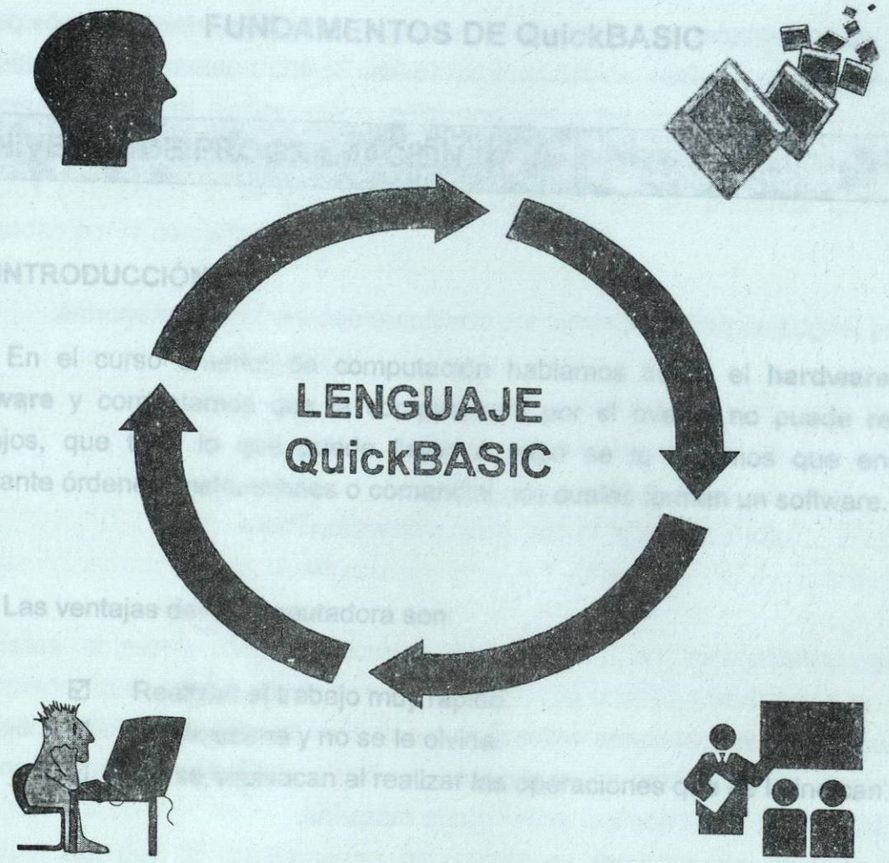
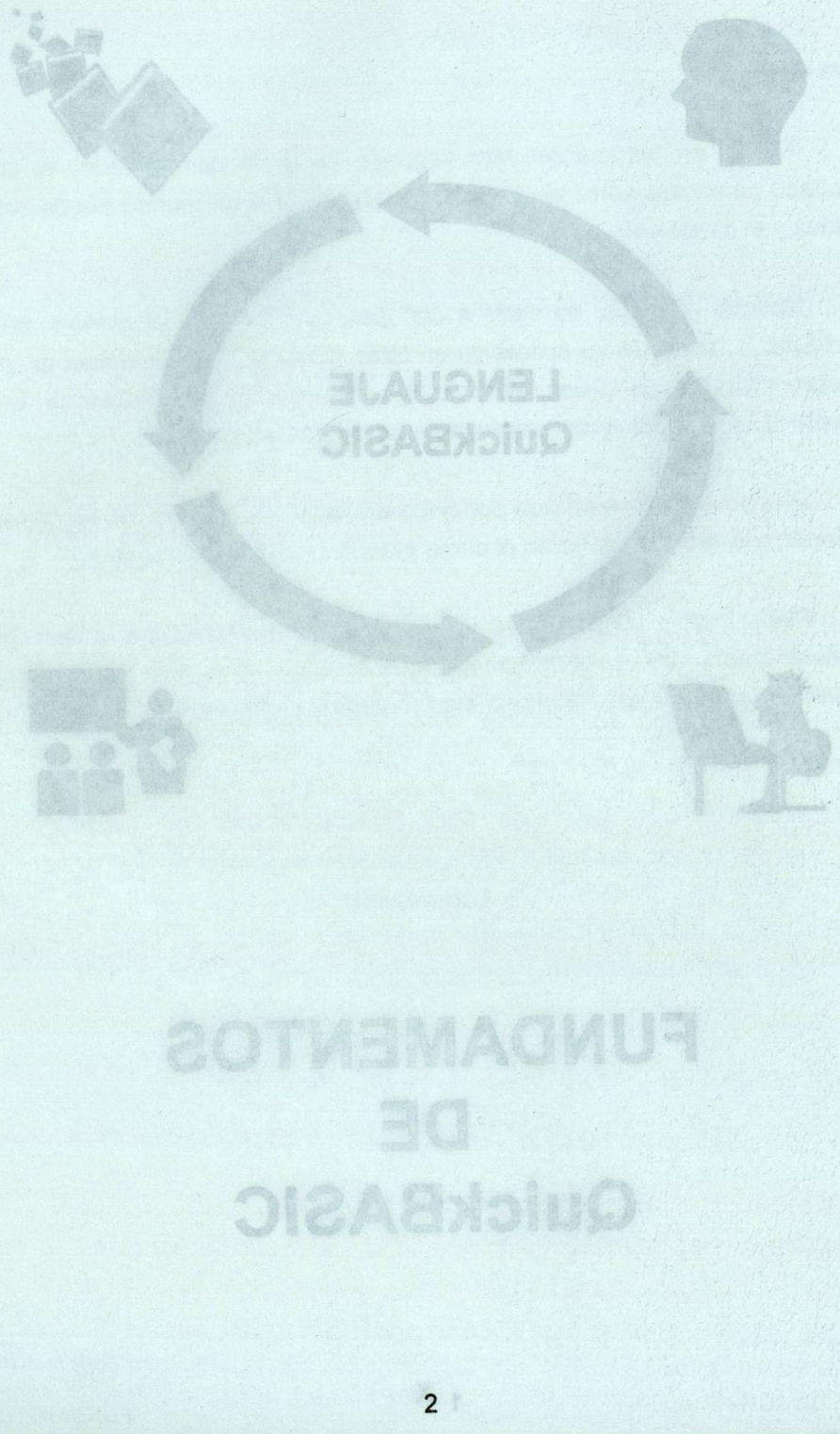


FUNDAMENTOS DE QuickBASIC



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.

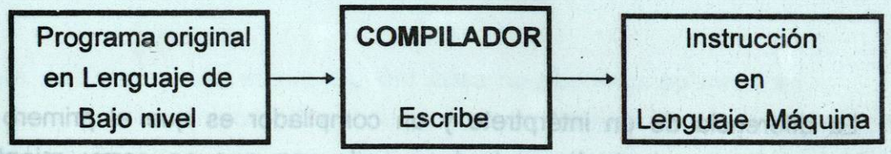
b.- Lenguaje de bajo nivel

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.



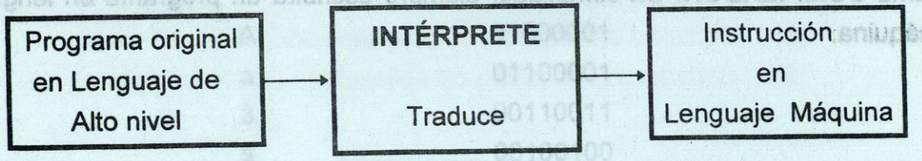
c.- Lenguaje de alto nivel

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"

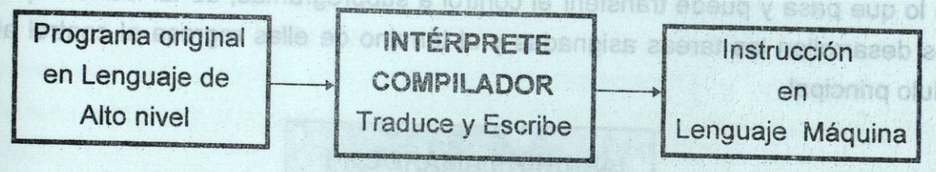
Los programas en lenguaje de alto nivel no los puede interpretar la computadora, por lo que se requiere de un traductor que pueda hacer la conversión a código binario, o sea un compilador, a menos que el lenguaje provenga de un intérprete. Los intérpretes traducen y ejecutan una por una cada instrucción 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.

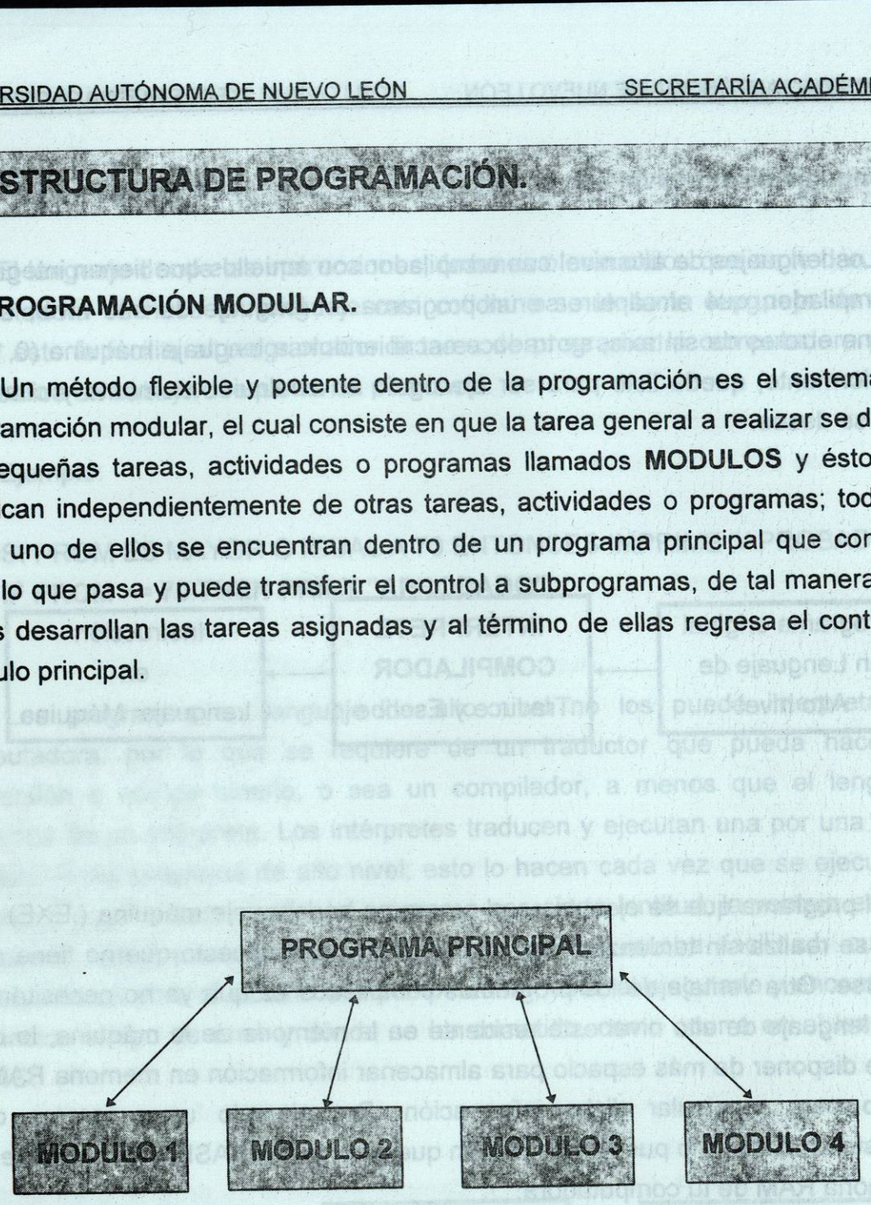


El programa que se ejecuta es el programa en lenguaje máquina (.EXE) por lo que se realiza sin tardanzas, con una rapidez mayor, puesto que no tiene que traducirse. Otra ventaja de los programas compilados es que ya no necesitamos que el lenguaje de alto nivel esté residente en la memoria de la máquina, 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.

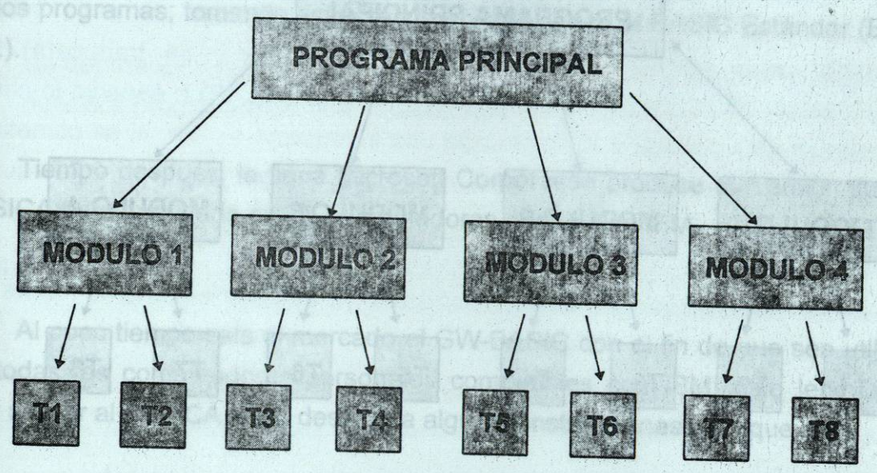
Un método flexible y potente dentro de la programación es el sistema de programación modular, el cual consiste en que la tarea general a realizar se divide en pequeñas tareas, actividades o programas llamados **MODULOS** y éstos se codifican independientemente de otras tareas, actividades o programas; 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.

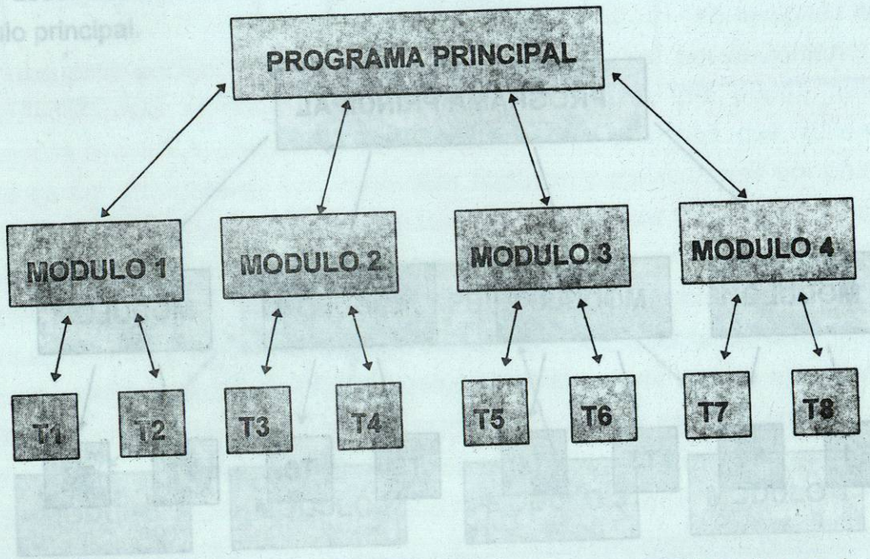
La programación descendente significa que a partir de la tarea general o global, ésta se descomponga en diferentes subtareas o módulos que, en conjunto, resuelvan el problema general y si alguna de estas subtareas o módulos es demasiado compleja, ésta deberá dividirse en otros módulos más simples que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