del mouse.

Syntax Checking Esta es la función que verifica cada renglón del programa para ver que esté bien escrito; si hay algún error manda un mensaje y el cursor se posiciona en la palabra mal escrita. Esta función se activa o desactiva con la misma orden. El símbolo (•) nos indica que está activada.

Full Menus Activa o desactiva los menús completos (Full Menus); si está marcado con un punto está activo; si no tiene marca es que se está trabajando con los menús simples (Easy Menus).

MENÚ HELP

Index	Proporciona ayuda sobre palabras reservadas (Comandos), las cuales aparecen en forma alfabética.
Contents	Despliega los nombres de los temas donde existe alguna información de ayuda.
Topic:	Proporciona ayuda sobre la palabra clave en que está situado el cursor.
Help	Proporciona información sobre como usar el menú Help.

2.- INSTRUCCIONES UTILIZADAS EN QuickBASIC

- **CLS**.- Limpia pantalla.
- **LET**.- Indica una operación arimética o asigna u valor determinado a una variable.
- **PRINT**.- Imprime el contenido de las variables y/o expresiones en el monitor.
- **LPRINT**.- Imprime el contenido de las variables y/o expresiones en la impresora.
- **INPUT.-** Acepta datos de entrada por el teclado.
- **END**.- Termina la ejecución de un programa.
- **LOCATE**.- Permite posicionar el cursor en lugar determinado de la pantalla, sobre un renglón, el cual debe de estar en el rango de 1 al 24 y una columna especificada, la cual debe de estar en el rango de 1 al 80.
- **DEF FN**.- Con esta instrucción podemos nombrar y definir una función o una fórmula y así obtener el resultado de ella.
- **SPC**.- Instrucción que sirve para especificar **n** cantidad de espacios en blanco.
- **SQR.-** Es una función numérica que proporciona un valor igual a la raíz cuadrada de una expresión aritmética, equivalente a obtener la potencia 0.5 de la base.
- **FUNCIONES TRIGONOMETRICAS**.- Las angulos que se utilizan an las funciones trigonométricas que se trabaja QB, deben de estar en radianes, por lo tanto si se realiza un programa y los datos se proporcionan en grados, se tendrá que hacer la conversión de grados a radianes con la siguiente fórmula: R = G (3.1416)/180

- **SELECT CASE**.- Permite establecer dos o más opciones en un mismo programa.

- **INKEY$.-** Efectúa la lectura de un solo caracter del teclado al momento de pulsarla y no se muestra en la pantalla como lo hace la instrucción INPUT.

- **UCASE$**.- Convierte todas las letras de una expresión en mayúsculas.

- **LCASE$.-** Convierte todas las letras de una expresión en minúsculas.

- **SOUND**.- Da un sonido en una frecuencia desde 37 hasta 32767 Hertz y una duración en la que va a permanecer el sonido durante el tiempo que nosotros indiquemos, cada unidad de duración equivale a 1/18 de segundo.

- **RND.-** Sirve para semejar números aleatorios de 0 a 0.999999, con RANDOMIZE TIMER la función RND obtendrá un valor de semilla diferente cada vez que se ejecute, pero dependiendo de la escala de números que se quiera trabajar, se hacen los arreglos necesarios.

- **INT**.- Sirve para eliminar la parte decimal del número, dejando sólo la parte entera.

- **GOSUB**.- Se utiliza en programas cuando se tienen operaciones repetitivas; se puede crear una subrutina en forma interna o externa del programa principal. Al final de la subrutina se concluye con RETURN para que regrese a la línea del programa después del GOSUB; cuando una subrutina es grabada como externa es necesario que sea en modo texto.

- **DO / LOOP**.- Esta instrucción se le considera como bucle infinito, para salir de él es necesario que en una parte del ciclo se utilice EXIT DO.

- **DO / UNTIL**.- El bucle se ejecuta mientras la condición sea falsa, es decir, HASTA que la condición sea verdadera. La condición puede ir al principio o al final del bucle. Si la condición va al principio se utiliza la opción DO UNTIL / LOOP y si la condición va al final se convierte en DO / LOOP UNTIL.

- **DO / WHILE**.- El bucle se ejecuta MIENTRAS la condición sea vedadera, es decir, hasta que la condición sea falsa. Permite tener la condición al principio o al final del bucle, cuando la condición va al principio del bucle, la instrucción es DO WHILE / LOOP y si la condición va al final se convierte en DO / LOOP WHILE.

- **SLEEP**.- Sirve pa... retardar el tiempo en QB.

- **ASC.-** Convierte la primera letra o caracter de una cadena en su valor numérico correspondiente al código ASCII.

- **CHR$.-** Es la función inversa de ASC, permite generar caracteres alfanuméricos correspondientes a un código dado de 0 a 255.

- **LEN**.- Proporciona la longitud de una expresión cadena, que va de 0 hasta 32,767 caracteres.

- **LEFT$**.- Es una función subcadena que proporciona los n primeros caracteres de una expresión cadena, contando desde la izquierda.

- **RIGHT$**.- Devuelve n caracteres que se encuentran a la derecha de la cadena.

- **MID$**.- Es una función subcadena que extrae una parte del interior de una expresión alfanumérica; n es el primer caracter que va a tomar a partir de la izquierda; m es el primer caracter que va a tomar a partir de la izquierda.

- **INSTR.-** Localiza un string dentro de otro y si lo encuentra, regresa al lugar que ocupa en el string secundario, si no lo encuentra regresa un cero (0), r es el lugar donde se iniciará la búsqueda, si se omite se toma como uno.

- **LINE INPUT.-** Instrucción para recibir datos, en cual se puede incluir la coma (,); la inclusión de la coma no es posible con el INPUT

- **$INCLUDE**.- Sirve para incluir un programa o subrutinas que están fuera de un programa, es decir, programas o subrutinas externas

- **DIM**.- Esta instrucción permite fijar el tamaño de la memoria que hay que reservar.
- **OPEN**.- Esta instrucción sirve para abrir un archivo de datos ya existente o crear uno nuevo.
- **FIELD**.- Esta instrucción sirve para especificar cuántos campos tiene cada registro, cuales son y de qué longitud es cada uno de ellos.
- **LOF**.- Esta instrucción indica la cantidad de bytes que se han utilizado en el disco para el archivo de datos que abrimos.
- **LSET - RSET**.- Instrucciones para acomodar la información en cada campo del buffer, justificados a la derecha o a la izquierda.
- **PUT**.- Instrucción para guardar en el disco, los datos del buffer del registro.
- **GET**.- Instrucción para leer y traer un registro especifico, coloca en la memoria temporal (buffer) toda la información del registro, para poder utilizarla o manipularla según nuestras necesidades.
- **CLOSE**.- Esta instrucción sirve para cerrar nuestros archivos; es muy conveniente cerrarlo, pues si se nos olvida hacerlo puede perderse información o llenarse de basura.
- **CHAIN**.- Instrucción permite pasar el control de la computadora de un programa a otro.

3.- CODIGOS DE CARACTERES DE ASCII

Valor ASCII	Caracter	Valor ASCII	Caracter	Valor ASCII	Caracter
1	☺	31	Cursor abajo	61	=
2	☻	32	Espacio	62	>
3	♥	33	!	63	?
4	♦	34	"	64	@
5	♣	35	#	65	A
6	♠	36	$	66	B
7	Bip	37	%	67	C
8	◘	38	&	68	D
9	Tabulador	39	'	69	E
10	Avance de línea	40	(	70	F
11	Hogar	41	)	71	G
12	Avance de Form.	42	*	72	H
13	Enter	43	+	73	I
14	♫	44	,	74	J
15	¤	45	-	75	K
16	▻	46	.	76	L
17	◅	47	/	77	M
18	↕	48	0	78	N
19	‼	49	1	79	O
20	¶	50	2	80	P
21	§	51	3	81	Q
22	▬	52	4	82	R
23	↨	53	5	83	S
24	↑	54	6	84	T
25	↓	55	7	85	U
26	→	56	8	86	V
27	←	57	9	87	W
28	Cursor a la derecha	58	:	88	X
29	Cursor a la izquierda	59	;	89	Y
30	Cursor arriba	60	<	90	Z

Valor ASCII	Caracter	Valor ASCII	Caracter	Valor ASCII	Caracter
91	[	121	y	151	ù
92	\	122	z	152	ÿ
93	]	123	{	153	Ö
94	^	124	|	154	Ü
95	_	125	}	155	ø
96	`	126	~	156	£
97	a	127	⌂	157	¥
98	b	128	Ç	158	₧
99	c	129	ü	159	ƒ
100	d	130	é	160	á
101	e	131	â	161	í
102	f	132	ä	162	ó
103	g	133	à	163	ú
104	h	134	å	164	ñ
105	i	135	ç	165	Ñ
106	j	136	ê	166	ª
107	k	137	ë	167	º
108	l	138	è	168	¿
109	m	139	ï	169	⌐
110	n	140	î	170	¬
111	o	141	ì	171	½
112	p	142	Ä	172	¼
113	q	143	Å	173	¡
114	r	144	É	174	«
115	s	145	æ	175	»
116	t	146	Æ	176	▒
117	u	147	ô	177	▒
118	v	148	ö	178	▒
119	w	149	ò	179	│
120	x	150	û	180	┤

Valor ASCII	Caracter	Valor ASCII	Caracter	Valor ASCII	Caracter
181	╡	211	╙	241	±
182	╢	212	╘	242	≥
183	╖	213	╒	243	≤
184	╕	214	╓	244	⌠
185	╣	215	╫	245	⌡
186	║	216	╪	246	÷
187	╗	217	┘	247	≈
188	╝	218	┌	248	°
189	╜	219	█	249	∙
190	╛	220	▄	250	·
191	┐	221	▌	251	√
192	└	222	▐	252	^{n}
193	┴	223	▀	253	^{2}
194	┬	224	α	254	■
195	├	225	ß	255	Espacio en blanco
196	─	226	Γ		
197	┼	227	π		
198	╞	228	Σ		
199	╟	229	σ		
200	╚	230	µ		
201	╔	231	τ		
202	╩	232	Φ		
203	╦	233	θ		
204	╠	234	Ω		
205	═	235	δ		
206	╬	236	∞		
207	╧	237	ø		
208	╨	238	ε		
209	╤	239	∩		
210	╥	240	≡		

BIBLIOGRAFÍA

H. TORNSDORF, M. TORNSDORF. QuickBASIC para principiantes. Marcombo Boixareu Editores. España,1992.

CEBALLOS SIERRA, FCO. JAVIER. Manual para QuickBASIC Guía del programador. Coedición Macrobit Editores S.A. de C.V. -- Ra-Ma Editorial. México,1990.

JOYANES AGULIAR. Metodología de la programación. Mc Graw Hill. Segunda Edición. México.1992.