**LICEO COMPU - MAKET**

**Catedrático:**

Erick Gonzales

**Cátedra:**

Programación ll

**Temas:**

“Informe Teórico de Programación”

**Nombre:**

Brandon Eduardo

Godínez Suret

**Grado:**

5to. BACO “B” J.M.

**Fecha:**

21 de abril de 2017

**ÍNDICE**

[CARÁTULA](#Carátula)……………………………………………………………………………………………….……1

ÍNDICE………………………………………………………………………………………………………2 - 4

[INTRODUCCIÓN](#Introducción)......................................................................................................................5

[CAPITULO I. CONCEPTOS BÁSICOS Y METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DE COMPUTADORAS](#Capitulo1)...................................................................6 [1.1 Introducción](#Introducción)..............................................................................................7

[1.2 Definición de lenguaje](#Definicion)..............................................................................9

[1.3 Definición de algoritmo](#Definiciondeal)............................................................................9

[1.4 Algoritmos cotidianos](#Tiposdeal).............................................................................10

[1.5 Definición de lenguajes algoritmicos](#Definiciondeal)......................................................10

[1.6 Metodología para la solución de problemas por medio de computadora](#Metodo)……………………………………………………………………………………………10

[1.7 Definición del problema](#Defdelpro).........................................................................10

[1.8 Análisis del problema](#Analisisdelpro)..............................................................................10

[1.9 Diseño del algoritmo](#Diseñodeal)...............................................................................11

[1.10 Codificación](#Codifica)..........................................................................................11

[1.11 Prueba y depuración](#Pruebaydep).............................................................................11

[1.12 Documentación](#Document).....................................................................................11

[1.13 Mantenimiento](#Mantenimi).....................................................................................12

[CAPITULO II. ENTIDADES PRIMITIVAS PARA EL DESARROLLO DE ALGORITMOS](#Capitulo2)...............................................................................................13

[2.1 Tipos de datos](#Tiposdeda).........................................................................................14

[2.2 Expresiones](#Expresio).............................................................................................14

[2.3 Operadores y operandos](#Operadoreyopera)........................................................................15

[2.4 Identificadores como localidades de memoria](#Identifi).......................................19

[CAPITULO III. TÉCNICAS DE DISEÑO](#Capitulo3)....................................................................................22

[3.1 Top down](#Topdown)................................................................................................23

[3.2 Bottom up](#Up)...............................................................................................23

[CAPITULO IV. TÉCNICAS PARA LA FORMULACIÓN DE ALGORITMOS](#Capitulo4)..................................24

[4.1 Diagrama de flujo](#Diagramade)....................................................................................25

[4.2 Pseudocodigo](#Pseudo)..........................................................................................26

[4.3 Diagrama estructurado (nassi-schneiderman)](#Diagramasestr)........................................27

[CAPITULO V. ESTRUCTURAS ALGORITMICAS](#Capitulo5).......................................................................28

[5.1 Secuenciales](#Estructusecue)............................................................................................29

[5.2 Condicionales](#Estructucondi)..........................................................................................34

[5.3 Repetición fila condicional](#Estructucicli)......................................................................50

[CAPITULO VI. ARREGLOS](#Capitulo6).....................................................................................................65

[6.1 Vectores](#Vectores)..................................................................................................66

[6.2 Matrices](#Matriz)..................................................................................................68

[CAPITULO VII. MANEJO DE MÓDULOS](#Capitulo7).................................................................................72

[7.1 Definición, Función y Manipulación](#DefiniFunMani)........................................................73

[CAPITULO VIII. VISUAL BASIC](#Capitulo8)……………………………………………………………………………………………76

[8.1 Características de Visual Basic](#Caracteridevisua)……………………………………………………………77

[8.2 Descripción de ventanas principales](#Descripdeventaprinci)………………………………………………….77

[8.3 Las partes del entorno de Visual Basic](#Laspartesdelent)…………………………….…………………78

[8.4 Variable](#Variable)……………………………………………………………………………………………..80

[8.5 Tipos de variables](#Tiposdevari)………………………………………………………………………………81

[8.6 Expresiones lógicas](#Expresioneslogica)……………………………………………………………………………81

[8.7 Operadores de Visual Basic](#Operadoresenvi)………………………………………………………………..82

[8.8 Operadores Racionales](#Operadoresraci)………………………………………………………………………82

[8.9 Sentencias](#Sentencia)…………………………………………………………………………………………83

[8.10 Bucle por contador](#Bucleporconta)………………………………………………………………………….85

[8.11 Bucles por condición](#Bucleporcondi)………………………………………………………………………..85

[8.12 Instrucciones de control de programa](#Intrucciones)……………………………………………..89

[8.13 Teclas de método abreviado](#Teclasde)……………………………………………………………94

[8.14 Eventos del menú](#Eventosdelme)……………………………………………………………………………95

[8.15 MSGBOX](#MSGBOX)………………………………………………………………………………………….95

[8.16 Los controles](#Loscontrol)……………………………………………………………………………………96

[8.17 Controles básicos](#Controlesbasic)…………………………………………………………………………….97

[CONCLUSIONES PERSONALES](#Conclusion)………………….……………………………………………………………….……104

[BIBLIOGRAFIA Y EGRAFÍA](#Bibligografía)...................................................................................................105

**INTRODUCCIÓN**

La programación informática o programación algorítmica, acortada como programación, es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas de computadora. El código fuente es escrito en un lenguaje de programación.

El desarrollo de algoritmos es un tema fundamental en el diseño de programas por lo cual el alumno debe tener buenas bases que le sirvan para poder desarrollar de manera fácil y rápida sus programas.

Estos apuntes servirán de apoyo al catedrático del Liceo Compu-Market, en su labor cotidiana de enseñanza y al estudiante le facilitará desarrollar su capacidad analítica y creadora, para de esta manera mejorar su destreza en la elaboración de algoritmos que sirven como base para la codificación de los diferentes programas que tendrá que desarrollar a lo largo de su carrera.

**CAPITULO I.**

**CONCEPTOS BÁSICOS Y METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DE COMPUTADORAS.**

1.1 Introducción

* De los problemas a los programas
* Breves prácticas de programación

1.2 Definición de lenguaje

1.3 Definición de algoritmo

1.4 Algoritmos cotidianos

1.5 Definición de lenguajes algorítmicos

1.6 Metodología para la solución de problemas por medio de computadora

1.7 Definición del problema

1.8 Análisis del problema

1.9 Diseño del algoritmo

1.10 Codificación

1.11 Prueba y depuración

1.12 Documentación

1.13 Mantenimiento

**Objetivo educacional:**

El alumno:

* Conocerá la terminología relacionada con los algoritmos; así como la importancia de aplicar técnicas adecuadas de programación.
* Conocerá la metodología en cada una de sus etapas.

***1.1 Introducción***

La computadora no solamente es una máquina que puede realizar procesos para darnos resultados, sin que tengamos la noción exacta de las operaciones que realiza para llegar a esos resultados. Con la computadora además de lo anterior también podemos diseñar soluciones a la medida, de problemas específicos que se nos presenten. Más aún, si estos involucran operaciones matemáticas complejas y/o repetitivas, o requieren del manejo de un volumen muy grande de datos.

El diseño de soluciones a la medida de nuestros problemas, requiere como en otras disciplinas una metodología que nos enseñe de manera gradual, la forma de llegar a estas soluciones.

A las soluciones creadas por computadora se les conoce como ***programas*** y no son más que una serie de operaciones que realiza la computadora para llegar a un resultado, con un grupo de datos específicos. Lo anterior nos lleva al razonamiento de que un ***programa*** nos sirve para solucionar un problema específico.

Para poder realizar ***programas*,** además de conocer la metodología mencionada, también debemos de conocer, de manera específica las funciones que puede realizar la computadora y las formas en que se pueden manejar los elementos que hay en la misma.

***Computadora:*** Es un dispositivo electrónico utilizado para procesar información y obtener resultados. Los datos y la información se pueden introducir en la computadora como entrada (input) y a continuación se procesan para producir una salida (output).

***Proceso de información en la computadora***

Datos de Proceso Datos de

entrada salida

***Programa:*** Es el conjunto de instrucciones escritas de algún lenguaje de programación y que ejecutadas secuencialmente resuelven un problema específico.

***Organización física de una computadora***

***CPU***

Unida de Unidad

Dispositivos de Control Arit.-Log. Dispositivos de Entrada Salida

***Memoria***

***Dispositivos de Entrada:*** Como su nombre lo indica, sirven para introducir datos (información) en la computadora para su proceso. Los datos se leen de los dispositivos de entrada y se almacenan en la memoria central o interna. Ejemplos: teclado, scanner (digitalizadores de rastreo), mouse (ratón), trackball (bola de ratón estacionario), joystick (palancas de juego), lápiz óptico.

***Dispositivos de Salida:*** Regresan los datos procesados que sirven de información al usuario. Ejemplo: monitor, impresora.

***La Unidad Central de Procesamiento (C.P.U)*** se divide en dos:

1. Unidad de control
2. Unidad Aritmético - Lógica

***Unidad de Control:*** Coordina las actividades de la computadora y determina que operaciones se deben realizar y en qué orden; así mismo controla todo el proceso de la computadora.

***Unidad Aritmético - Lógica:*** Realiza operaciones aritméticas y lógicas, tales como suma, resta, multiplicación, división y comparaciones.

***La Memoria*** de la computadora se divide en dos:

1. Memoria Central o Interna
2. Memoria Auxiliar o Externa

***Memoria Central (interna):*** La CPU utiliza la memoria de la computadora para guardar información mientras trabaja con ella; mientras esta información permanezca en memoria, la computadora puede tener acceso a ella en forma directa. Esta memoria construida internamente se llama memoria de acceso aleatorio (RAM).

La ***memoria interna*** consta de dos áreas de memoria:

La memoria ***RAM (Randon Access Memory):*** Recibe el nombre de memoria principal o memoria del usuario, en ella se almacena información solo mientras la computadora está encendida. Cuando se apaga o arranca nuevamente la computadora, la información se pierde, por lo que se dice que la memoria RAM es una memoria volátil.

La memoria ***ROM (Read Only Memory):*** Es una memoria estática que no puede cambiar, la computadora puede leer los datos almacenados en la memoria ROM, pero no se pueden introducir datos en ella, o cambiar los datos que ahí se encuentran; por lo que se dice que esta memoria es de solo lectura. Los datos de la memoria ROM están grabados en forma permanente y son introducidos por el fabricante de la computadora.

***Memoria Auxiliar (Externa):*** Es donde se almacenan todos los programas o datos que el usuario desee. Los dispositivos de almacenamiento o memorias auxiliares (externas o secundarias) más comúnmente utilizados son: cintas magnéticas y discos magnéticos.

***1.2 Definición de Lenguaje***

***Lenguaje:*** Es una serie de símbolos que sirven para transmitir uno o más mensajes (ideas) entre dos entidades diferentes. A la transmisión de mensajes se le conoce comúnmente como ***comunicación***.

La ***comunicación*** es un proceso complejo que requiere una serie de reglas simples, pero indispensables para poderse llevar a cabo. Las dos principales son las siguientes:

1. Los mensajes deben correr en un sentido a la vez.
2. Debe forzosamente existir 4 elementos: Emisor, Receptor, Medio de Comunicación y Mensaje.

***Lenguajes de Programación***

Es un conjunto de símbolos, caracteres y reglas (programas) que le permiten a las personas comunicarse con la computadora.

Los lenguajes de programación tienen un conjunto de instrucciones que nos permiten realizar operaciones de entrada/salida, calculo, manipulación de textos, lógica/comparación y almacenamiento/recuperación.

Los lenguajes de programación se clasifican en:

1. ***Lenguaje Maquina:*** Son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda comprender y ejecutar el programa. La instrucción en lenguaje maquina se expresan en términos de la unidad de memoria más pequeña el bit (dígito binario 0 o 1).
2. ***Lenguaje de Bajo Nivel (Ensamblador):*** En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos para las operaciones y direcciones simbólicas.
3. ***Lenguaje de Alto Nivel:*** Los lenguajes de programación de alto nivel (BASIC, pascal, cobol, frotran, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos (en general en inglés), lo que facilita la escritura y comprensión del programa.
   1. ***Definición de Algoritmo***

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

***1.4 Tipos de Algoritmos***

1. ***Cualitativos:*** Son aquellos en los que se describen los pasos utilizando palabras.
2. ***Cuantitativos:*** Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

***1.5 Lenguajes Algorítmicos***

Es una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso.

***Tipos de Lenguajes Algorítmicos***

1. ***Gráficos:*** Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo).
2. ***No Gráficos:*** Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocodigo).

***1.6 Metodología para la solución de problemas por medio de computadora***

***1.7 Definición del Problema***

Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea que realice la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.

* 1. ***Análisis del Problema***

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

Los datos de entrada.

Cuál es la información que se desea producir (salida)

Los métodos y fórmulas que se necesitan para procesar los datos.

Una recomendación muy práctica es el que nos pongamos en el lugar de la computadora y analicemos que es lo que necesitamos que nos ordenen y en que secuencia para producir los resultados esperados.

* 1. ***Diseño del Algoritmo***

Las características de un buen algoritmo son:

Debe tener un punto particular de inicio.

Debe ser definido, no debe permitir dobles interpretaciones.

Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.

Debe ser finito en tamaño y tiempo de ejecución.

***1.10 Codificación***

La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del diagrama de flujo o pseudocodigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora, la serie de instrucciones detalladas se le conoce como código fuente, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.

***1.11 Prueba y Depuración***

Los errores humanos dentro de la programación de computadoras son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama ***depuración***.

La ***depuración*** ***o prueba*** resulta una tarea tan creativa como el mismo desarrollo de la solución, por ello se debe considerar con el mismo interés y entusiasmo.

Resulta conveniente observar los siguientes principios al realizar una depuración, ya que de este trabajo depende el éxito de nuestra solución.

***1.12 Documentación***

Es la guía o comunicación escrita es sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.

A menudo un programa escrito por una persona, es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un programa o para facilitar futuras modificaciones (mantenimiento).

La ***documentación*** se divide en tres partes:

Documentación Interna

Documentación Externa

Manual del Usuario

1. Documentación Interna: Son los comentarios o mensaje que se añaden al código fuente para hacer más claro el entendimiento de un proceso.
2. Documentación Externa: Se define en un documento escrito los siguientes puntos:

Descripción del Problema

Nombre del Autor

Algoritmo (diagrama de flujo o pseudocodigo)

Diccionario de Datos

Código Fuente (programa)

1. Manual del Usuario: Describe paso a paso la manera cómo funciona el programa, con el fin de que el usuario obtenga el resultado deseado.

***1.13 Mantenimiento***

Se lleva acabo después de terminado el programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta. Para poder realizar este trabajo se requiere que el programa este correctamente documentado.

**CAPITULO II.**

**ENTIDADES PRIMITIVAS PARA EL DESARROLLO DE**

**ALGORITMOS**

2.1 Tipos de datos

2.2 Expresiones

2.3 Operadores y operandos

2.4 Identificadores como localidades de memoria

**Objetivo educacional:**

El alumno:

* Conocerá las reglas para cambiar fórmulas matemáticas a expresiones válidas para la computadora, además de diferenciar constantes e identificadores y tipos de datos simples.

***2.1 Tipos De Datos***

Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter, tal como ‘b’, un valor entero tal como 35. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable.

Numéricos

Simples Lógicos

Alfanuméricos (string)

Tipos de

datos Arreglos (Vectores, Matrices)

Estructurados Registros

(Def. por el Archivos

usuario) Apuntadores

***Tipos de Datos Simples***

1. ***Datos Numéricos:*** Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.
2. ***Datos Lógicos:*** Son aquellos que solo pueden tener dos valores (cierto o falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).
3. ***Datos Alfanuméricos (String):*** Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

Ejemplo:

“Instituto Tecnológico de Tuxtepec”

“1997”

***2.2 Expresiones***

Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Por ejemplo:

a+(b + 3)/c

Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

Una expresión consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

* Aritméticas
* Relaciónales
* Lógicas

***2.3 Operadores y Operandos***

1. ***Operadores:*** Son elementos que relacionan de forma diferente, los valores de una o mas variables y/o constantes. Es decir, los operadores nos permiten manipular valores.

Aritméticos

Tipos de Operadores Relaciónales

Lógicos

1. ***Operadores Aritméticos:*** Los operadores aritméticos permiten la realización de operaciones matemáticas con los valores (variables y constantes).

Los operadores aritméticos pueden ser utilizados con tipos de datos enteros o reales. Si ambos son enteros, el resultado es entero; si alguno de ellos es real, el resultado es real.

Operando (Operador) Operando

Valor

(constante o variable)

***Operadores Aritméticos***

+ Suma

- Resta

\* Multiplicación

/ División

Mod Modulo (residuo de la división entera)

Ejemplos:

Expresión Resultado

7 / 2 3.5

12 mod 7 5

4 + 2 \* 5 14

***Prioridad de los Operadores Aritméticos***

1. Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera, el paréntesis más interno se evalúa primero.
2. Dentro de una misma expresión los operadores se evalúan en el siguiente orden.

1.- ^ Exponenciación

2.- \*, /, mod Multiplicación, división, modulo.

3.- +, - Suma y resta.

1. Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Ejemplos:

4 + 2 \* 5 = 14

23 \* 2 / 5 = 9.2 46 / 5 = 9.2

3 + 5 \* (10 - (2 + 4)) = 23 3 + 5 \* (10 - 6) = 3 + 5 \* 4 = 3 + 20 = 23

3.5 + 5.09 - 14.0 / 40 = 5.09 3.5 + 5.09 - 3.5 = 8.59 - 3.5 = 5.09

2.1 \* (1.5 + 3.0 \* 4.1) = 28.98 2.1 \* (1.5 + 12.3) = 2.1 \* 13.8 = 28.98

1. ***Operadores Relaciónales:***
2. Se utilizan para establecer una relación entre dos valores.
3. Compara estos valores entre si y esta comparación produce un resultado de certeza o falsedad (verdadero o falso).
4. Los operadores relaciónales comparan valores del mismo tipo (numéricos o cadenas)
5. Tienen el mismo nivel de prioridad en su evaluación.
6. Los operadores relaciónales tiene menor prioridad que los aritméticos.

***Operadores Relaciónales***

> Mayor que

< Menor que

> = Mayor o igual que

< = Menor o igual que

< > Diferente

= Igual

Ejemplos:

Si a = 10 b = 20 c = 30

a + b > c Falso

a - b < c Verdadero

a - b = c Falso

a \* b < > c Verdadero

Ejemplos no lógicos:

a < b < c

10 < 20 < 30

T < 30 (no es lógico porque tiene diferentes operandos)

1. ***Operadores Lógicos:***
2. Estos operadores se utilizan para establecer relaciones entre valores lógicos.
3. Estos valores pueden ser resultado de una expresión relacional.

***Operadores Lógicos***

And Y

Or O

Not Negación

***Operador And***

*Operando1 Operador Operando2 Resultado*

T ANDT T

T F F

F T F

F F F

***Operador Or***

*Operando1 Operador Operando2 Resultado*

T OR T T

T F T

F T T

F F F

***Operador Not***

*Operando Resultado*

T F

F T

Ejemplos:

(a < b) and (b < c)

(10<20) and (20<30)

T and T

T

***Prioridad de los Operadores Lógicos***

Not

And

Or

***Prioridad de los Operadores en General***

1.- ( )

2.- ^

3.- \*, /, Mod, Not

4.- +, -, And

5.- >, <, > =, < =, < >, =, Or

***Ejemplos:***

a = 10 b = 12 c = 13 d =10

**1)** ((a > b)or(a < c)) and ((a = c) or (a > = b))

F T F F

T F

F

**2)** ((a > = b) or (a < d)) and (( a > = d) and (c > d))

F F T T

F T

F

**3)** not (a = c) and (c > b)

F T

T

T

***2.4 Identificadores***

Los *identificadores* representan los datos de un programa (constantes, variables, tipos de datos). Un identificador es una secuencia de caracteres que sirve para identificar una posición en la memoria de la computadora, que nos permite accesar a su contenido.

Ejemplo: Nombre

Num\_hrs

Calif2

***Reglas para formar un Identificador***

1. Debe comenzar con una letra (A a Z, mayúsculas o minúsculas) y no deben contener espacios en blanco.
2. Letras, dígitos y caracteres como la subraya (\_) están permitidos después del primer carácter.
3. La longitud de identificadores puede ser de hasta 8 caracteres.

***Constantes y Variables***

1. ***Constante:*** Una constante es un dato numérico o alfanumérico que no cambia durante la ejecución del programa.

Ejemplo:

pi = 3.1416

1. ***Variable:*** Es un espacio en la memoria de la computadora que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambia durante la ejecución del programa. Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con el cual podamos identificarla dentro de un algoritmo.

Ejemplo:

área = pi \* radio ^ 2

Las variables son: el radio, el área y la constate es pi

***Clasificación de las Variables***

Numéricas

Por su Contenido Lógicas

Alfanuméricas (String)

Variables

De Trabajo

Por su Uso Contadores

Acumuladores

***Por su Contenido***

1. ***Variable Numéricas:*** Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal. Ejemplo:

IVA=0.15 pi=3.1416 costo=2500

1. ***Variables Lógicas:*** Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.
2. ***Variables Alfanuméricas:*** Está formada por caracteres alfanuméricos (letras, números y caracteres especiales). Ejemplo:

letra=’a’ apellido=’López’ dirección=’Av. Libertad #190’

***Por su Uso***

1. ***Variables de Trabajo:*** Variables que reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un programa. Ejemplo:

suma=a+b/c

1. ***Contadores:*** Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con los incrementos generalmente de uno en uno.
2. ***Acumuladores:*** Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

**CAPITULO III.**

**TÉCNICAS DE DISEÑO**

3.1 Top down

3.2 Bottom up

**Objetivo educacional:**

El alumno:

* Conocerá las características de las técnicas de diseño más empleadas, así como su aplicación a cada tipo de problemas

***3.1 Top Down***

También conocida como de arriba-abajo y consiste en establecer una serie de niveles de mayor a menor complejidad (arriba-abajo) que den solución al problema. Consiste en efectuar una relación entre las etapas de la estructuración de forma que una etapa jerárquica y su inmediato inferior se relacionen mediante entradas y salidas de información.

Este diseño consiste en una serie de descomposiciones sucesivas del problema inicial, que recibe el refinamiento progresivo del repertorio de instrucciones que van a formar parte del programa.

La utilización de la técnica de diseño ***Top-Down*** tiene los siguientes objetivos básicos:

- Simplificación del problema y de los subprogramas de cada descomposición.

- Las diferentes partes del problema pueden ser programadas de modo independiente e incluso por diferentes personas.

- El programa final queda estructurado en forma de bloque o módulos lo que hace más sencilla su lectura y mantenimiento.

***3.2 Bottom Up***

El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse con forme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Cuando la programación se realiza internamente y haciendo un enfoque ascendente, es difícil llegar a integrar los subsistemas al grado tal de que el desempeño global, sea fluido. Los problemas de integración entre los subsistemas son sumamente costosos y muchos de ellos no se solucionan hasta que la programación alcanza la fecha límite para la integración total del sistema. En esta fecha, ya se cuenta con muy poco tiempo, presupuesto o paciencia de los usuarios, como para corregir aquellas delicadas interfaces, que, en un principio, se ignoran.

Aunque cada subsistema parece ofrecer lo que se requiere, cuando se contempla al sistema como una entidad global, adolece de ciertas limitaciones por haber tomado un enfoque ascendente. Uno de ellos es la duplicación de esfuerzos para accesar el software y más aún al introducir los datos. Otro es, que se introducen al sistema muchos datos carentes de valor. Un tercero y tal vez el más serio inconveniente del enfoque ascendente, es que los objetivos globales de la organización no fueron considerados y en consecuencia no se satisfacen.

**CAPITULO IV.**

**TÉCNICAS PARA LA FORMULACIÓN DE ALGORITMOS**

4.1 Diagrama de flujo

4.2 Pseudocodigo

4.3 Diagrama estructurado (nassi-schneiderman)

**Objetivo educacional:**

El alumno:

* Será capaz de diferenciar los métodos de representación y formulación de algoritmos, así como de conocer las características más importantes de cada técnica.

Las dos herramientas utilizadas comúnmente para diseñar algoritmos son:

Diagrama de Flujo

Pseuducodigo

***4.1 Diagrama de Flujo***

Un diagrama de flujo es la representación gráfica de un algoritmo. También se puede decir que es la representación detallada en forma gráfica de cómo deben realizarse los pasos en la computadora para producir resultados.

Esta representación gráfica se da cuando varios símbolos (que indican diferentes procesos en la computadora), se relacionan entre sí mediante líneas que indican el orden en que se deben ejecutar los procesos.

Los símbolos utilizados han sido normalizados por el instituto norteamericano de normalización (ANSI).

***SÍMBOLO DESCRIPCIÓN***

Indica el inicio y el final de nuestro diagrama de flujo.

Indica la entrada y salida de datos.

Símbolo de proceso y nos indica la asignación

de un valor en la memoria y/o la ejecución de

una operación aritmética.

Símbolo de decisión indica la realización de una comparación de valores.

Se utiliza para representar los subprogramas.

Conector dentro de página. Representa la continuidad del diagrama dentro de la misma página.

Conector fuera de página. Representa la continuidad del diagrama en otra página.

Indica la salida de información por impresora.

Indica la salida de información en la pantalla o

monitor.

Líneas de flujo o dirección. Indican la

secuencia en que se realizan las operaciones.

***Recomendaciones para el diseño de Diagramas de Flujo***

1. Se deben se usar solamente líneas de flujo horizontales y/o verticales.
2. Se debe evitar el cruce de líneas utilizando los conectores.
3. Se deben usar conectores solo cuando sea necesario.
4. No deben quedar líneas de flujo son conectar.
5. Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.
6. Todo texto escrito dentro de un símbolo deberá ser escrito claramente, evitando el uso de muchas palabras.

***4.2 Pseudocodigo***

Mezcla de lenguaje de programación y español (o inglés o cualquier otro idioma) que se emplea, dentro de la programación estructurada, para realizar el diseño de un programa. En esencial, el pseudocodigo se puede definir como un lenguaje de especificaciones de algoritmos.

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocodigo utiliza palabras que indican el proceso a realizar.

***Ventajas de utilizar un Pseudocodigo a un Diagrama de Flujo***

1. Ocupa menos espacio en una hoja de papel
2. Permite representar en forma fácil operaciones repetitivas complejas
3. Es muy fácil pasar de pseudocodigo a un programa en algún lenguaje de programación.
4. Si se siguen las reglas se puede observar claramente los niveles que tiene cada operación.

***4.3 Diagramas estructurados (Nassi-Schneiderman)***

El diagrama estructurado N-S también conocido como diagrama de chapin es como un diagrama de flujo en el que se omiten las flechas de unión y las cajas son contiguas. Las acciones sucesivas se pueden escribir en cajas sucesivas y como en los diagramas de flujo, se pueden escribir diferentes acciones en una caja. Un algoritmo se represente en la sig. forma:

Inicio

Acción 1

Acción 2

Acción 3

Fin

Inicio

Leer

Nombre, Hrs., Precio

Calcular

Salario = Hrs. \* Precio

Calcular

Imp = Salario\* 0.15

Calcular

Neto = Salario + Imp

Escribir

Nombre, Imp., SNeto

Fin

**CAPITULO V.**

**ESTRUCTURAS ALGORITMICAS**

5.1 Secuenciales

- Asignación

- Entrada

- Salida

5.2 Condicionales

- Simples

- Múltiples

5.3 Repetición fila condicional

**Objetivo educacional:**

El alumno:

* Conocerá las diferentes estructuras algorítmicas como componentes básicos de los programas y aplicará la combinación de ellas para el desarrollo de algoritmos más complejos.

ESTRUCTURAS ALGORITMICAS

Las estructuras de operación de programas son un grupo de formas de trabajo, que permiten, mediante la manipulación de variables, realizar ciertos procesos específicos que nos lleven a la solución de problemas. Estas estructuras se clasifican de acuerdo con su complejidad en:

- Asignación

Secuenciales - Entrada

- Salida

- Simples

Estructuras Condicionales

Algorítmicas - Múltiples

- Hacer para

Cíclicas - Hacer mientras

- Repetir hasta

***5.1. Estructuras Secuenciales***

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. Una estructura secuencial se representa de la siguiente forma:

**Inicio**

Acción 1

Acción 2 .

Acción N

**Fin**

***- Asignación:*** La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

* ***Simples:*** Consiste en pasar un valor constate a una variable (a=15)
* ***Contador:*** Consiste en usarla como un verificador del número de veces que se realiza un proceso (a=a+1)
* ***Acumulador:*** Consiste en usarla como un sumador en un proceso (a=a+b)
* ***De trabajo:*** Donde puede recibir el resultado de una operación matemática que involucre muchas variables (a=c+b\*2/4).

***- Lectura:*** La lectura consiste en recibir desde un dispositivo de entrada (p.ej. el teclado) un valor. Esta operación se representa en un pseudocodigo como sigue:

Leer a, b

Donde “a” y “b” son las variables que recibirán los valores

***Escritura:*** Consiste en mandar por un dispositivo de salida (p.ej. monitor o impresora) un resultado o mensaje. Este proceso se representa en un pseudocodigo como sigue:

Escribe “El resultado es:”, R

Donde “El resultado es:” es un mensaje que se desea aparezca y R es una variable que contiene un valor.

***Problemas Secuenciales***

1) Suponga que un individuo desea invertir su capital en un banco y desea saber cuánto dinero ganara después de un mes si el banco paga a razón de 2% mensual.

**Inicio**

Leer cap\_inv

gan = cap\_inv \* 0.02

Imprimir gan

**Fin**

2) Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

**Inicio**

Leer sb, v1, v2, v3

tot\_vta = v1 + v2 + v3

com = tot\_vta \* 0.10

tpag = sb + com

Imprimir tpag, com

**Fin**

3) Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuánto deberá pagar finalmente por su compra.

**Inicio**

Leer tc

d = tc \* 0.15

tp = tc - d

Imprimir tp

**Fin**

4) Un alumno desea saber cuál será su calificación final en la materia de Algoritmos. Dicha calificación se compone de los siguientes porcentajes:

55% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

15% de la calificación de un trabajo final.

**Inicio**

Leer c1, c2, c3, ef, tf

prom = (c1 + c2 + c3)/3

ppar = prom \* 0.55

pef = ef \* 0.30

ptf = tf \* 0.15

cf = ppar + pef + ptf

Imprimir cf

**Fin**

5) Un maestro desea saber qué porcentaje de hombres y que porcentaje de mujeres hay en un grupo de estudiantes.

**Inicio**

Leer nh, nm

ta = nh + nm

ph = nh \* 100 / ta

pm = nm \* 100 / ta

Imprimir ph, pm

**Fin**

6) Realizar un algoritmo que calcule la edad de una persona.

**Inicio**

Leer fnac, fact

edad = fact - fnac

Imprimir edad

**Fin.**

# Problemas Propuestos

1. Dada una cantidad en pesos, obtener la equivalencia en dólares, asumiendo que la unidad cambiaría es un dato desconocido.
2. Leer un número y escribir el valor absoluto del mismo.

3) La presión, el volumen y la temperatura de una masa de aire se relacionan por la fórmula:

masa = (presión \* volumen) /(0.37 \* (temperatura + 460))

4) Calcular el número de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la fórmula es:

num. pulsaciones = (220 - edad) /10

5) Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.

1. En un hospital existen tres áreas: Ginecología, Pediatría, Traumatología. El presupuesto anual del hospital se reparte conforme a la sig. tabla:

Área Porcentaje del presupuesto

Ginecología 40%

Traumatología 30%

Pediatría 30%

Obtener la cantidad de dinero que recibirá cada área, para cualquier monto presupuestal.

1. El dueño de una tienda compra un artículo a un precio determinado. Obtener el precio en que lo debe vender para obtener una ganancia del 30%.
2. Todos los lunes, miércoles y viernes, una persona corre la misma ruta y cronometra los tiempos obtenidos. Determinar el tiempo promedio que la persona tarda en recorrer la ruta en una semana cualquiera.
3. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada quien invierte con respecto a la cantidad total invertida.
4. Un alumno desea saber cuál será su promedio general en las tres materias más difíciles que cursa y cuál será el promedio que obtendrá en cada una de ellas. Estas materias se evalúan como se muestra a continuación:

La calificación de Matemáticas se obtiene de la sig. manera:

Examen 90%

Promedio de tareas 10%

En esta materia se pidió un total de tres tareas.

La calificación de Física se obtiene de la sig. manera:

Examen 80%

Promedio de tareas 20%

En esta materia se pidió un total de dos tareas.

La calificación de Química se obtiene de la sig. manera:

Examen 85%

Promedio de tareas 15%

En esta materia se pidió un promedio de tres tareas.

***5.2 Estructuras de Condicionales***

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que, en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen dos tipos básicos, las simples y las múltiples.

* ***Simples:*** Las estructuras condicionales simples se les conoce como “Tomas de decisión”. Estas tomas de decisión tienen la siguiente forma:

Si <condición> entonces

Acción(es)

Fin-si

* ***Dobles:*** Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representa de la siguiente forma:

Si <condición> entonces

Acción(es)

si no

Acción(es)

Fin-si

Donde:

Si……………………… Indica el comando de comparación

Condición………… Indica la condición a evaluar

Entonces…………… Precede a las acciones a realizar cuando se cumple la condición

Acción(es)………… Son las acciones a realizar cuando se cumple o no la condición

Si no………………… Precede a las acciones a realizar cuando no se cumple la condición

Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.

* ***Múltiples:*** Las estructuras de comparación múltiples, son tomas de decisión especializadas que permiten comparar una variable contra distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma común es la siguiente:

Si <condición> entonces

Acción(es)

si no

Si <condición> entonces

Acción(es)

si no

.

. Varias condiciones

.

* ***Forma General***

Casos Variable

Op1: Acción(es)

Op2: Acción(es)

.

.

OpN: acción

Fin-casos

# Problemas Condicionales

***a) Problemas Selectivos Simples***

1) Un hombre desea saber cuánto dinero se genera por concepto de intereses sobre la cantidad que tiene en inversión en el banco. El decidirá reinvertir los intereses siempre y cuando estos excedan a $7000, y en ese caso desea saber cuánto dinero tendrá finalmente en su cuenta.

**Inicio**

Leer p\_int, cap

int = cap \* p\_int

si int > 7000 entonces

capf = cap + int

fin-si

Imprimir capf

**Fin**

2) Determinar si un alumno aprueba a reprueba un curso, sabiendo que aprobara si su promedio de tres calificaciones es mayor o igual a 70; reprueba en caso contrario.

**Inicio**

Leer calif1, calif2, calif3

prom = (calif1 + calif2 + calif3) /3

Si prom >= 70 entonces

Imprimir “alumno aprobado”

si no

Imprimir “alumno reprobado”

Fin-si

**Fin**

3) En un almacén se hace un 20% de descuento a los clientes cuya compra supere los $1000 ¿Cuál será la cantidad que pagara una persona por su compra?

**Inicio**

Leer compra

Si compra > 1000 entonces

desc = compra \* 0.20

si no

desc = 0

fin-si

tot\_pag = compra - desc

imprimir tot\_pag

**Fin**

4) Un obrero necesita calcular su salario semanal, el cual se obtiene de la sig. manera:

Si trabaja 40 horas o menos se le paga $16 por hora

Si trabaja más de 40 horas se le paga $16 por cada una de las primeras 40 horas y $20 por cada hora extra.

**Inicio**

Leer ht

Si ht > 40 entonces

he = ht - 40

ss = he \* 20 + 40 \* 16

si no

ss = ht \* 16

Fin-si

Imprimir ss

**Fin**

5) Un hombre desea saber cuánto dinero se genera por concepto de intereses sobre la cantidad que tiene en inversión en el banco. El decidirá reinvertir los intereses siempre y cuando estos excedan a $7000, y en ese caso desea saber cuánto dinero tendrá finalmente en su cuenta.

**Inicio**

Leer p\_int, cap

int = cap \* p\_int

si int > 7000 entonces

capf = cap + int

fin-si

Imprimir capf

**Fin**

6) Que lea dos números y los imprima en forma ascendente

**Inicio**

Leer num1, num2

Si num1 < num2 entonces

Imprimir num1, num2

si no

Imprimir num2, num1

fin-si

**Fin**

7) Una persona enferma, que pesa 70 kg, se encuentra en reposo y desea saber cuántas calorías consume su cuerpo durante todo el tiempo que realice una misma actividad. Las actividades que tiene permitido realizar son únicamente dormir o estar sentado en reposo. Los datos que tiene son que estando dormido consume 1.08 calorías por minuto y estando sentado en reposo consume 1.66 calorías por minuto.

**Inicio**

Leer act$, tiemp

Si act$ = “dormido” entonces

cg = 1.08 \* tiemp

si no

cg = 1.66 \* tiemp

fin-si

Imprimir cg

**Fin**

8) Hacer un algoritmo que imprima el nombre de un artículo, clave, precio original y su precio con descuento. El descuento lo hace en base a la clave, si la clave es 01 el descuento es del 10% y si la clave es 02 el descuento en del 20% (solo existen dos claves).

**Inicio**

Leer nomb, cve, prec\_orig

Si cve = 01 entonces

prec\_desc = prec\_orig - prec\_orig \* 0.10

si no

prec\_desc = prec\_orig - prec\_orig \* 0.20

fin-si

Imprimir nomb, cve, prec\_orig, prec\_desc

**Fin**

9) Hacer un algoritmo que calcule el total a pagar por la compra de camisas. Si se compran tres camisas o más se aplica un descuento del 20% sobre el total de la compra y si son menos de tres camisas un descuento del 10%

**Inicio**

Leer num\_camisas, prec

tot\_comp = num\_camisas \* prec

Si num\_camisas > = 3 entonces

tot\_pag = tot\_comp - tot\_comp \* 0.20

si no

tot\_pag = tot\_comp - tot\_comp \* 0.10

fin-si

Imprimir tot\_pag

**Fin**

10) Una empresa quiere hacer una compra de varias piezas de la misma clase a una fábrica de refacciones. La empresa, dependiendo del monto total de la compra, decidirá qué hacer para pagar al fabricante.

Si el monto total de la compra excede de $500 000 la empresa tendrá la capacidad de invertir de su propio dinero un 55% del monto de la compra, pedir prestado al banco un 30% y el resto lo pagará solicitando un crédito al fabricante.

Si el monto total de la compra no excede de $500 000 la empresa tendrá capacidad de invertir de su propio dinero un 70% y el restante 30% lo pagará solicitando crédito al fabricante.

El fabricante cobra por concepto de intereses un 20% sobre la cantidad que se le pague a crédito.

**Inicio**

Leer costopza, numpza

totcomp = costopza \* numpza

Si totcomp > 500 000 entonces

cantinv = totcomp \* 0.55

préstamo = totcomp \* 0.30

crédito = totcomp \* 0.15

si no

cantinv = totcomp \* 0.70

crédito = totcomp \* 0.30

préstamo = 0

fin-si

int = crédito \* 0.20

Imprimir cantinv, préstamo, crédito, int

**Fin**

# Problemas Propuesto

1) Calcular el total que una persona debe pagar en una llantera, si el precio de cada llanta es de $800 si se compran menos de 5 llantas y de $700 si se compran 5 o más.

2) En un supermercado se hace una promoción, mediante la cual el cliente obtiene un descuento dependiendo de un número que se escoge al azar. Si el numero escogido es menor que 74 el descuento es del 15% sobre el total de la compra, si es mayor o igual a 74 el descuento es del 20%. Obtener cuánto dinero se le descuenta.

3) Calcular el número de pulsaciones que debe tener una persona por cada 10 segundos de ejercicio aeróbico; la fórmula que se aplica cuando el sexo es femenino es:

num. pulsaciones = (220 - edad) /10

y si el sexo es masculino:

num. pulsaciones = (210 - edad) /10

4) Una compañía de seguros está abriendo un depto. de finanzas y estableció un programa para captar clientes, que consiste en lo siguiente: Si el monto por el que se efectúa la fianza es menor que $50 000 la cuota a pagar será por el 3% del monto, y si el monto es mayor que $50 000 la cuota a pagar será el 2% del monto. La afianzadora desea determinar cuál será la cuota que debe pagar un cliente.

5) En una escuela la colegiatura de los alumnos se determina según el número de materias que cursan. El costo de todas las materias es el mismo.

Se ha establecido un programa para estimular a los alumnos, el cual consiste en lo siguiente: si el promedio obtenido por un alumno en el último periodo es mayor o igual que 9, se le hará un descuento del 30% sobre la colegiatura y no se le cobrara IVA; si el promedio obtenido es menor que 9 deberá pagar la colegiatura completa, la cual incluye el 10% de IVA.

Obtener cuanto debe pagar un alumno.

6) Una empresa de bienes raíces ofrece casas de interés social, bajo las siguientes condiciones: Si los ingresos del comprador son menores de $8000 o más el enganche será del 15% del costo de la casa y el resto se distribuirá en pagos mensuales, a pagar en diez años. Si los ingresos del comprador son menos de $8000 o más el enganche será del 30% del costo de la casa y el resto se distribuirá en pagos mensuales a pagar en 7 años.

La empresa quiere obtener cuanto debe pagar un comprador por concepto de enganche y cuanto por cada pago parcial.

7) El gobierno ha establecido el programa SAR (Sistema de Ahorro para el Retiro) que consiste en que los dueños de la empresa deben obligatoriamente depositar en una cuenta bancaria un porcentaje del salario de los trabajadores; adicionalmente los trabajadores pueden solicitar a la empresa que deposite directamente una cuota fija o un porcentaje de su salario en la cuenta del SAR, la cual le será descontada de su pago.

Un trabajador que ha decidido aportar a su cuenta del SAR desea saber la cantidad total de dinero que estará depositado a esa cuenta cada mes, y el pago mensual que recibirá.

8) Una persona desea iniciar un negocio, para lo cual piensa verificar cuánto dinero le prestara el banco por hipotecar su casa. Tiene una cuenta bancaria, pero no quiere disponer de ella a menos que el monto por hipotecar su casa sea muy pequeño. Si el monto de la hipoteca es menor que $1 000 000 entonces invertirá el 50% de la inversión total y un socio invertirá el otro 50%. Si el monto de la hipoteca es de $ 1 000 000 o más, entonces invertirá el monto total de la hipoteca y el resto del dinero que se necesite para cubrir la inversión total se repartirá a partes iguales entre el socio y el.

9) El gobierno del estado de México desea reforestar un bosque que mide determinado número de hectáreas. Si la superficie del terreno excede a 1 millón de metros cuadrados, entonces decidirá sembrar de la sig. manera:

Porcentaje de la superficie del bosque Tipo de árbol

70% pino

20% oyamel

10% cedro

Si la superficie del terreno es menor o igual a un millón de metros cuadrados, entonces decidirá sembrar de la sig. manera:

Porcentaje de la superficie del bosque Tipo de árbol

50% pino

30% oyamel

20% cedro

El gobierno desea saber el número de pinos, oyameles y cedros que tendrá que sembrar en el bosque, si se sabe que en 10 metros cuadrados caben 8 pinos, en 15 metros cuadrados caben 15 oyameles y en 18 metros cuadrados caben 10 cedros. También se sabe que una hectárea equivale a 10 mil metros cuadrados.

10) Una fábrica ha sido sometida a un programa de control de contaminación para lo cual se efectúa una revisión de los puntos IMECA generados por la fábrica. El programa de control de contaminación consiste en medir los puntos IMECA que emite la fábrica en cinco días de una semana y si el promedio es superior a los 170 puntos entonces tendrá la sanción de parar su producción por una semana y una multa del 50% de las ganancias diarias cuando no se detiene la producción. Si el promedio obtenido de puntos IMECA es de 170 o menor entonces no tendrá ni sanción ni multa. El dueño de la fábrica desea saber cuánto dinero perderá después de ser sometido a la revisión.

11) Una persona se encuentra con un problema de comprar un automóvil o un terreno, los cuales cuestan exactamente lo mismo. Sabe que mientras el automóvil se devalúa, con el terreno sucede lo contrario. Esta persona comprara el automóvil si al cabo de tres años la devaluación de este no es mayor que la mitad del incremento del valor del terreno. Ayúdale a esta persona a determinar si debe o no comprar el automóvil.

# Problemas Selectivos Compuestos

1) Leer 2 números; si son iguales que los multiplique, si el primero es mayor que el segundo que los reste y si no que los sume.

**Inicio**

Leer num1, num2

si num1 = num2 entonces

resul = num1 \* num2

si no

si num1 > num2 entonces

resul = num1 - num2

si no

resul = num1 + num2

fin-si

fin-si

**Fin**

2) Leer tres números diferentes e imprimir el número mayor de los tres.

**Inicio**

Leer num1, num2, num3

Si (num1 > num2) and (num1 > num3) entonces

mayor = num1

si no

Si (num2 > num1) and (num2 > num3) entonces

mayor = num2

si no

mayor = num3

fin-si

fin-si

Imprimir mayor

**Fin**

3) Determinar la cantidad de dinero que recibirá un trabajador por concepto de las horas extras trabajadas en una empresa, sabiendo que cuando las horas de trabajo exceden de 40, el resto se consideran horas extras y que estas se pagan al doble de una hora normal cuando no exceden de 8; si las horas extras exceden de 8 se pagan las primeras 8 al doble de lo que se pagan las horas normales y el resto al triple.

**Inicio**

Leer ht, pph

Si ht < = 40 entonces

tp = ht \* pph

si no

he = ht - 40

Si he < = 8 entonces

pe = he \* pph \* 2

si no

pd = 8 \* pph \* 2

pt = (he - 8) \* pph \* 3

pe = pd + pt

fin-si

tp = 40 \* pph + pe

fin-si

Imprimir tp

**Fin**

4) Calcular la utilidad que un trabajador recibe en el reparto anual de utilidades si este se le asigna como un porcentaje de su salario mensual que depende de su antigüedad en la empresa de acuerdo con la sig. tabla:

Tiempo Utilidad

Menos de 1 año 5 % del salario

1 año o más y menos de 2 años 7% del salario

2 años o más y menos de 5 años 10% del salario

5 años o más y menos de 10 años 15% del salario

10 años o más 20% del salario

**Inicio**

Leer sm, antig

Si antig < 1 entonces

util = sm \* 0.05

si no

Si (antig > = 1) and (antig < 2) entonces

util = sm \* 0.07

si no

Si (antig > = 2) and (antig < 5) entonces

util = sm \* 0.10

si no

Si (antig > = 5) and (antig < 10) entonces

util = sm \* 0.15

si no

util = sm \* 0.20

fin-si

fin-si

fin-si

fin-si

Imprimir util

**Fin**

5) En una tienda de descuento se efectúa una promoción en la cual se hace un descuento sobre el valor de la compra total según el color de la bolita que el cliente saque al pagar en caja. Si la bolita es de color blanco no se le hará descuento alguno, si es verde se le hará un 10% de descuento, si es amarilla un 25%, si es azul un 50% y si es roja un 100%. Determinar la cantidad final que el cliente deberá pagar por su compra. se sabe que solo hay bolitas de los colores mencionados.

**Inicio**

leer tc, b$

si b$ = ‘blanca’ entonces

d=0

si no

si b$ = ‘verde’ entonces

d=tc\*0.10

si no

si b$ = ‘amarilla’ entonces

d=tc\*0.25

si no

si b$ = ‘azul’ entonces

d=tc\*0.50

si no

d=tc

fin-si

fin-si

fin-si

fin-si

**Fin**

6) El IMSS requiere clasificar a las personas que se jubilaran en el año de 1997. Existen tres tipos de jubilaciones: por edad, por antigüedad joven y por antigüedad adulta. Las personas adscritas a la jubilación por edad deben tener 60 años o más y una antigüedad en su empleo de menos de 25 años. Las personas adscritas a la jubilación por antigüedad joven deben tener menos de 60 años y una antigüedad en su empleo de 25 años o más.

Las personas adscritas a la jubilación por antigüedad adulta deben tener 60 años o más y una antigüedad en su empleo de 25 años o más.

Determinar en qué tipo de jubilación, quedara adscrita una persona.

**Inicio**

leer edad,ant

si edad >= 60 and ant < 25 entonces

imprimir “la jubilación es por edad”

si no

si edad >= 60 and ant > 25 entonces

imprimir “la jubilación es por edad adulta”

si no

si edad < 60 and ant > 25 entonces

imprimir “la jubilación es por antigüedad joven”

si no

imprimir “no tiene por qué jubilarse”

fin-si

fin-si

fin-si

**Fin**

# Problemas Propuestos

1) En una fábrica de computadoras se planea ofrecer a los clientes un descuento que dependerá del número de computadoras que compre. Si las computadoras son menos de cinco se les dará un 10% de descuento sobre el total de la compra; si el número de computadoras es mayor o igual a cinco, pero menos de diez se le otorga un 20% de descuento; y si son 10 o más se les da un 40% de descuento. El precio de cada computadora es de $11,000

2) En una llantera se ha establecido una promoción de las llantas marca “Ponchadas”, dicha promoción consiste en lo siguiente:

Si se compran menos de cinco llantas el precio es de $300 cada una, de $250 si se compran de cinco a 10 y de $200 si se compran más de 10.

Obtener la cantidad de dinero que una persona tiene que pagar por cada una de las llantas que compra y la que tiene que pagar por el total de la compra.

3) En un juego de preguntas a las que se responde “Si” o “No” gana quien responda correctamente las tres preguntas. Si se responde mal a cualquiera de ellas ya no se pregunta la siguiente y termina el juego. Las preguntas son:

1. Colon descubrió América?

2. La independencia de México fue en el año 1810?

3. The Doors fue un grupo de rock americano?

4) Un proveedor de estéreos ofrece un descuento del 10% sobre el precio sin IVA, de algún aparato si esta cuesta $2000 o más. Además, independientemente de esto, ofrece un 5% de descuento si la marca es “NOSY”. Determinar cuánto pagara, con IVA incluido, un cliente cualquiera por la compra de su aparato.

5) Una frutería ofrece las manzanas con descuento según la siguiente tabla:

NUM. DE KILOS COMPRADOS % DESCUENTO

0 - 2 0%

2.01 - 5 10%

5.01 - 10 15%

10.01 en adelante 20%

Determinar cuánto pagara una persona que compre manzanas es esa frutería.

6) El dueño de una empresa desea planificar las decisiones financieras que tomara en el siguiente año. La manera de planificarlas depende de lo siguiente:

Si actualmente su capital se encuentra con saldo negativo, pedirá un préstamo bancario para que su nuevo saldo sea de $10 000. Si su capital tiene actualmente un saldo positivo pedirá un préstamo bancario para tener un nuevo saldo de $20 000, pero si su capital tiene actualmente un saldo superior a los $20 000 no pedirá ningún préstamo.

Posteriormente repartirá su presupuesto de la siguiente manera.

$5 000 para equipo de computo

$2 000 para mobiliario

y el resto la mitad será para la compra de insumos y la otra para otorgar incentivos al personal.

Desplegar que cantidades se destinaran para la compra de insumos e incentivos al personal y, en caso de que fuera necesario, a cuánto ascendería la cantidad que se pediría al banco.

7) Tomando como base los resultados obtenidos en un laboratorio de análisis clínicos, un médico determina si una persona tiene anemia o no, lo cual depende de su nivel de hemoglobina en la sangre, de su edad y de su sexo. Si el nivel de hemoglobina que tiene una persona es menor que el rango que le corresponde, se determina su resultado como positivo y en caso contrario como negativo. La tabla en la que el medico se basa para obtener el resultado es la siguiente:

EDAD NIVEL HEMOGLOBINA

0 - 1 mes 13 - 26 g%

> 1 y < = 6 meses 10 - 18 g%

> 6 y < = 12 meses 11 - 15 g%

> 1 y < = 5 años 11.5 - 15 g%

> 5 y < = 10 años 12.6 - 15.5 g%

> 10 y < = 15 años 13 - 15.5 g%

mujeres > 15 años 12 - 16 g%

hombres > 15 años 14 - 18 g%

8) Una institución educativa estableció un programa para estimular a los alumnos con buen rendimiento académico y que consiste en lo siguiente:

Si el promedio es de 9.5 o más y el alumno es de preparatoria, entonces este podrá cursar 55 unidades y se le hará un 25% de descuento.

Si el promedio es mayor o igual a 9 pero menor que 9.5 y el alumno es de preparatoria, entonces este podrá cursar 50 unidades y se le hará un 10% de descuento.

Si el promedio es mayor que 7 y menor que 9 y el alumno es de preparatoria, este podrá cursar 50 unidades y no tendrá ningún descuento.

Si el promedio es de 7 o menor, el número de materias reprobadas es de 0 a 3 y el alumno es de preparatoria, entonces podrá cursar 45 unidades y no tendrá descuento.

Si el promedio es de 7 o menor, el número de materias reprobadas es de 4 o más y el alumno es de preparatoria, entonces podrá cursar 40 unidades y no tendrá ningún descuento.

Si el promedio es mayor o igual a 9.5 y el alumno es de profesional, entonces podrá cursar 55 unidades y se le hará un 20% de descuento.

Si el promedio es menor de 9.5 y el alumno es de profesional, entonces podrá cursar 55 unidades y no tendrá descuento.

Obtener el total que tendrá que pagar un alumno si la colegiatura para alumnos de profesional es de $300 por cada cinco unidades y para alumnos de preparatoria es de $180 por cada cinco unidades.

9) Que lea tres números diferentes y determine el número medio del conjunto de los tres números (el número medio es aquel número que no es ni mayor, ni menor).

***5.3. Estructuras Cíclicas***

Se llaman problemas repetitivos o cíclicos a aquellos en cuya solución es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces. Esta cantidad puede ser fija (previamente determinada por el programador) o puede ser variable (estar en función de algún dato dentro del programa). Los ciclos se clasifican en:

* ***Ciclos con un Número Determinado de Iteraciones (Hacer-Para)***

Son aquellos en que el número de iteraciones se conoce antes de ejecutarse el ciclo. La forma de esta estructura es la siguiente:

Hacer para V.C = L.I a L.S VC=LI

Accion1 vc=vc+1 Vc = LS V

Accion2

. F

. Cuerpo del ciclo

.

AccionN

Fin-para

Donde:

V.C Variable de control del ciclo

L.I Limite inferir

L.S Límite superior

En este ciclo la variable de control toma el valor inicial del ciclo y el ciclo se repite hasta que la variable de control llegue al límite superior.

***Problemas (Hacer para)***

1) Calcular el promedio de un alumno que tiene 7 calificaciones en la materia de Diseño Estructurado de Algoritmos

**Inicio**

Sum=0

Leer Nom

Hacer para c = 1 a 7

Leer calif

Sum = sum + calif

Fin-para

prom = sum /7

Imprimir prom

**Fin**

2) Leer 10 números y obtener su cubo y su cuarta.

**Inicio**

Hacer para n = 1 a 10

Leer num

cubo = num \* num \* num

cuarta = cubo \* num

Imprimir cubo, cuarta

Fin-para

**Fin**

3) Leer 10 números e imprimir solamente los números positivos

**Inicio**

Hacer para n = 1 a 10

Leer num

Si num > 0 entonces

Imprimir num

fin-si

Fin-para

**Fin**

4) Leer 20 números e imprimir cuantos son positivos, cuantos negativos y cuantos neutros.

**Inicio**

cn = 0

cp = 0

cneg = 0

Hacer para x = 1 a 20

Leer num

Sin num = 0 entonces

cn = cn + 1

si no

Si num > 0 entonces

cp = cp + 1

si no

cneg = cneg + 1

Fin-si

Fin-si

Fin-para

Imprimir cn, cp, cneg

**Fin**

5) Leer 15 números negativos y convertirlos a positivos e imprimir dichos números.

**Inicio**

Hacer para x = 1 a 15

Leer num

pos = num \* -1

Imprimir num, pos

Fin-para

**Fin**

6) Suponga que se tiene un conjunto de calificaciones de un grupo de 40 alumnos. Realizar un algoritmo para calcular la calificación media y la calificación más baja de todo el grupo.

**Inicio**

sum = 0

baja = 9999

Hacer para a = 1 a 40

Leer calif

sum = sum + calif

Si calif < baja entonces

baja = calif

fin-si

Fin-para

media = sum / 2

Imprimir media, baja

**Fin**

7) Calcular e imprimir la tabla de multiplicar de un numero cualquiera. Imprimir el multiplicando, el multiplicador y el producto.

**Inicio**

Leer num

Hacer para X = 1 a 10

resul = num \* x

Imprimir num, “ \* “, X, “ = “, resul

Fin-para

**Fin**

8) Simular el comportamiento de un reloj digital, imprimiendo la hora, minutos y segundos de un día desde las 0:00:00 horas hasta las 23:59:59 horas

**Inicio**

Hacer para h = 1 a 23

Hacer para m = 1 a 59

Hacer para s = 1 a 59

Imprimir h, m, s

Fin-para

Fin-para

Fin-para

**Fin**

***Problemas Propuestos***

1) Una persona debe realizar un muestreo con 50 personas para determinar el promedio de peso de los niños, jóvenes, adultos y viejos que existen en su zona habitacional. Se determinan las categorías con base en la sig., tabla:

CATEGORIA EDAD

Niños 0 - 12

Jóvenes 13 - 29

Adultos 30 - 59

Viejos 60 en adelante

2) Al cerrar un expendio de naranjas, 15 clientes que aún no han pagado recibirán un 15% de descuento si compran más de 10 kilos. Determinar cuánto pagará cada cliente y cuanto percibirá la tienda por esas compras.

3) En un centro de verificación de automóviles se desea saber el promedio de puntos contaminantes de los primeros 25 automóviles que lleguen. Asimismo, se desea saber los puntos contaminantes del carro que menos contamino y del que más contamino.

4) Un entrenador le ha propuesto a un atleta recorrer una ruta de cinco kilómetros durante 10 días, para determinar si es apto para la prueba de 5 Kilómetros o debe buscar otra especialidad. Para considerarlo apto debe cumplir por lo menos una de las siguientes condiciones:

- Que en ninguna de las pruebas haga un tiempo mayor a 16 minutos.

- Que al menos en una de las pruebas realice un tiempo mayor a 16 minutos.

- Que su promedio de tiempos sea menor o igual a 15 minutos.

5) Un Zoólogo pretende determinar el porcentaje de animales que hay en las siguientes tres categorías de edades: de 0 a 1 año, de más de 1 año y menos de 3 y de 3 o más años. El zoológico todavía no está seguro del animal que va a estudiar. Si se decide por elefantes solo tomara una muestra de 20 de ellos; si se decide por las jirafas, tomara 15 muestras, y si son chimpancés tomara 40.

* ***Ciclos con un Número Indeterminado de Iteraciones (Hacer-Mientras, Repetir-Hasta)***

Son aquellos en que el número de iteraciones no se conoce con exactitud, ya que esta dado en función de un dato dentro del programa.

* ***Hacer-Mientras:*** Esta es una estructura que repetirá un proceso durante “N” veces, donde “N” puede ser fijo o variable. Para esto, la instrucción se vale de una condición que es la que debe cumplirse para que se siga ejecutando. Cuando la condición ya no se cumple, entonces ya no se ejecuta el proceso. La forma de esta estructura es la siguiente:

Hacer mientras <condición>

Accion1 NO

CONDICION

Accion2

.

. SI

CUERPO DEL

CICLO

Acción N

Fin-mientras

***Problemas (Hacer Mientras)***

1) Una compañía de seguros tiene contratados a *n* vendedores. Cada uno hace tres ventas a la semana. Su política de pagos es que un vendedor recibe un sueldo base, y un 10% extra por comisiones de sus ventas. El gerente de su compañía desea saber cuánto dinero obtendrá en la semana cada vendedor por concepto de comisiones por las tres ventas realizadas, y cuanto tomando en cuenta su sueldo base y sus comisiones.

2) En una empresa se requiere calcular el salario semanal de cada uno de los *n* obreros que laboran en ella. El salario se obtiene de la sig. forma:

Si el obrero trabaja 40 horas o menos se le paga $20 por hora

Si trabaja más de 40 horas se le paga $20 por cada una de las primeras 40 horas y $25 por cada hora extra.

3) Determinar cuántos hombres y cuantas mujeres se encuentran en un grupo de *n* personas, suponiendo que los datos son extraídos alumno por alumno.

4) El Depto. de Seguridad Publica y Transito del D.F. desea saber, de los *n* autos que entran a la ciudad de México, cuantos entran con calcomanía de cada color. Conociendo el ultimo dígito de la placa de cada automóvil se puede determinar el color de la calcomanía utilizando la sig. relación:

DÍGITO COLOR

1 o 2 amarilla

3 o 4 rosa

5 o 6 roja

7 o 8 verde

9 o 0 azul

5) Obtener el promedio de calificaciones de un grupo de *n* alumnos.

6) Una persona desea invertir su dinero en un banco, el cual le otorga un 2% de interés. ¿Cuál será la cantidad de dinero que esta persona tendrá al cabo de un año si la ganancia de cada mes es reinvertida?

7) Calcular el promedio de edades de hombres, mujeres y de todo un grupo de alumnos.

8) Encontrar el menor valor de un conjunto de *n* números dados.

9) Encontrar el mayor valor de un conjunto de *n* números dados.

10) En un supermercado un cajero captura los precios de los artículos que los clientes compran e indica a cada cliente cual es el monto de lo que deben pagar. Al final del día le indica a su supervisor cuanto fue lo que cobro en total a todos los clientes que pasaron por su caja.

11) Cinco miembros de un club contra la obesidad desean saber cuánto han bajado o subido de peso desde la última vez que se reunieron. Para esto se debe realizar un ritual de pesaje en donde cada uno se pesa en diez básculas distintas para así tener el promedio más exacto de su peso. Si existe diferencia positiva entre este promedio de peso y el peso de la última vez que se reunieron, significa que subieron de peso. Pero si la diferencia es negativa, significa que bajaron. Lo que el problema requiere es que por cada persona se imprima un letrero que diga: “SUBIO” o “BAJO” y la cantidad de kilos que subió o bajo de peso.

3) Se desea obtener el promedio de *g* grupos que están en un mismo año escolar; siendo que cada grupo puede tener *n* alumnos que cada alumno puede llevar *m* materias y que en todas las materias se promedian tres calificaciones para obtener el promedio de la materia. Lo que se desea desplegar es el promedio de los grupos, el promedio de cada grupo y el promedio de cada alumno.

* ***Repetir-Hasta:*** Esta es una estructura similar en algunas características, a la anterior. Repite un proceso una cantidad de veces, pero a diferencia del Hacer-Mientras, el Repetir-Hasta lo hace hasta que la condición se cumple y no mientras, como en el Hacer-Mientras. Por otra parte, esta estructura permite realizar el proceso cuando menos una vez, ya que la condición se evalúa al final del proceso, mientras que en el Hacer-Mientras puede ser que nunca llegue a entrar si la condición no se cumple desde un principio. La forma de esta estructura es la siguiente:

Repetir

Acción 1

CUERPO DEL

CICLO

Acción 2

.

SI

.

CONDICION

Acción N

Hasta <condición>

NO

# Problemas Repetir - Hasta

1) En una tienda de descuento las personas que van a pagar el importe de su compra llegan a la caja y sacan una bolita de color, que les dirá que descuento tendrán sobre el total de su compra. Determinar la cantidad que pagara cada cliente desde que la tienda abre hasta que cierra. Se sabe que si el color de la bolita es rojo el cliente obtendrá un 40% de descuento; si es amarilla un 25% y si es blanca no obtendrá descuento.

2) En un supermercado una ama de casa pone en su carrito los artículos que va tomando de los estantes. La señora quiere asegurarse de que el cajero le cobre bien lo que ella ha comprado, por lo que cada vez que toma un artículo anota su precio junto con la cantidad de artículos iguales que ha tomado y determina cuánto dinero gastara en ese artículo; a esto le suma lo que ira gastando en los demás artículos, hasta que decide que ya tomo todo lo que necesitaba. Ayúdale a esta señora a obtener el total de sus compras.

3) un teatro otorga descuentos según la edad del cliente. determinar la cantidad de dinero que el teatro deja de percibir por cada una de las categorías. Tomar en cuenta que los niños menores de 5 años no pueden entrar al teatro y que existe un precio único en los asientos. Los descuentos se hacen tomando en cuenta el siguiente cuadro:

Edad Descuento

Categoría 1 5 - 14 35 %

Categoría 2 15 - 19 25 %

Categoría 3 20 - 45 10 %

Categoría 4 46 - 65 25 %

Categoría 5 66 en adelante 35 %

# Problemas Propuestos

1) La presión, volumen y temperatura de una masa de aire se relacionan por la fórmula:

masa= presión \* volumen .

0.37 \* (temperatura + 460)

Calcular el promedio de masa de aire de los neumáticos de *n* vehículos que están en compostura en un servicio de alineación y balanceo. Los vehículos pueden ser motocicletas o automóviles.

2) Determinar la cantidad semanal de dinero que recibirá cada uno de los *n* obreros de una empresa. Se sabe que cuando las horas que trabajo un obrero exceden de 40, el resto se convierte en horas extras que se pagan al doble de una hora normal, cuando no exceden de 8; cuando las horas extras exceden de 8 se pagan las primeras 8 al doble de lo que se paga por una hora normal y el resto al triple.

3) En una granja se requiere saber alguna información para determinar el precio de venta por cada kilo de huevo. Es importante determinar el promedio de calidad de las *n* gallinas que hay en la granja. La calidad de cada gallina se obtiene según la fórmula:

calidad = peso de la gallina \* altura de la gallina

numero de huevos que pone

Finalmente, para fijar el precio del kilo de huevo, se toma como base la siguiente tabla:

PRECIO TOTAL DE CALIDAD PESO POR KILO DE HUEVO

mayor o igual que 15 1.2 \* promedio de calidad

mayor que 8 y menor que 15 1.00 \* promedio de calidad

menor o igual que 8 0.80 \* promedio de calidad

4) En la Cámara de Diputados se levanta una encuesta con todos los integrantes con el fin de determinar qué porcentaje de los *n* diputados está a favor del Tratado de Libre Comercio, que porcentaje está en contra y que porcentaje se abstiene de opinar.

5) Una persona que va de compras a la tienda “Enano, S.A.”, decide llevar un control sobre lo que va comprando, para saber la cantidad de dinero que tendrá que pagar al llegar a la caja. La tienda tiene una promoción del 20% de descuento sobre aquellos artículos cuya etiqueta sea roja. Determinar la cantidad de dinero que esta persona deberá pagar.

6) Un censador recopila ciertos datos aplicando encuestas para el último Censo Nacional de Población y Vivienda. Desea obtener de todas las personas que alcance a encuestar en un día, que porcentaje tiene estudios de primaria, secundaria, carrera técnica, estudios profesionales y estudios de posgrado.

7) Un jefe de casilla desea determinar cuántas personas de cada una de las secciones que componen su zona asisten el día de las votaciones. Las secciones son: norte, sur y centro. También desea determinar cuál es la sección con mayor número de votantes.

8) Un negocio de copias tiene un límite de producción diaria de 10 000 copias si el tipo de impresión es offset y de 50 000 si el tipo es estándar. Si hay una solicitud de un el empleado tiene que verificar que las copias pendientes hasta el momento y las copias solicitadas no excedan del límite de producción. Si el límite de producción se excediera el trabajo solicitado no podría ser aceptado. El empleado necesita llevar un buen control de las copias solicitadas hasta el momento para decidir en forma rápida si los trabajos que se soliciten en el día se deben aceptar o no.

9) Calcular la suma siguiente:

100 + 98 + 96 + 94 + . . . + 0 en este orden

10) Leer 50 calificaciones de un grupo de alumnos. Calcule y escriba el porcentaje de reprobados. Tomando en cuenta que la calificación mínima aprobatoria es de 70.

11) Leer por cada alumno de Diseño estructurado de algoritmos su número de control y su calificación en cada una de las 5 unidades de la materia. Al final que escriba el número de control del alumno que obtuvo mayor promedio. Suponga que los alumnos tienen diferentes promedios.

12) El profesor de una materia desea conocer la cantidad de sus alumnos que no tienen derecho al examen de nivelación.

Diseñe un algoritmo que lea las calificaciones obtenidas en las 5 unidades por cada uno de los 40 alumnos y escriba la cantidad de ellos que no tienen derecho al examen de nivelación.

13) Leer los 250,000 votos otorgados a los 3 candidatos a gobernador e imprimir el número del candidato ganador y su cantidad de votos.

14) Suponga que tiene usted una tienda y desea registrar las ventas en su computadora. Diseñe un algoritmo que lea por cada cliente, el monto total de su compra. Al final del día que escriba la cantidad total de ventas y el número de clientes atendidos.

***Problemas Repetitivos Compuestos***

1.- El profesor de una materia desea conocer la cantidad de sus alumnos que no tienen derecho al examen de nivelación.

Diseñe un pseudocódigo que lea las calificaciones obtenidas en las 5 unidades por cada uno de los 40 alumnos y escriba la cantidad de ellos que no tienen derecho al examen de nivelación.

2.- Diseñe un diagrama que lea los 2,500,000 votos otorgados a los 3 candidatos a gobernador e imprima el número del candidato ganador y su cantidad de votos.

3.- Suponga que tiene usted una tienda y desea registrar las ventas en una computadora. Diseñe un pseudocódigo que lea por cada cliente, el monto total de su compra. Al final del día escriba la cantidad total de las ventas y el número de clientes atendidos.

4.- Suponga que tiene una tienda y desea registrar sus ventas por medio de una computadora. Diseñe un pseudocódigo que lea por cada cliente:

a). - el monto de la venta,

b). - calcule e imprima el IVA,

c). -calcule e imprima el total a pagar,

d). - lea la cantidad con que paga el cliente,

e). -calcule e imprime el cambio.

Al final del día deberá imprimir la cantidad de dinero que debe haber en la caja.

5.- Modificar el pseudocódigo anterior de tal forma que no permita que la cantidad con la que paga el cliente sea menor a lo que debe pagar.

6.- Se tiene un conjunto de 1,000 tarjetas cada una contiene la información del censo para una persona:

1.- Número de censo

2.- Sexo

3.- Edad

4.- Estado civil (a.- soltero, b. Casado, c. Viudo, d. Divorciado)

Diseñe un pseudocódigo estructurado que lea todos estos datos, e imprima el número de censo de todas las jóvenes solteras que estén entre 16 y 21 años.

7.- Diseñe un pseudocódigo que lea el valor de un ángulo expresado en radianes y calcule e imprima el valor del seno de dicho ángulo. Se leerá también el número de términos de la serie.

SEN(X) = X - (X 3 / 3 ! ) + (X 5 / 5 ! ) - (X7/ 7!) + .....

8.-Un jeep puede viajar 500 km con un tanque lleno de gasolina. Desde una posición inicial, conteniendo ‘n’ tanques de gasolina el mismo jeep puede viajar:

L = 500 ( 1 + 1/3 + 1/5 + ...+ 1 / (2n -1) ) km

Estableciendo economía de combustible en una ruta. Diseñe un pseudocódigo que calcule el valor de ‘L’ dado ‘ n ‘ .

9.- Se ofrece un trabajo que pague un centavo en la primera semana, pero dobla su salario cada semana, es decir, $.01 la primera semana; $.02 la segunda semana; $0.4 la tercera semana; ... etc. Hasta $(2n-1)/100 la n- ésima . Diseñar el pseudocódigo que determine ( y escriba ) el salario por cada semana y el salario pagado hasta la fecha por espacio de 50 semanas.

10.-Diseñe un pseudocódigo que calcule e imprima el pago de 102 trabajadores que laboran en la Cía. GACMAN. Los datos que se leerán serán los siguientes:

a) Las horas trabajadas

b) El sueldo por hora

c) El tipo de trabajador (1.-obrero,2.-empleado)

Para calcular los pagos considerar lo siguiente:

- Los obreros pagan 10 % de impuesto

- Los empleados pagan 10 % de impuesto.

- Los trabajadores (obreros y empleados) que reciban un pago menor de 100,000 pesos no pagan impuesto.

-Al final se deberá imprimir el total a pagar a los trabajadores y a los empleados.

11.- Diseñar un pseudocódigo que convierta un número del sistema decimal a:

a) sistema binario b) sistema octal c) sistema hexadecimal. Según se elija.

12.- Un objeto es dejado caer a una altura de 100 mts. Diseñe un pseudocódigo que imprima cada décima de segundo la distancia entre el objeto y el suelo y al final imprima el tiempo necesario en décimas de segundo para que el objeto toque el suelo.

13.- La Cía. Automovilística Mexicana, S.A. de C.V premia anualmente a sus mejores vendedores de acuerdo a la siguiente tabla:

Si vendió Le corresponde de Comisión

sobre ventas totales

1,000,000 <= v < 3,000,000 3%

3,000,000 <= v < 5,000,000 4%

5,000,000 <= v < 7,000,000 5%

7,000,000 <= v 6%

Diseñar un pseudocódigo que lea las ventas de 100 vendedores y que escriba la comisión anual que le corresponda a cada vendedor. Suponer que nadie vende más de 10,000,000 al año.

14.- Diseñe un pseudocódigo que imprima la fecha en palabras a partir de la representación siguiente: S,DD,MM, AA.

En donde:

S = Día de la semana, 1 a 7 ( 1 = lunes; 2 = martes; etc..);

DD = Día del mes, 1 a 30 ó 31, según el mes. Fijar el mes de febrero con 28 días;

AA = Dos últimas cifras del año.

15.- Un grupo de 100 estudiantes presentan un exámen de Física. Diseñe un diagrama que lea por cada estudiante la calificación obtenida y calcule e imprima:

A.- La cantidad de estudiantes que obtuvieron una calificación menor a 50.

B.- La cantidad de estudiantes que obtuvieron una calificación de 50 o más pero menor que 80.

C.- La cantidad de estudiantes que obtuvieron una calificación de 70 o más pero menor que 80.

D. La cantidad de estudiantes que obtuvieron una calificación de 80 o más.

16.- Un avión que viaja 800 Km/hr. Dispara un proyectil auto impulsado, en el momento del disparo, el avión hace un giro de 90 0 y acelera a 20 mtrs/seg2. El proyectil sigue su curso, acelerando a 10 mtrs. /seg2.

Diseñe un pseudocódigo que escriba cada segundo, la distancia que separa al avión del proyectil, hasta que estén a 10,000 mtrs. o más.

17.- Una pizzería, vende sus pizzas en tres tamaños:

pequeña (10 pulg. De diámetro); mediana (12 pulg. De diámetro); y grandes (16 pulg. De diámetro); Una pizza puede ser sencilla (con sólo salsa y carne), o con ingredientes extras, tales como pepinillos, champiñones o cebollas

Los propietarios desean desarrollar un programa que calcule el precio de venta de una pizza, dándole el tamaño y el número de ingredientes extras. El precio de venta será 1.5 veces el costo total, que viene determinado por el área de la pizza, más el número de ingredientes.

En particular el costo total se calcula sumando:

- un costo fijo de preparación

- un costo base variable que es proporcional al tamaño de la pizza

- un costo adicional por cada ingrediente extra. Por simplicidad se supone que cada ingrediente extra tiene el mismo costo por unidad de área.

18.- Diseñar un pseudocódigo que calcule el promedio ponderado para alumno del ITT. El cálculo se hace de la siguiente forma:

- Se multiplica cada calificación por los créditos de cada materia

- El resultado anterior se suma con los resultados de todas las materias, por separado se suman los créditos de cada materia y finalmente se divide la suma de todas las materias por sus respectivos créditos, entre la suma de todos los créditos.

19.- Calcule la suma de los términos de la serie FIBONACCI cuyos valores se encuentran entre 100 y 10,000.

20.- Calcule exactamente el número de días vividos por una persona hasta la fecha. Contemplar los años bisiestos.

**CAPITULO VI.**

**ARREGLOS**

6.1 Vectores

6.2 Matrices

**Objetivo educacional:**

El alumno:

* Será capaz de utilizar los datos de tipo arreglo para plantear la solución de problemas que requieran de esta estructura.

**Arreglo:** Un *Arreglo* es una estructura de datos que almacena bajo el mismo nombre (variable) a una colección de datos del mismo tipo.

Los arreglos se caracterizan por:

* Almacenan los elementos en posiciones contiguas de memoria
* Tienen un mismo nombre de variable que representa a todos los elementos. Para hacer referencia a esos elementos es necesario utilizar un índice que especifica el lugar que ocupa cada elemento dentro del archivo.

Unidimensionales (Vectores)

Tipo de

Arreglos

Bidimensionales (Matrices)

***6.1. Vectores***

Es un arreglo de “N” elementos organizados en una dimensión donde “N” recibe el nombre de longitud o tamaño del vector. Para hacer referencia a un elemento del vector se usa el nombre del mismo, seguido del índice (entre corchetes), el cual indica una posición en particular del vector. Por ejemplo:

Vec[x]

Donde:

Vec………… Nombre del arreglo

x…………… Numero de datos que constituyen el arreglo

## **Representación gráfica de un vector**

Vec[1]

Vec[2]

Vec[3]

Vec[4]

|  |
| --- |
| 7 |
| 8 |
| 9 |
| 10 |

## **Llenado de un Vector**

* Hacer para I = 1 a 10

Leer vec[I]

Fin-para

* Hacer mientras I <= 10

Leer vec[I]

Fin-mientras

* I=1

Repetir

Leer vec[I]

I = I + 1

Hasta-que I>10

# Problemas propuestos

1. Calcular el promedio de 50 valores almacenados en un vector. Determinar además cuantos son mayores que el promedio, imprimir el promedio, el número de datos mayores que el promedio y una lista de valores mayores que el promedio.
2. Llenar dos vectores A y B de 45 elementos cada uno, sumar el elemento uno del vector A con el elemento uno del vector B y así sucesivamente hasta 45, almacenar el resultado en un vector C, e imprimir el vector resultante.
3. Llenar un vector de 20 elementos, imprimir la posición y el valor del elemento mayor almacenado en el vector. Suponga que todos los elementos del vector son diferentes.
4. Almacenar 500 números en un vector, elevar al cuadrado cada valor almacenado en el vector, almacenar el resultado en otro vector. Imprimir el vector original y el vector resultante.
5. Almacenar 300 números en un vector, imprimir cuantos son ceros, cuantos son negativos, cuantos positivos. Imprimir además la suma de los negativos y la suma de los positivos.
6. Almacenar 150 números en un vector, almacenarlos en otro vector en orden inverso al vector original e imprimir el vector resultante.
7. Se tienen almacenados en la memoria dos vectores M y N de cien elementos cada uno. Hacer un algoritmo que escriba la palabra “Iguales” si ambos vectores son iguales y “Diferentes” si no lo son.

Serán iguales cuando en la misma posición de ambos vectores se tenga el mismo valor para todos los elementos.

1. Se tiene el vector A con 100 elementos almacenados. Diseñe un algoritmo que escriba “SI” si el vector esta ordenado ascendentemente o “NO” si el vector no está ordenado
2. Diseñe un algoritmo que lea un numero cualquiera y lo busque en el vector X, el cual tiene almacenados 80 elementos. Escribir la posición donde se encuentra almacenado el numero en el vector o el mensaje “NO” si no lo encuentra. Búsqueda secuencial.
3. Diseñe un algoritmo que lea dos vectores A y B de 20 elementos cada uno y multiplique el primer elemento de A con el último elemento de B y luego el segundo elemento de A por el diecinueveavo elemento de B y así sucesivamente hasta llegar al veinteavo elemento de A por el primer elemento de B. El resultado de la multiplicación almacenarlo en un vector C.
4. Diseñe un algoritmo que almacene en un vector llamado FIB [100] los 100 primeros números de la serie Fibonacci.

***6.2 Matriz***

Es un arreglo de M \* N elementos organizados en dos dimensiones donde “M” es el número de filas o reglones y “N” el número de columnas.

Para representar una matriz se necesita un nombre de matriz se necesita un nombre de matriz acompañado de dos índices.

Mat [R,C]

Donde R indica el renglón y C indica la columna, donde se encuentra almacenado el dato.

## **Representación gráfica de una matriz**

Fila o Renglón

[R]

## ***Mat [R,C]***

|  |  |  |  |
| --- | --- | --- | --- |
| 1,1 | 1,2 | 1,3 | 1,4 |
| 2,1 | 2,2 | 2,3 | 2.4 |
| 3,1 | 3,2 | 3,3 | 3,4 |
| 4,1 | 4,2 | 4,3 | 4,4 |

Columna

[C]

### **Llenado de una matriz**

* ***Por renglones***

Hacer para R = 1 a 5

Hacer para C = 1 a 5

Leer Mat [R,C]

Fin-para

Fin-para

* ***Por columnas***

Hacer para C = 1 a 5

Hacer para R = 1 a 5

Leer Mat [R,C]

Fin-para

Fin-para

**Nota:** Para hacer el llenado de una matriz se deben de usar dos variables para los índices y se utilizan 2 ciclos uno para los renglones y otro para las columnas; a estos ciclos se les llama ciclos anidados (un ciclo dentro de otro ciclo).

# Problemas propuestos

1. Hacer un algoritmo que almacene números en una matriz de 5 \* 6. Imprimir la suma de los números almacenados en la matriz.
2. Hacer un algoritmo que llene una matriz de 10 \* 10 y determine la posición [renglón, columna] del número mayor almacenado en la matriz. Los números son diferentes.
3. Hacer un algoritmo que llene una matriz de 7 \* 7. Calcular la suma de cada renglón y almacenarla en un vector, la suma de cada columna y almacenarla en otro vector.
4. Hacer un algoritmo que llene una matriz de 20 \* 20. Sumar las columnas e imprimir que columna tuvo la máxima suma y la suma de esa columna.
5. Hacer un algoritmo que llene una matriz de 5 \* 5 y que almacene la diagonal principal en un vector. Imprimir el vector resultante.
6. Hacer un algoritmo que llene una matriz de 10 \* 10 y que almacene en la diagonal principal unos y en las demás posiciones ceros.
7. Hacer un algoritmo que llene una matriz de 6 \* 8 y que almacene toda la matriz en un vector. Imprimir el vector resultante.
8. Hacer un algoritmo que llene una matriz de 8 \* 8, que almacene la suma de los renglones y la suma de las columnas en un vector. Imprimir el vector resultante.
9. Hacer un algoritmo que llene una matriz de 5 \* 6 y que imprima cuantos de los números almacenados son ceros, cuantos son positivos y cuantos son negativos.
10. Diseñe un pseudocódigo que escriba el número de la hilera cuya suma sea mayor que las demás hileras. Suponga que todas las hileras suman diferente cantidad.

11) El dueño de una cadena de tiendas de artículos deportivos desea controlar sus ventas por medio de una computadora. Los datos de entrada son:

a) El número de la tienda (1 a 50)

b) Un número que indica el deporte del articulo (1 a 20)

c) El costo del artículo.

Hacer un pseudocódigo que escriba al final del día lo siguiente

1. Las ventas totales en el día para cada tienda

2.Las ventas totales para cada uno de los deportes.

3. Las ventas totales de todas las tiendas.

12) El departamento de policía de la ciudad de Tuxtepec ha acumulado información referente a las infracciones de los límites de velocidad durante un determinado periodo de tiempo. El departamento ha dividido la ciudad en cuatro cuadrantes y desea realizar una estadística de las infracciones a los límites de velocidad en cada uno de ellos. Para cada infracción se ha preparado una tarjeta que contiene la siguiente información:

- número de registro del vehículo;

- cuadrante en el que se produjo la infracción

- límite de velocidad en milla por hora

Diseñe un diagrama para producir 2 informes; el 1o. Que contiene una lista de la multa de velocidad recolectadas, donde la multa se calcula como la suma del costo de la corte ($20,000) mas $ 1,250 por cada mph que exceda la velocidad limite. Prepare una tabla con los siguientes resultados:

INFRACCIONES A LOS LIMITES DE VELOCIDAD

Registro del Velocidad Velocidad Multa

vehículo registrada (MPH) limite

Este informe debe ser seguido de un segundo en el cual se proporcione un análisis de las infracciones por cuadrante. Para cada uno de los 4 cuadrantes mencionados,

debe darse el número de infracciones y la multa promedio.

13) Se tiene almacenada la matriz M (50,5) la cual contiene la información sobre las calificaciones de la materia de LENGUAJES ALGORITMICOS. Diseñe un pseudocódigo que imprima:

A). - Cantidad de alumnos que aprobaron la materia.

B). - Cantidad de alumnos que tienen derecho a nivelación.

C). - El (o los) numero (s) de control de lo(s) alumno(s) que haya (n) obtenido la máxima calificación final.

**CAPITULO VII.**

***MANEJO DE MÓDULOS***

7.1 Definición

7.2 Función

7.3 Manipulación

**Objetivo educacional:**

El alumno:

* Identificará y aplicará los datos de tipo cadena de caracteres (string) para la solución de problemas de tipo administrativo.

***7.1 Definición, Función y Manipulación***

Un problema complejo se puede dividir en pequeños subproblemas más sencillos. Estos subproblemas se conocen como *“Módulos”* y su complementación en un lenguaje se llama subprograma (procedimientos y funciones).

Un subprograma realiza las mismas acciones que un programa, sin embargo, un subprograma lo utiliza solamente un programa para un propósito específico.

Un subprograma recibe datos de un programa y le devuelve resultados (el programa “llama” o “invoca” al subprograma, este ejecuta una tarea específica y devuelve el “control” al programa que lo llamo).

Funciones

Tipos de

Módulos

Procedimientos

***Función:*** Una función en matemáticas, es una operación que toma un o más valores (argumentos) y devuelve un resultado (valor de la función para los argumentos dados). Por ejemplo:

F(X) = X / (1+X2)

Donde:

F …………. Nombre de la función

X …………. Es el argumento (también conocido como parámetro formal)

***Definición de funciones:*** Una definición de función se presenta de la siguiente manera:

Función nombre función (p1, p2, …, pn)

**Inicio**

Bloque de instrucciones

**Fin**

Donde:

Función …………… Es la palabra clave que nos indica una definición de función.

Nombre función ….. Es el identificador con el cual se reconoce a la función en el cuerpo del algoritmo principal.

P1, p2,…,pn ……... Es el grupo de parámetros que define a la función.

***Llamado a una función***

Cuando definimos una función solo le indicamos al algoritmo que esta función existe, pero una definición de función no implica la realización de las instrucciones que la constituyen. Para hacer uso de una función, el algoritmo principal la debe llamar. Por ejemplo:

Función F(X)

**Inicio**

F = X /(1 + X^2)

**Fin**

**Inicio**

Imprimir “Este es el algoritmo principal”

Leer N

R = F(N) llamado de la función

Imprimir “El resultado de la función es:”,R

**Fin**

***Problemas Propuestos***

1. Diseñe un algoritmo que llene una matriz de 10 \* 10 y determine:
2. El número mayor almacenado en la matriz
3. El número mayor almacenado en cada renglón
4. La columna que tuvo la máxima suma
5. El renglón que tuvo la máxima suma

Diseña una función para cada inciso.

1. Diseñe un algoritmo que lea un número y mediante una función regrese el valor de 1 si el número es positivo y -1 si es negativo).

**CAPITULO VIII.**

**CONCEPTOS BÁSICOS Y METODOLOGÍA PARA VISUAL BASIC.**

8.1 Características de Visual Basic

8.2 Descripción de ventanas principales

8.3 Las partes del entorno de Visual Basic

8.4 Variable

8.5 Tipos de variables

8.6 Expresiones lógicas

8.7 Operadores de Visual Basic

8.8 Operadores relacionales

8.9 Sentencias

8.10 Bucle por contador

8.11 Bucles por condición

8.12 Instrucciones de control de programa

8.13 Teclas de método abreviado

8.14 Eventos del menú

8.15 MSGBOX

8.16 Los controles

8.17 Controles básicos

**Objetivo educacional:**

El alumno:

* Conocerá la terminología relacionada con Visual Basic.
* Conocerá la metodología en cada una de sus etapas.
* Tendrá conceptos básicos acerca de la elaboración de programas.

## **VISUAL BASIC**

Es un lenguaje de programación que se ha diseñado para facilitar el desarrollo de aplicaciones en un entorno grafico (GUI-GRAPHICAL USER INTERFACE) Como Windows 98, Windows NT o superior.

***8.1 Características de Visual Basic***

Diseñador de entorno de datos: Es posible generar, de manera automática, conectividad entre controles y datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Los Objetos Actives son una nueva tecnología de acceso a datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Asistente para formularios: Sirve para generar de manera automática formularios que administran registros de tablas o consultas pertenecientes a una base de datos, hoja de cálculo u objeto (ADO-ACTIVE DATA OBJECT)

Asistente para barras de herramientas es factible incluir barras de herramientas es factible incluir barra de herramientas personalizada, donde el usuario selecciona los botones que desea visualizar durante la ejecución.

**En las aplicaciones HTML:** Se combinan instrucciones de Visual Basic con código HTML para controlar los eventos que se realizan con frecuencia en una página web.

La Ventana de Vista de datos proporciona acceso a la estructura de una base de datos. Desde este también acceso al Diseñador de Consultas y diseñador de Base de datos para administrar y registros.

# 8.2 Descripción de ventanas principales

Ejecutamos Visual Basic (VB) y al comenzar podemos ver la siguiente ventana que nos permitirá saber que en que formato deseamos el programa:



Está nos está mostrando las diferentes formas para comenzar, hay 3 formas, la primera **Nuevo** es iniciando un Nuevo Proyecto, seleccionando uno tipo de proyecto, la segunda **Existente** es abriendo un proyecto ya existente, el archivo tiene que contener la extensión \*.vbp, \*.mak o \*.vbg, y por último la tercera **Recientes** es seleccionando un proyecto de la lista de archivos recientemente abiertos (VB se encarga automáticamente de mostrar una lista de los últimos proyectos abiertos).

Para continuar se elige EXE estándar y luego Abrir, ahora se explicará la Ventana principal de Visual Basic.

***8.3 Las partes del entorno de Visual Basic***

**Barra de título:** muestra el nombre del proyecto y del formulario q se está diseñando actualmente

**Barra de menús:** agrupa los menús despegables que contienes todas las operaciones que pueden llevarse a cabo con Visual Basic 6.0.

**Barra de herramientas estándar:** contienen los botones que se utilizan con mayor frecuencia cuando se trabaja con un proyecto. Simplifica la elección de opciones de los menús Archivo, Edición, Ver y Ejecutar; además, en el área derecha presenta la ubicación (coordenadas) y el tamaño del objeto seleccionado

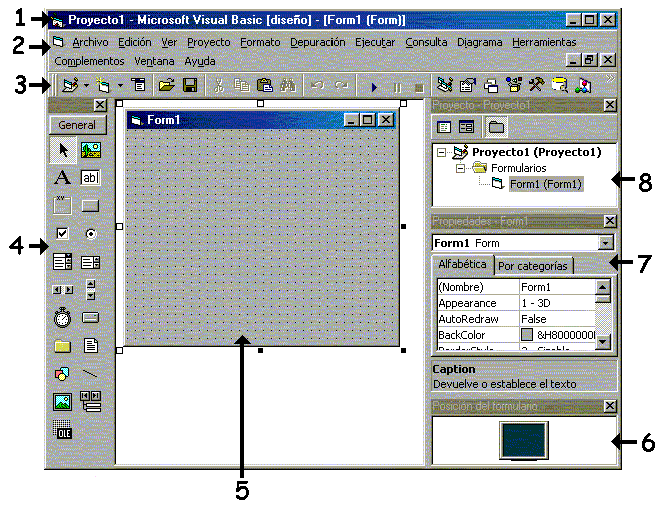
**Ventana de formulario:** es el área donde se diseña la interfaz gráfica, es decir, es donde se inserta electo gráfico, como botones, imágenes, casilla de verificación, cuadros de listas, etc.

Cuadro de herramientas: presenta todos los controles necesarios para diseñar una aplicación, como cuadros de texto, etiquetas, cuadros de listas, botones de comandos, etc.

**Ventana de proyecto:** muestra los elementos involucrados en el proyecto, como formularios, módulos, controles oxc, etc. Cada elemento puede seleccionarse en forma independiente para su edición.

Ventana de posición del formulario: muestra la ubicación que tendrá el formulario en la pantalla, cuando ejecute la aplicación. Esta ubicación puede cambiarse si se hace clic con el botón izquierdo del mouse.

**La Ventana propiedades:** muestra todas las propiedades del control actualmente seleccionado, en este caso muestra las propiedades del Form1, luego podemos ver que abajo dice "Form1 Form", lo que está en negrita es el nombre del objeto, y lo que le sigue es el tipo de objeto, en este caso es un Formulario (Form)



**1.-** Barra de título, en esta se muestra el nombre del proyecto actual en el que se está trabajando, luego viene el nombre del programa, continuado a "Microsoft Visual Basic viene el estado en el que está el proyecto, hay 3 tipos de estado en **[Diseño]** cuando se está escribiendo el código, **[Ejecución]** el programa se encuentra en ejecución, o en estado de **[Interrupción]** que es cuando en estado de ejecución se produjo un error, por último en la barra de título se puede visualizar el nombre del Form (Formulario) actual en que se está trabajando y como se está trabajando **[Form1 [Form]]** cuando se trabaja en el diseño del Formulario o **[Form1 [Código]]** cuando se trabaja en el código para ese formulario.

**2.-** En la barra de menús se encuentran todas las opciones para manejar Visual Basic, a medida que se vayan utilizando se irán explicando.

**3.-** La barra de herramientas, contiene varios accesos directos a los menús, para agilizar el manejo de Visual Basic.

**4.-** Se le llama Cuadro de herramientas y sirve para colocar objetos en la Ventana de Diseño, como Cuadros de textos, Botones, Imágenes, y otros. Esto funciona muy fácil ya que seleccionamos por ejemplo un CommandButton (botón), y vamos a la ventana diseño y lo creamos haciendo Clic con el Mouse (¡sin soltarlo!) y lo arrastramos hasta el tamaño que deseemos, en ese momento soltamos el botón. Y listo ya tienes un botón estilo Windows para usar.

**5.-** La Ventana diseño, es aquí donde se le da la apariencia al programa, es decir lo que el usuario va a ver. Los puntillos del fondo son utilizados con el fin de alinear más fácil los objetos en el Formulario(Ventana).

**6.-** En la ventana Posición del Formulario, se puede ver un monitor y dentro de él una ventanita, esto sirve para definir la posición de la ventana cuando se inicie el programa, es decir donde se va a mostrar. Pruebe hacer clic sobre la ventana (¡sin soltar el botón!) y arrastre la ventana dentro del monitor, luego suelte el botón del Mouse. Y cuando se ejecute la aplicación la ventana aparecerá en esa posición.

**7.-** La Ventana propiedades muestra todas las propiedades del control actualmente seleccionado, en este caso muestra las propiedades del Form1, luego podemos ver que abajo dice "**Form1** Form", lo que está en negrita es el nombre del objeto, y lo que le sigue es el tipo de objeto, en este caso es un Formulario (Form).

**8.-** En la ventana Posición del Formulario, se puede ver un monitor y dentro de él una ventanita, esto sirve para definir la posición de la ventana cuando se inicie el programa, es decir donde se va a mostrar. Pruebe hacer clic sobre la ventana (¡sin soltar el botón!) y arrastre la ventana dentro del monitor, luego suelte el botón del Mouse. Y cuando se ejecute la aplicación la ventana aparecerá en esa posición.

**8.-** El Explorador de proyectos es donde visualizamos todas los Formularios(Ventanas), Módulos, Clases, entre otros del proyecto actualmente abierto, es decir de la aplicación que se está creando, en este caso hay un solo Formulario llamado Form1, lo que aparece dentro de paréntesis es el nombre de archivo. Y este Formulario se encuentra dentro de una carpeta llamada "Formularios", por ser que VB clasifica los archivos por tipos, cuando use otros tipos de archivos además de Formularios podrá visualizar otras carpetas.

***8.4 Variable***

Basic, desde siempre, al contrario de otros sistemas de programación, no exigió la definición previa de una variable. Una variable, como Vd. seguro que conoce, es un nombre que en el programa le asignamos a un dato. Ese dato podrá cambiar.

**Dim:** Al declarar una variable con esta palabra estamos diciendo que la variable sea local al ámbito en que se declara. Puede ser dentro de un procedimiento o dentro de un formulario, de esta forma no sería accesible desde los demás procedimientos o formularios.

**Public:** Las variables declaradas serán públicas y podrán estar accesibles desde todos los formularios de la aplicación. Para conseguirlo tendremos que declararlas en un módulo de código, no en la sección declarations de cualquier formulario de los que conste la aplicación. Para crear un módulo de código en el menú principal de Visual Basic marcamos en INSERT/MODULE y aparecerá junto a los demás formularios de la ventana de proyecto, aunque con un icono distinto indicando que se trata de un módulo de código.

**Static:** Con esta forma de declarar variables conseguiremos que las variables locales no se creen y se destruyan al entrar y salir de los procedimientos donde fueron declaradas, sino que se mantenga su valor durante todo el periodo de ejecución de la aplicación. De esta forma a entrar en algún procedimiento las variables recuerdan el valor que tenían cuando se salió de él.

## ***8.5 Tipos de variables***

|  |  |
| --- | --- |
| **TIPO** | **COMENTARIO** |
| BOOLEAN | Sólo admite 2 valores TRUE o FALSE |
| BYTE | admite valores entre 0 y 255 |
| INTEGER | admite valores entre -32768 y 32767 |
| LONG | admite valores entre -2.147.483.648 y 2.147.483.647 |
| SINGLE | admite valores decimales con precisión simple |
| DOUBLE | admite valores decimales de doble precisión |
| CURRENCY | válido para valores de tipo moneda |
| STRING | cadenas de caracteres |
| DATE | fechas, permite operar con ellas |
|  |  |

**Constante:** Declaración de constantes que pueden ser usadas en cualquier punto en lugar de su valor, permitiendo cambiarlo cuando sea necesario, sin tener que cambiarlo en todos los sitios en que se utiliza. La expresión no puede utilizar llamadas a funciones, pues la constante se calcula en tiempo de compilación, no en tiempo de ejecución.

***8.6 Expresiones lógicas***

Existen los llamados **operadores lógicos**, que nos permiten establecer condiciones que dependan de más de un criterio de selección. Todos los operadores hacen que la expresión en la que se encuentren se evalúa a verdadero o falso, sin posibilidad de cualquier otro valor.

Estas expresiones son:

**And:** exp1 And exp2, donde se evalúa a verdadero s&oacutelo en el caso que el exp1 como exp2 se evalúen a verdadero. En cualquier otro caso se evalúa a falso.

**Or:** exp1 Or exp2, donde se evalúa a verdadero cuando alguna de las expresiones exp1 o exp2 se evalúa a verdadero.   
**Not:** Not exp1. aquí se evalúa a verdadero si exp1 es falso y se evalúa a falso si exp1 es verdadero.   
**Xor:** exp1 Xor exp2, se evalúa a verdadero sólo en el caso de que una y sólo una de las expresiones exp1 o exp2 se evalúa a verdadero.

***8.7 Operadores de Visual Basic***

En Visual Basic existe un gran número de operadores que se pueden utilizar para crear fórmulas.   
Los operadores más utilizados en una aplicación de Visual Basic son los siguientes:

|  |  |
| --- | --- |
| **Operador** | **Operación que realiza** |
| + | Suma / Concatenación de cadenas de caracteres |
| - | Resta |
| \* | Multiplicación |
| / | División |
| \ | División entera |
| Mod | Resto de la división entera |
| ^ | Exponenciación |
| & | Concatenación de cadena de caracteres |

## ***8.8 Operadores relacionales***

Los operadores relaciónales que reconoce vcpp son:

|  |  |
| --- | --- |
| **Operador** | **Significado** |
| == | Igual que |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| != | No es igual que o es diferente que |

***8.9 Sentencias***

**SENTENCIA DIM**

Es la forma más común de declarar una variable como

Privada. Puede emplearse en un Procedimiento, Función, Formulario o Módulo. La sintaxis es de la siguiente forma:

**Dim**  nombrevariable **As Integer** (o el tipo que sea)

Declarando una variable con la sentencia DIM, en un ***formulario***, ***Función***, ***procedimiento*** o ***módulo***, el entorno de la variable será el explicado anteriormente para una variable declarada como Privada. Es decir, esa variable no sale del formulario, procedimiento ó módulo donde se declaró. Cada vez que entremos al formulario, procedimiento o módulo, esa variable tomará el valor cero (si es numérica) o nulo (si es string).

 Resumimos la forma de declarar una variable:

*En un Procedimiento* (La variable no puede usarse fuera de esta Procedimiento)

**Dim** Variable **As** Tipovariable

En un Formulario (En su sección de declaraciones)

***SENTENCIAS CONDICIONALES.***

Llamamos sentencias condicionales a aquellas que se realizan si se cumple una determinada condición. Son las sentencias por las que empieza cualquier texto de Basic, y este no va ser menos.

 La sentencia condicional más usada es:

**Si** se cumple una condición **Entonces**

  Realiza estas instrucciones

**Si no se cumple**

  Realiza estas otras instrucciones

**Fin de la sentencia.**

Así de fácil es programar en Basic. Lo que ocurre es que esta herramienta habla inglés, y lo descrito anteriormente toma la forma:

**If condición** Then

Instrucciones

**Else**

Otras instrucciones

**End If**

**LA SENTENCIA SELECT**

La sentencia SELECT "selecciona" los campos que conformarán la consulta, es decir, que establece los campos que se visualizarán o compondrán la consulta. El parámetro 'lista\_campo' está compuesto por uno o más nombres de campos, separados por comas, pudiéndose especificar también el nombre de la tabla a la cual pertenecen, seguido de un punto y del nombre del campo correspondiente. Si el nombre del campo o de la tabla está compuesto de más de una palabra, este nombre ha de escribirse entre corchetes ([nombre]). Si se desea seleccionar todos los campos de una tabla, se puede utilizar el asterisco (\*) para indicarlo.

Una sentencia SELECT no puede escribirse sin la cláusula FROM. Una cláusula es una extensión de un mandato que complementa a una sentencia o instrucción, pudiendo complementar también a otras sentencias. Es, por decirlo así, un accesorio imprescindible en una determinada máquina, que puede también acoplarse a otras máquinas. En este caso, la cláusula FROM permite indicar en qué tablas o en qué consultas (queries) se encuentran los campos especificados en la sentencia SELECT. Estas tablas o consultas se separan por medio de comas (,), y, si sus nombres están compuestos por más de una palabra, éstos se escriben entre corchetes ([nombre]).

Tenemos aquí algunos ejemplos de mandatos SQL en la estructura SELECT...FROM...:  
SELECT nombre,apellidos FROM clientes;  
Selecciona los campos 'nombre' y 'apellidos' de la tabla 'clientes'.  
SELECT clientes.nombre, producto FROM clientes, productos;  
Selecciona el campo 'nombre' de la tabla 'clientes', y el campo 'producto' de la tabla productos.  
Hay que tener en cuenta que si dos tablas poseen el mismo nombre de campo (un 'nombre' de cliente y un 'nombre' de producto, hay que especificar también la tabla a la cual pertenece dicho campo, ya, que de lo contrario, seleccionaría ambos nombres).  
SELECT pedidos.\* FROM pedidos;  
Selecciona todos los campos de la tabla 'pedidos'.  
SELECT \* FROM pedidos;  
Selecciona todos los campos de la tabla 'pedidos'.  
SELECT nombre, apellidos, teléfono FROM clientes;  
Selecciona los campos 'nombre', 'apellidos' y 'teléfono' de la tabla 'clientes'. De esta manera obtenemos una agenda telefónica de nuestros clientes.  
SELECT [codigo postal] FROM [tabla morosos];  
Selecciona el campo 'codigo postal' de la tabla 'tabla morosos'.

### ***8.10 Bucle por contador***

Realiza el bucle tantas veces como le indiquemos. Por ejemplo, en este bucle nos va a presentar las 26 letras mayúsculas del alfabeto inglés

For N=65 To 90

  Label1.caption = Chr ( N )

  Next N

Este "programa" nos presentará en una caja (Label) los caracteres cuyo número ASCII vaya desde el 65 (A) al 90 (Z) Comenzará presentando el correspondiente al número 65, e irá presentando sucesivamente el 66, el 67, etc., hasta llegar al 90, donde se parará.

**8.11 Bucles por condición**

  Ejecuta las instrucciones del bucle mientras se cumple una condición

  X = 0

Do While X < 1000

X = X + 1

Loop

El programa toma una variable ( X ) que previamente tuvimos la curiosidad de ponerla a cero, e incrementa su valor una unidad. Analiza si el valor de X es menor que 1000, y si es cierto, vuelve a realizar el bucle. Así hasta que X ya no sea menor que 1000. Al dejar de cumplirse que X sea menor que 1000, sale del bucle. Acabamos de realizar un temporizador, y también de exponer las sentencias condicionales y los bucles, inicio de cualquier curso de Basic. Como final de lección, se propone un problema. Con el primer bucle, donde visualizábamos los caracteres A a la Z, posiblemente no nos diese tiempo de ver cada una de las letras que iban apareciendo en la pantalla, en la etiqueta Label1, dado que cambiaría con mucha velocidad, y solamente veríamos la Z, que es donde se detuvo el programa. Podemos poner un temporizador cada vez que presente una letra, y así nos dará tiempo a verla. Con los dos bucles vistos anteriormente ya podemos hacerlo. Si lo que queremos es que el programa se ejecute mientras no se cumpla una determinada condición, la sentencia será:

  X = 0

Do Until X > 1000

X = X + 1

Loop

 Observe que la diferencia entre una y otra es la condición, While para indicar Mientras se cumpla que ... y Until para indicar Mientras no se cumpla que ....

 Utilizaremos la condición de Whilepara: (Mientras se cumpla la condición)

  For N=65 To 90

Label1.caption = Chr ( N )

Label1.RefreshN ' Refresca la etiqueta

X = 0

Do While X < 1000

X = X + 1

Loop

Next N

La instrucción de la cláusula Else, sólo se ejecuta cuando el resto de condiciones han fracasado. Cualquiera de los bloques de instrucción puede contener un número arbitrario de instrucciones, incluida la posibilidad de que existan otras instrucciones IF-THEN-ELSE. Se pueden colocar todas las sentencias de código que queramos en cada uno de los bloques de sentencias, siempre que cada sentencia vaya en una línea distinta.

Un ciclo similar a el 'Do While' es el 'Loop While'. La principal diferencia es que el ciclo Do While la condición para poder continuar esta al principio (primero sea la condición y luego ejecuta el código que se desea repetir), mientras que en el Loop While está al final, esto es después de que ejecuta el código checa si debe continuar. Esto significa que un ciclo Do While, puede nunca ser ejecutado, mientras que un ciclo Loop While siempre al menos se ejecutará una vez.

Cuando existe un gran número de instrucciones a evaluar es aconsejable utilizar otra estructura de decisión, como puede ser la instrucción **Select Case**. Esta instrucción no da más potencia al lenguaje, pero hace que el código sea más legible y eficiente. Esta sentencia permite realizar operaciones diferentes dependiendo del valor de una variable.

La estructura de repetición **For...Next** es adecuada cuando conocemos el número de veces que debe repetirse un conjunto de instrucciones y deseamos reducir la cantidad de código escrito.

Otra estructura de repetición es **Do...Loop**. Esta estructura se utiliza cuando desconocemos cuántas veces se ha de ejecutar el bucle.

Si utilizamos **While** el bucle se repite mientras la condición se cumpla y si utilizamos **Until** el bucle se repetirá hasta que la condición dé valor verdadero. Con While, la condición es comprobada al principio del bucle, por lo que, si no se cumple al iniciarse, el cuerpo del bucle no se ejecutará. Con Until se comprueba a la salida del bucle, por lo que por lo menos, una vez se ejecutará el bucle.

Los siguientes comandos pueden ser usados para repetir varias veces la ejecución de una sección de código.

Ciclo Do (Sintaxis):

Do While *condición*

*líneas de código*

Loop

Ciclo While (Sintaxis):

Do

*líneas de código*

Loop While *condición*

Los ciclos For si los programadores saben cuántas veces un cliclo debe ser ejecutado.

Ciclo For (Sintaxis):

For *contador* = *valor\_inicial* To *valor\_final* [Step *incremento*]

*líneas de código*

Next [*contador*]

If estructura de decisisón

If en Línea simple

End If

El If de multi-línea necesita una expresión 'End If' para definir el final del bloque *If*.

Si cualquiera de las dos condiciones es falsa, el programa ejecutará el código que está en el bloque 'Else'.

El comando 'ElseIf' puede usarse para checar una segunda condición en caso de que la primera sea falsa.

Select' define el final del bloque de casos. Si deseamos que presente un mensaje si se introduce un valor no definido en el bloque de casos,

El incremento en Step como parte del ciclo, es usado para decirle a VB cuanto debe aumentar en cada ciclo la variable de control. Introduce el siguiente código y observa que valores se imprimen.:

For test = 1 To 10 Step 2

Print test

Next test

Una de las más poderosas características de cualquier computador es la capacidad que tiene de tomar decisiones.

Es decir, al comparar dos alternativas diferentes el computador puede tomar una decisión basándose en la evaluación que hace de alguna condición.

ejemplo de instrucciones condicionales:

si sueldo > 3000

desplegar rico

si no

desplegar pobre

Fin-si

si sexo = 'm'

imprime mujer

si no

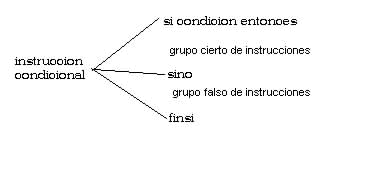
imprime hombre

Fin-si

De los ejemplos observar que los caminos a seguir por el computador dependerán de la evaluación que el computador hace con y de la condición.

Todo lenguaje de programación debe tener instrucciones que permitan formar condiciones e instrucciones que pueden evaluar esas condiciones.

Pero recordar que lenguajes modernos y orientados a clientes-servidores de igual forma tienen componentes que permiten del mismo modo al usuario tomar decisiones incluso directamente en pantalla, es decir también existen los objetos, controles o componentes de selección y decisión en HTML.

El formato general de una instrucción condicional es:

Como se observa, son cuatro partes bien diferenciadas entre sí;

1. La propia instrucción condicional en si
2. La condición
3. El grupo cierto de instrucciones
4. El grupo falso de instrucciones

Cuando el computador evalúa una condición el resultado de esa evaluación solo es evaluado de dos maneras o la condición es CIERTA o la condición es FALSA.

Esto dependerá del valor que tenga asignado o que se haya capturado para la variable que está en la condición, por ejemplo, si se capturo 6000 en sueldo en el ejemplo a) entonces el computador indicaría que la condición es CIERTA, pero en otro caso si a la variable sueldo primero se le asignó un valor de 250 entonces el computador indicaría que la condición es FALSA.

Ya dependiendo del resultado de la evaluación, el computador ejecuta las instrucciones contenidas en el grupo de cierto o falso respectivamente.

Empezaremos el análisis por la CONDICIÓN.

## ***8.12 Instrucciones de control de programa***

## **CONDICIONES SIMPLES**

En general todas las condiciones se forman con;

|  |  |  |
| --- | --- | --- |
| **Variables** | **Operadores Relaciónales** | **Constante o Variables** |
| sexo | = | m |
| sueldo | > | 300,000 |

Una condición simple se define como el conjunto de variables y/o constantes unidas por los llamados operadores relaciónales.

## **INSTRUCCIÓN IF**

Es la instrucción condicional mas usada en los diversos lenguajes de programación, su formato completo y de trabajo en vcpp es:

cargar o asignar la variable de condición;

if (condición)

{grupo cierto de instrucciones;}

else

{grupo falso de instrucciones;};

**Primus. -** Observar **donde van y donde no van** los puntos y comas;

**Secundus. -** La condición va entre paréntesis;

**Tertius. -** Si un if no ocupa un grupo falso de instrucciones entonces no se pone el else, y la llave antes del else si terminaría con punto y coma.

Ejemplo:

<HTML>

<FORM ACTION=/CGI-BIN/tusitio/PROG2.EXE METHOD=GET>

SUELDO:<INPUT TYPE=TEXT NAME=SUELDO><BR>

<INPUT TYPE=SUBMIT VALUE=OK>

</FORM></HTML>

#using <mscorlib.dll>

#using <System.dll>

#using <lcnet.dll>

using namespace System;

void main(){

//declarando variables

int sueldo;

// capturando,cargando y convirtiendo

//los datos de la forma a las variables

sueldo = Int32::Parse(lcnet::getparametro("SUELDO"));

//construyendo y desplegando la página de salida

Console::WriteLine("Content-Type:text/html\n");

if (sueldo >= 3000)

{Console::WriteLine(S"RICO");}

else

{Console::WriteLine(L"POBRE");};

}//fin main

**CONDICIONES COMPUESTAS**

En muchas ocasiones es necesario presentar más de una condición para su evaluación al computador.

Por ejemplo, que el computador muestre la boleta de un alumno si este estudia la carrera de medicina y su promedio de calificaciones es mayor de 70.

Una condición compuesta se define como dos o más condiciones simples unidas por los llamados operadores lógicos.

Los operadores lógicos que vcpp reconoce son;

|  |  |
| --- | --- |
| **OPERADOR** | **SIGNIFICADO** |
| && | "Y" LOGICO |
| || | "O" LOGICO |
| ! | "NO" NEGACION |

Ejemplo:

if ( (carrera=="informatica") && ( sexo=="m") ) etc, etc, etc.

**Notas:**

 Observar que cada condición simple lleva sus propios parentesis.

 Si la variable es de tipo string el dato va entre comillas ("), pero si la variable es de tipo char el dato va entre apostrofes (').

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "&&", las dos condiciones simples deben ser ciertas.

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "||", basta con que una de las condiciones simples sea cierta.

La cantidad total de casos posibles cuando se unen dos o más condiciones simples está dada por la relación donde n = cantidad de condiciones, la primera mitad de ellos empieza en cierto y la segunda mitad en falso.

Ejemplo, si formamos una condición compuesta con dos condiciones simples y el operador lógico "y", la cantidad total de casos posibles serian , y se puede construir la siguiente tabla de verdad.

**Tabla de verdad con "y"**

|  |  |  |
| --- | --- | --- |
| **IRA COND SIMPLE** | **2DA COND SIMPLE** | **EVALUACION** |
| C | C | C |
| C | F | F |
| F | C | F |
| F | F | F |

La evaluación final, se obtiene usando la regla anteriormente descrita para una condición compuesta, que contiene el operador "Y".

Esta tabla significa lo siguiente;

1.- Cualesquiera que sean la cantidad de datos procesados, siempre caerá en uno de estos cuatro casos generales.

La tabla de verdad para una condición compuesta con "Or" es la siguiente;

|  |  |  |
| --- | --- | --- |
| **IRA COND SIMPLE** | **2DA COND SIMPLE** | **EVALUACION** |
| C | C | C |
| C | F | C |
| F | C | C |
| F | F | F |

Construir una tabla de verdad para una condición compuesta de tres o más condiciones simples, es también tarea sencilla, solo recordar que;

**1.-** La cantidad posible de casos es posible, la mitad empiezan con Cierto y la otra mitad empiezan con Falso.

**2.-** Para evaluar esta condición triple primero se evalúan las dos primeras incluyendo su operador bajo las reglas ya descritas y luego se evalúa el resultado parcial contra la última condición y ultimo operador para obtener la evaluación final.

Ejemplo una condición compuesta de tres condiciones simples, donde el primer operador lógico es el "y" y el segundo operador lógico es el "O", daría la siguiente tabla de verdad.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ira cond** | **2da cond** | **Eval 1a Y 2a** | **3ra cond** | **Eval eval O 3ra** |
| C | C | C | C | C |
| C | C | C | F | C |
| C | F | F | C | C |
| C | F | F | F | F |
| F | C | F | C | C |
| F | C | F | F | F |
| F | F | F | C | C |
| F | F | F | F | F |

**CAPTION:**

Representa es texto que aparecerá en el menú.

**NAME:**

Es el nombre del control. Se puede anteponer el prefijo mnu más el texto del título del menú.   
**CHECKED:**

Establece si debe aparecer una marca de verificación a la izquierda del texto del objeto del menú. La marca la puedes utilizar cuando deseas informar de la opción que está activada en un momento determinado.   
**ENABLED:**

Indica si el objeto podrá recibir eventos del usuario. Si tiene el valor False el elemento aparecerá atenuado en el menú.

**VISIBLE:**

Indica si el objeto debe mostrarse o no, si se establece este valor a False, los elementos situados a la derecha ocupan el lugar del elemento no visible.

**WINDOWSLIST:**

Establece el valor que determina si un objeto de menú mantiene una lista de las ventanas MDI secundarias del formulario actual. Elementos del Menú:

Utilizando los botones de sangría se puede determinar el nivel del objeto menú que quieras seleccionar. Así, los títulos de menú se sitúan en el nivel superior, mostrándose en la lista inferior del editor de menús, lo más a la izquierda posible. En muchas aplicaciones se pueden ver unas barras horizontales en los menús desplegables que realizan la función de dividir en secciones dichos menús. Este elemento es conocido como un separador, y sirve para separar aquellos comandos del menú que tengan\_un\_motivo\_en\_común.   
Para crear un separador, basta con introducir un guión (-) en la propiedad Caption. No se puede crear un elemento en un nivel inferior inmediatamente después de un separador, es decir, no puedes tener como título de menú un separador.

***8.13 Teclas de método abreviado***

Al introducir el carácter & el menú está creando una tecla de acceso a dicho objeto. Esta tecla se representa subrayada en el texto de menú y representa la tecla del teclado que el usuario puede pulsar para ejecutar la acción. Dicha tecla se corresponde con la letra que sigue al caracter & en la propiedad Caption del objeto menú. Si se trata de un título de menú tienes que usar la tecla de acceso en combinación con la tecla [Alt.]. Una vez abierto un título de menú y desplegado sus elementos, para acceder rápidamente a uno de ellos, puedes usar la tecla de acceso directamente.

Tienes que tener cuidado en no utilizar las mismas teclas de acceso para elementos de un mismo nivel de menú ya que entonces no funcionarán. El editor de menú también admite la incorporación de teclas de método abreviado. Las abreviaturas del menú son combinaciones de teclas que se pueden utilizar en lugar de elegir el elemento del menú correspondiente. Para crear una tecla de método abreviado tienes que utilizar la propiedad Shortcut, a la que puedes acceder al crear el menú desde el editor de menús.

## ***8.14 Eventos del menú***

Cuando en tiempo de diseño seleccionamos la opción de menú de un formulario que estamos creando, Visual Basic muestra el procedimiento de evento click de dicho elemento del menú. En el caso del título del menú, el procedimiento click tiene como acción predeterminada la de mostrar los elementos de menú del nivel inferior, por lo que no será necesario tener que programar esta acción. Esto también se puede aplicar a los elementos de un menú que son a su vez títulos de submenús. En casi todos los casos tienes que utilizar una opción del menú para descargar el formulario o para finalizar la aplicación. La opción de texto salir suele ser el más apropiado. La instrucción Unload descarga de memoria el formulario que se especifique. Se puede usar de varias formas, por ejemplo, si estamos dentro de un Form llamado frmconsulta podremos salir de él de dos formas: Una es Unload Me o Unload frmconsulta. ***8.15 MSGBOX***

La síntaxis completa de la función **MsgBox** es:

**MsgBox(mensaje[, botones][, t&iacutetulo][, archivoAyuda, contexto])**

Los valores que pueden tomar el parámetro botones son:

|  |  |  |
| --- | --- | --- |
| **Constante** | **Valor** | **Descipción** |
| vbOKOnly | 0 | Muestra sólo el botón aceptar |
| vbOKCancel | 1 | Botones aceptar y cancelar |
| vbAbortRetryIgnore | 2 | Botones anular, reintentar e ignorar |
| vbYesNoCancel | 3 | Botones sí, no y cancelar |
| vbYesNo | 4 | Botones si y no |
| vbRetryCancel | 5 | Botones reintentar y cancelar |
| vbCritical | 16 | Muestra el icono de mensaje crítico |
| vbQuestion | 32 | Icono de interrogación |
| vbExclamation | 48 | Icono de exclamación |
| vbInformation | 64 | Icono de mensaje de información |
| vbAplicationModal | 0 | Cuadro de diálogo modal de la aplicación |
| vbSystemModal | 4096 | Cuadro de diálogo modal del sistema |

El parámetro botones es de valor numérico, y además de los botones que aparecen en le cuadro de diálogo, también puede indicar el icono que acompañe al mensaje y el tipo de diálogo modal que es.   
Cuando un cuadro de diálogo es modal de aplicacióen tienes que cerrarlo para interactuar con otra ventana de la misma aplicación. Cuando es un cuadro de diálogo modal del sistema, todas las aplicaciones que estén ejecutándose se suspenden hasta que el usuario responda al cuadro de diálogo. El cuadro **InputBox** se consigue a través de la función InputBox. Se utiliza cuando se necesita que el usuario introduzca alguna información.

***8.16 Los controles***

Realmente son objetos que disponen de sus propias propiedades y métodos, y cuya utilidad es la de facilitarnos el desarrollo de nuestras aplicaciones. Bueno, este intento de definición puede haber quedado bien, pero para que lo tengáis más claro, tener en cuenta que en cualquier aplicación con la que trabajamos estamos rodeados de controles. Quien no ha visto en multitud de programas los botones ACEPTAR y CANCELAR, un cuadro para introducir texto, una lista con datos, etc. Pues todos ellos son controles y no tendremos que preocuparnos por crearlos para nuestras aplicaciones, sino que ya vienen con el paquete de VB, lo único que tendremos que hacer es modificar sus propiedades: tamaño, color, etc. para incorporarlos en nuestras aplicaciones y asociarles el código necesario para que se comporten como esperamos al ejecutar la aplicación.

Antes de empezar a conocer los controles básicos veamos cuáles son sus características generales:

1. **Propiedades**: Todos los controles disponen de una serie de propiedades las cuales podemos cambiar al incluirlos en nuestras aplicaciones. Ejemplos de propiedades son el color, el tipo de letra, el nombre, el texto, etc.
2. **Métodos:** Son procedimientos asociados a los controles, es decir, rutinas ya establecidas que podemos invocar desde nuestras aplicaciones para que se realice alguna operación sobre el control. Por ejemplo, el control ListView (la lista de archivos que aparece en el explorador de Windows) dispone del método order que te ordena los datos aparecidos en la lista.
3. **Eventos:** Son acciones que pueden ser motivadas por el propio usuario o por mismo sistema operativo. Ejemplos pueden ser el movimiento del ratón o hacer click sobre su botón. En Visual Basic digamos que se utiliza la programación orientada a eventos, lo cual es una de las diferencias más importantes respecto a la programación lineal de MS DOS. No necesitamos detectar cuando se ha producido un evento determinado, Windows lo detecta automáticamente. Los eventos ya están definidos, son bastantes y cada control cuenta con los suyos propios, aunque son muy parecidos. Lo único que tendremos que hacer es asociar el código necesario al evento que necesitemos tratar.

Para mostrar la ventana donde aparecen los controles que Visual Basic carga por defecto nada más arrancar la aplicación tendremos que marcar en **View** del menú principal (versión inglesa) y activar la opción **Toolbox.** Obtendremos una ventana como esta en la pantalla:

Realmente existen muchos más controles, aunque estos son los más utilizados y por eso aparecen por defecto. Para tener acceso a los demás controles tanto de Visual Basic como los controles que incorporan otras aplicaciones marcaremos en **Tools/Custom Controls** del menú principal.

Moviendo el ratón por encima de cualquier control aparecerá una pista indicándonos el control de que se trata.

Para que esta ventana aparezca siempre en primer plano, aunque no sea la ventana activa marcaremos con el botón derecho del ratón en cualquier lugar de la ventana y activaremos la opción **Always On Top** del menú contextual.

## ***8.17 Controles básicos***

**TextBox**

Mediante este control podremos realizar tanto la entrada como la salida de datos en nuestras aplicaciones.

No hace falta que indiquemos las coordenadas de la situación del formulario en pantalla, simplemente tendremos que marcar sobre el control de la caja de herramientas y dibujarlo con el tamaño que queramos en nuestro formulario.

## **PROPIEDADES**

Las propiedades de las que dispone el control son las siguientes:(para obtener el cuadro de propiedades, seleccionar el control y pulsar **F4** o pulsar con el botón derecho para obtener el menú contextual y marcar **Propierties**)

**Text:** Aquí indicamos el texto que aparecerá en el control. Podemos asignarle cualquier texto en tiempo de diseño o ejecución. También podemos tomar el texto que haya introducido el usuario para tratarlo durante la ejecución.

**Name:** Esta propiedad la tienen todos los controles, el nombre que viene por defecto en este caso Text1 y es el nombre con el que se conocerá el control cuando lo utilicemos en el código. En un mismo formulario no puede haber 2 controles con el mismo nombre. Conviene poner un nombre que represente la función que tiene el control en la aplicación para que el código quede más claro. Ejemplo, si en el textbox vamos a introducir la dirección de una persona podemos asignarle a esta propiedad el valor Dirección.

**MultiLine:** Permite que introduzcamos varias líneas de texto en el control en lugar de sólo una.

**Alignment:** Alineación que tendrá el texto dentro del control: izquierda, centro o derecha. Para que funcione la propiedad MultiLine debe estar con el valor true.

**Locked:** Si esta con valor true bloquea el control, es decir, el usuario no puede introducir ni modificar el texto que contenga. Nos puede servir para utilizar el control como salida de datos sin que el usuario pueda modificarlos por error.

Otras propiedades que son comunes a la mayoría de los controles:

**Backcolor:** color de fondo.

**Forecolor:** color de letra.

**Font:** tipo y tamaño de letra.

## **METODOS**

Recordemos que por métodos se entienten los procedimientos o funciones asociados a un control, los cuales nos permiten realizar ciertas operaciones útiles sobre dicho control: Ej. ordenar sus elementos, buscar un dato, etc.

Pues bien, los controles básicos que vamos a ver en este capítulo únicamente contienen métodos avanzados que no vamos a analizar por ahora, ya que son métodos que no se suelen utilizar. Más adelante cuando veamos otros tipos de controles estudiaremos cuales son los métodos que nos podrán servir. Si alguien está interesado en conocer todas las características de los controles puede hacerlo mirando en la ayuda que proporciona VB, haciendo clic sobre cualquier control de la caja de herramientas y pulsando a continuación F1 obtendrá ayuda referente a ese control donde aparecerán todas sus propiedades, métodos y eventos.

## **EVENTOS**

Los eventos son acciones que se pueden realizar en cualquier control: click, doble click, movimiento del ratón. A estos eventos se les puede asociar código para que se ejecute al producir el evento.

**MouseMove**: al mover el ratón por encima del control.

**Mousedown:** al pulsar cualquier botón del ratón

**Change:** al cambiar el contenido del control

**Click:** al hacer click con el botón izquierdo del ratón sobre el control

**Doubleclick:** al hacer doble click con él con el botón izquierdo del ratón sobre el control

**Getfocus:** este evento se activa cuando el control recibe el enfoque, es decir, cuando se activa el control en tiempo de ejecución para introducir datos en él o realizar alguna operación.

**Lostfocus:** Es el contrario del anterior evento, se activa cuando el control pierde el enfoque, es decir, se pasa a otro control para seguir introduciendo datos.

## **EJEMPLO**

Vamos a probar el uso del control TextBox mediante un pequeño ejemplo en el que, teniendo un único control de este tipo en un formulario, lo programaremos de forma que al pasar el ratón sobre el control (evento **mousemove**) aparecerá en el formulario el texto que contenga.

Observamos que al situar el control en el formulario aparece por defecto el texto **Text1**. Para que no aparezca ese texto al ejecutar la aplicación, debemos cambiar la propiedad **Text** pulsando **F4** y colocar el texto que queramos o no colocar nada.

Lo que queremos hacer es que cada vez que movamos el ratón por el control aparezca su contenido en el formulario. Entonces lo que habrá que hacer abrir la ventana de código, seleccionando el control y pulsando **F7,** o con el botón derecho del ratón y la opción **View code** del menú contextual. Este proceso nos llevará al cuadro de la imagen siguiente.

Lo que tendremos que hacer es seleccionar el evento que necesitemos de la sección **Proc**, en nuestro caso mousemove y a continuación teclear el código correspondiente: La instrucción print visualiza un texto en el formulario y si le ponemos **text1.text** le decimos que nos muestre la propiedad **Text** del control **Text1** que ese será el nombre que tendrá el control por defecto si no lo hemos cambiado en la propiedad name.

Al ejecutar esta pequeña aplicación pulsando **F5** observaremos como aparece en el formulario lo que hayamos tecleado en el control cada vez que movemos el ratón sobre el Textbox.

Podemos modificar el programa para que responda a cualquier otro evento sin más que seleccionarlo en la sección **Proc** e introduciendo el código que sea necesario.

**Label**

Este control es también uno de los más utilizados, aunque su utilidad queda restringida a la visualización de datos en el mismo, no permitiendo la introducción de datos por parte del usuario.

La forma de utilizarlo es similar a la del control anterior, dibujar el control en el formulario con el tamaño que queramos y asignarle un texto en tiempo de diseño o de ejecución esta vez sin utilizar la propiedad text puesto que no la incorpora, sino utilizando la propiedad caption.

Este control sirve para mostrar mensajes en nuestro formulario que orienten al usuario sobre la utilidad de los demás controles que tengamos en la aplicación o para indicarnos acciones que podemos realizar. En el ejemplo anterior donde aparecía un textbox en el formulario, hubiera quedado mejor con un mensaje aclaratorio contenido en un control label:

###### **PROPIEDADES**

**Caption:** Es el texto que contendrá el control.

**Alignment:** Alineación del texto contenido en el control, no necesita que esté activada ninguna otra propiedad.

**BorderStyle**: Si queremos que aparezca un borde alrededor del control activaremos esta propiedad.

Para este control no se suelen utilizar los eventos ya que su contenido suele cambiar poco a lo largo de la ejecución de la aplicación. De todas formas, los eventos son casi los mismos del control textbox excepto que no dispone de los eventos **GetFocus** y **LostFocus** ya que a este control no se le puede dar el enfoque.

En la parte final de este capítulo veremos un ejemplo donde se muestra el funcionamiento de todos los controles que vamos a ir viendo. Por ahora a ver si conseguís que ahora el mensaje no aparezca en el formulario sino en un segundo label situado en el formulario, dejando un control label que muestre el mensaje aclaratorio que hemos visto antes.

**CommandButton**

Este control es el típico botón que aparece en todas las aplicaciones y que al hacer click sobre él nos permite realizar alguna operación concreta, normalmente Aceptar o Cancelar. Aunque según el código que le asociemos podremos realizar las operaciones que queramos.

En el ejemplo anterior podemos añadir un control de este tipo para salir de la aplicación sin tener pulsar sobre la equis de la esquina superior derecha.

Pero sólo con introducir un control de este tipo con el texto salir que se introduce a través de la propiedad caption no basta. Habrá que asociarle un código que nos permita salir de la aplicación en el evento adecuado. Y el evento por excelencia de este control es click. Así pues, accederemos al código del control y la sentencia nos permitirá salir de la aplicación es End, simplemente tecleamos esa palabra en el evento click y comprobar que realmente finalizaremos nuestra aplicación al pulsar sobre dicho botón.

## **PROPIEDADES**

**Caption:** Aqui pondremos el letrero que queremos que aparezca en el botón: aceptar, cancelar, salir, etc.

**Enabled:** Esta es una nueva propiedad, cuando su valor es true el botón funciona normalmente, cuando su valor es false el botón se encuentra desactivado, no responde a los eventos producidos sobre él y el texto aparece en un gris claro advirtiéndonos de su estado. Podemos utilizar esta propiedad para activar o desactivar un boton dependiendo del estado de otros controles. Por ejemplo, en un botón Aceptar, no activarlo hasta que se haya introducido una cantidad en un control textbox, ya que ese botón nos calculará el IVA de la cantidad.

## **EVENTOS**

**Click:** Es el evento típico de este control y el que más se utiliza.

**MouseMove**: Como sabemos detecta el movimiento del ratón sobre el control. Puede servir para que aparezca un mensaje en un control Label que nos aporte información sobre la utilidad del control ampliando el texto que hayamos colocado como caption del commandbutton.

**OptionButton**

Este control nos permite elegir una opción entre varias de las que se nos plantean. Cada opción será un control optionbutton diferente.

Facilita la introducción de datos por parte del usuario:

De todas las opciones que se nos ofrece, en este caso los 4 colores, sólo podremos activar una. Si activamos cualquier otra opción, se desactivará automáticamente la última que teníamos activada.

El marco que está alrededor de los 4 controles optionbutton se trata del control **Frame** , es opcional, aunque es conveniente colocarlo siempre que hagamos uso de las opciones. No sólo por motivos de presentación sino porque de esta manera podremos establecer grupos de controles optionbutton independientes en los que en cada grupo sólo pueda haber una opción activada a la vez. También, al mover el marco se moverán los controles incluidos en él facilitándonos las modificaciones.

Para que los controles Optionbutton queden englobados dentro de un control Frame, primero tendremos que colocar el control Frame en el formulario con el tamaño adecuado y despues ir colocando los controles Optionbutton dentro del Frame.

Del control Frame la única propiedad que nos interesará es caption, que es el texto que aparecerá en el encabezado, en el ejemplo anterior: colores.

## **PROPIEDADES DE OPTIONBUTTON**

**Caption:** El texto que aparecerá al lado del control: Rojo, verde, etc.

**Value:** Es el valor que tendrá el control: True si se encuentra activado y False si no lo está. Para comprobar que opción ha activado el usuario comprobaremos el estado de esta propiedad.

**Alignment:** Alineación del texto respecto al control: Left Justify: el control aparece a la izquierda del texto. Es el ejemplo anterior. Right Justify: el control aparece a la derecha del texto.

Los eventos del control son los mismos que en anteriores controles, aunque no se suele asociar código a los eventos de este tipo de controles, sino únicamente conocer el valor que tienen: true o false.

##### **APLICACIÓN DE EJEMPLO**

Para practicar con los controles que hemos visto vamos a realizar una pequeña aplicación que consistirá en realizar con 2 números que introduzcamos, una operación que seleccionemos y mostrar el resultado.

El formulario donde estarán todos los controles es el siguiente:

La propiedad **Caption** de cada uno de los controles es la que se muestra en el formulario.

He modificado la propiedad **Name** de cada control para que al utilizarlos desde el codigo sepamos cual es el control con el que trabajamos:

1. Los controles TextBox tienen los nombres: **Num1**, **Num2** y **Resul.**
2. Los controles Optionbutton tienen cada uno de ellos el mismo nombre que su caption.
3. Los controles CommandButton tienen los nombres: **Calcular**, **Limpiar** y **Salir.**
4. A los controles Label y al Frame no have falta cambiarles el nombre.

Lo que habrá que hacer ahora es asociar codigo a cada uno de los botones que es de donde se van a realizar las operaciones:

1. Para el botón **Calcular** que es el que nos mostrará el resultado según la operación seleccionada, he utilizado la instrucción If Then Else que vimos en el capítulo anterior:
2. El botón **Limpiar Datos** nos va a servir para borrar de una forma rápida los datos introducidos por el usuario y el resultado preparando los controles para introducir nuevos datos. El código que tendremos que introducir es muy simple.
3. El botón **Salir** únicamente contendrá la sentencia End.

**CONCLUSIONES PERSONALES**

La programación orientada a objetos permite la optimización del código generado gracias a que, mediante técnicas de herencia, atributos estáticos entre otros permiten, que el código sea genérico de manera que sea reutilizable.

Mediante las técnicas aprendida en el presente curso podemos establecer una solución primitiva de un problema real, tan solo con relacionarlo con objetos lógicos que serán usados para el desarrollo del software.

Podemos dar a conocer de una forma sencilla los mecanismos que se usan en este nivel de programación, a personas que deseen una explicación rápida y sencilla de lo que es la programación orientada a objetos.

Tenemos los conocimientos necesarios como para enfrentar un problema real y desarrollo en otro lenguaje de programación, pues concebimos la idea de que el lenguaje C es la base de la programación.

Al trabajar con la programación orientada a objetos sea esta desarrollada en otras plataformas de programación o en lenguaje C, sabemos las formas de lograr un mejor rendimiento del equipo a controlar y aplicar soluciones sencillas, de manera que sea fácilmente digeribles para el usuario y/o destinatario del trabajo final.

**BIBLIOGRÁFIA Y EGFRAFÍA**

* JEAN Paul Tremblay, B. Bunt Richard; "Introducción a la ciencia de las computadoras (enfoque algoritmico)" Mc Graw Hill
* JOYANES Aguilar Luis; "Metodología de la programación" Mc Graw Hill
* JOYANES Aguilar Luis; "Problemas de metodología de la programación" Mc Graw Hill
* CORREA Uribe Guillermo; "Desarrollo de algoritmos y sus aplicaciones en Basic, Pascal y C (3ª. Edición)" Mc Graw Hill
* Levine Guillermo; "Introducción a la computación y a la programación estructurada" Mc Graw Hill
* JOYANES Aguilar Luis; "Fundamentos de programación, algoritmos y estructura de datos"
* Mc Graw Hill
* JOYANES Aguilar Luis, Luis Rodríguez Baena y Matilde Fernández Azuela; "Fundamentos de programación, libro de problemas" Mc graw Hill
* Bores Rosario, Rosales Roman; "Computación. Metodología, lógica computacional y programación" Mc Graw Hill
* LOZANO Letvin; "Diagramación y programación estructurada y libre" Mc Graw Hill
* LOPEZ Roman Leobardo; "Programación estructurada (enfoque algorítmico)" Computec
* Ver B4X Basic4Android & [Basic4IPhone](http://b4x.vb-mundo.com/)
* [«What's new in VB6?»](http://www.insteptech.com/techLibrary/vbClassic/vb6_news.htm). www.insteptech.com.
* Al Tenhundfeld. [«Visual Basic 6.0 to be Supported on Windows 7»](http://www.infoq.com/news/2009/03/vb6_on_win7). The Register.
* [«Support Statement for Visual Basic 6.0 on Windows Vista, Windows Server 2008 and windows 7»](http://msdn.microsoft.com/en-us/vbrun/ms788708.aspx). Msdn.microsoft.com.
* [«Product Family Life Cycle Guidelines for Visual Basic 6.0»](http://msdn2.microsoft.com/en-us/vbrun/ms788707.aspx). Msdn2.microsoft.com.
* [«Petition to Microsoft»](http://web.archive.org/web/http:/www.classicvb.org/petition/). Classicvb.org. Archivado desde [el original](http://www.classicvb.org/petition/) el 29 de noviembre de 2015.
* Andrew Orlowski and Carey Bishop. [«MS Anti-Spyware built on MS Abandonware»](http://www.theregister.co.uk/2005/07/22/microsoft_spyware_vb6/). The Register.
* <http://www.microsoft.com/downloads/details.aspx?FamilyId=D7E31492-2595-49E6-8C02-1426FEC693AC&displaylang=en>
* [«VBOpenGL type library»](http://web.archive.org/web/http:/nio.astronomy.cz/vb/opengl.html). Archivado desde [el original](http://nio.astronomy.cz/vb/opengl.html) el 29 de noviembre de 2015.
* Alex Homer, Dave Sussman, Rob Howard, Brian Francis, Karli Watson, Richard Anderson (2004). [Professional ASP.NET 1.1](http://books.google.com/books?id=AsqK_AtFf1sC). Wiley. [ISBN](https://es.wikipedia.org/wiki/ISBN) [0764558900](https://es.wikipedia.org/wiki/Especial:FuentesDeLibros/0764558900).
* Marc D'Aoust. [«Avoid Writing Tedious, Boring Code»](http://msdn.microsoft.com/en-us/library/aa227881(VS.60).aspx). [Microsoft](https://es.wikipedia.org/wiki/Microsoft).
* Andrew Troelsen (2008). [Pro VB 2008 and the .NET 3.5 Platform: The expert's voice in .NET](http://books.google.com.au/books?id=y0uURUiVhgwC). Apress. p. 5. [ISBN](https://es.wikipedia.org/wiki/ISBN) [1590598229](https://es.wikipedia.org/wiki/Especial:FuentesDeLibros/1590598229).
* [DLL Hell](https://es.wikipedia.org/wiki/DLL_Hell), DB, Libro Programando en Visual Basic, 2002.
* [http://web.archive.org/web/http://www.realsoftware.com/realbasic/?lang=es](http://web.archive.org/web/http:/www.realsoftware.com/realbasic/?lang=es)