



COLEGIO DE BACHILLERES

SECRETARÍA ACADÉMICA

**COORDINACIÓN DE ADMINISTRACIÓN
ESCOLAR Y DEL SISTEMA ABIERTO**

COMPENDIO FASCICULAR

**LÓGICA COMPUTACIONAL
Y
PROGRAMACIÓN**

FASCÍCULO 1. LÓGICA COMPUTACIONAL

FASCÍCULO 2. DIAGRAMAS DE FLUJO

FASCÍCULO 3. SEUDOCÓDIGOS

FASCÍCULO 4. PROGRAMACIÓN PASCAL



DIRECTORIO

Roberto Castañón Romo
Director General

Luis Miguel Samperio Sánchez
Secretario Académico

Héctor Robledo Galván
**Coordinador de Administración
Escolar y del Sistema Abierto**

Derechos reservados conforme a la Ley
© 2004, COLEGIO DE BACHILLERES
Prolongación Rancho Vista Hermosa núm. 105
Col. Ex Hacienda Coapa
Delegación Coyoacán, CP 04920, México, D.F.

ISBN 970 632-240-X

Impreso en México
Printed in México

Primera edición: 2004

Í N D I C E

INTRODUCCIÓN	5
PROPÓSITO	7
FASCÍCULO 1. LÓGICA COMPUTACIONAL	11
1.1 ALGORITMO	12
1.2 ETAPAS DE UN ALGORITMO	13
1.3 TÉCNICAS DE SOLUCIÓN DE PROBLEMAS	16
1.3.1 TÉCNICA LINEAL	16
1.3.2 TÉCNICA ESTRUCTURADA	16
1.3.3 TÉCNICA ORIENTADA A OBJETOS	19
ACTIVIDAD DE REGULACIÓN	20
FASCÍCULO 2. DIAGRAMAS DE FLUJO	23
2.1 ELEMENTOS BÁSICOS	25
2.2 SIMBOLOGÍA	26

2.3 REGLAS DE CONSTRUCCIÓN	29
2.4 ESTRUCTURA DE DATOS	30
2.5 ESTRUCTURAS ALGORÍTMICAS O DE CONTROL	34
ACTIVIDAD DE REGULACIÓN	43
FASCÍCULO 3. SEUDOCÓDIGOS	47
3.1 DEFINICIÓN	48
3.2 TRANSFORMACIÓN DE ESTRUCTURAS ALGORÍTMICAS A SEUDOCÓDIGOS EN CASTELLANO	49
3.3 APLICACIÓN DE SEUDOCÓDIGOS	52
ACTIVIDAD DE REGULACIÓN	54
FASCÍCULO 4. PROGRAMACIÓN PASCAL	57
4.1 CONCEPTOS BÁSICOS DE LOS LENGUAJES	58
4.2 ELEMENTOS PARA PROGRAMAR EN TURBO PASCAL	61
4.3 AMBIENTE PROPIO DE OPERACIÓN EN TURBO PASCAL	71
4.4 ESTRUCTURA DE UN PROGRAMA EN TURBO PASCAL	73
ACTIVIDAD DE REGULACIÓN	75
ACTIVIDADES DE CONSOLIDACIÓN	76
AUTOEVALUACIÓN	85
GLOSARIO	91
BIBLIOGRAFÍA CONSULTADA	92

INTRODUCCIÓN

La informática se ha convertido en un instrumento estratégico para mejorar la calidad de los productos generados, permitiendo una mejor productividad, eficiencia y competitividad.

De esta forma la capacitación en informática tiene incorporada la asignatura de lógica computacional y programación, el cual es una asignatura clave para entender la forma de plantear propuestas de soluciones a problemas elementales de forma lógica y sistemática.

Esta asignatura se compone de cuatro fascículos; en la cual el fascículo 1 lógica computacional, nos permitirá adquirir las bases de la lógica mediante la generación de algoritmos y la aplicación de las técnicas para la resolución de problemas.

El fascículo 2 diagramas de flujo aprenderás a, elaborar diagramas de flujo mediante la utilización de su simbología, ya que te permitirá representar gráficamente la solución de un problema determinado.

El fascículo 3 pseudocódigos nos permitirá establecer las bases para poder programar en cualquier lenguaje.

Y por ultimo el fascículo 4 programación pascal, te permitirá elaborar programas en el lenguaje pascal los cuales te ayudaran a utilizar las técnicas de programación estructurada.

PROPÓSITO

En este fascículo:

¿Qué aprenderás?

La lógica y las bases para poder programar en cualquier lenguaje de programación.

¿Cómo lo Aprenderás?

Utilizando las bases de la lógica y abordando las técnicas de resolución de problemas, así como las bases para programar en el lenguaje Pascal. Obteniendo programas que den solución a problemas planteados.

¿Para qué lo Aprenderás?

Para entender la forma de plantear propuestas de soluciones a problemas elementales, de forma lógica y sistemática.



COLEGIO DE BACHILLERES

**LÓGICA COMPUTACIONAL
Y
PROGRAMACIÓN**

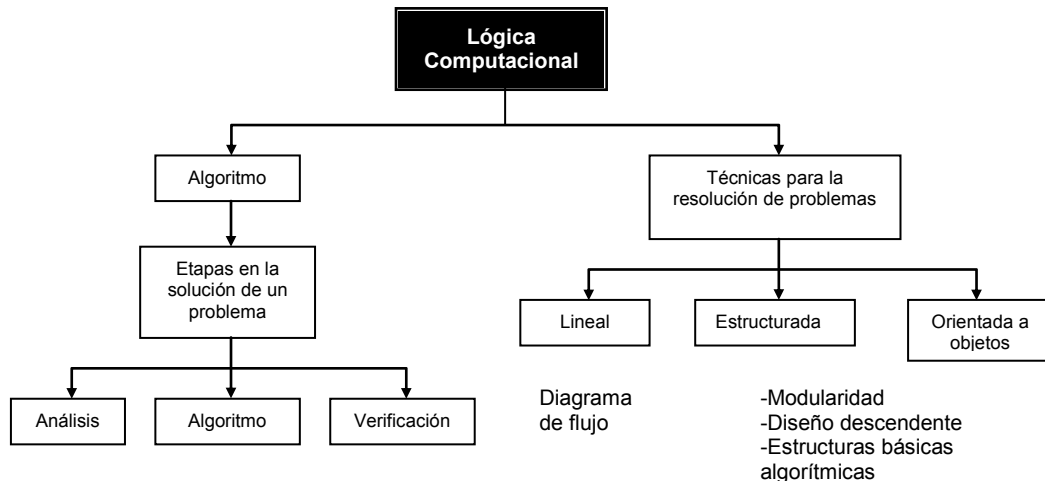
FASCÍCULO 1. LÓGICA COMPUTACIONAL

LÓGICA COMPUTACIONAL

OBJETIVO: *Adquirirás las bases de la lógica computacional*, mediante la generación de algoritmos y la aplicación de las técnicas para la resolución de problemas; para que obtengas un razonamiento sistemático y lógico que te permita seleccionar la mejor técnica para aplicarla en la solución de los problemas planteados.

1.1 ALGORITMO

Este fascículo se encuentra organizado de la siguiente forma:



La lógica computacional está relacionada íntimamente con la manera en que utilizas tu pensamiento lógico, esto es, con la forma en que resuelves un problema.

Cuando se plantea un problema, para su resolución se aplican diferentes técnicas, como lo son la lineal, las estructurada y la orientada a objetos.

Veamos lo siguiente:

1.2 ETAPAS DE UN ALGORITMO

Básicamente, existen tres etapas que son:

- 1) **Análisis profundo del problema**, en el cual se comprende con claridad, cuál es el problema, que debes lograr y perfilar una posible solución.
- 2) **Construcción del algoritmo o diseño de la solución del problema**, en donde se realiza una secuencia ordenada de pasos lógicos que conducen a la solución de un problema.
- 3) **Verificación del algoritmo**, se ejecuta y valida la secuencia anterior, también es factible realizarse mediante un programa de computadora.

La **etapa de análisis** es la parte medular para resolver problemas, ya que a partir de ésta se diseña, construye y prueba un algoritmo que presenta la solución de un problema.

Se incluye un nuevo concepto que nos lleva a la pregunta ¿qué es un algoritmo?

No podrías contar cuántos algoritmos utilizas día a día, ya sea en tu casa el trabajo cuando realizar actividades recreativas, etc. Para poder comprenderlos, se te presenta un algoritmo sencillo:

Se quiere abrir una puerta, la persona se encuentra frente de ella, tiene la llave que abre

¿Qué pasos requieres hacer para abrirla?

- | | |
|---------|---------------------------------|
| Paso 1 | Coloca la llave en la cerradura |
| Paso 2. | Gira la llave a la derecha |
| Paso 3. | Da una vuelta |
| Paso 4 | El pasador cede |
| Paso 5. | La puerta se abre |

Un **algoritmo** es una serie de pasos, procedimientos o acciones que llevan una secuencia lógica y sistemática que permiten alcanzar un resultado o resolver un problema.

Estos pueden ser tan sencillos o tan complejos como se requieran, pero la finalidad es que cumpla con las siguientes **características**:

- a) **Preciso** en el problema que se plantea (indica el orden de realización en cada paso).
- b) **Determinístico**, dados un conjunto de datos de entrada, deberán arrojar los mismos resultados siempre (si se sigue dos veces, obtiene el mismo resultado cada vez).

- c) **Finito**, el algoritmo siempre debe de tener un fin de importar si es simple o complejo (tiene fin; un número determinado de pasos).

Un algoritmo debe producir un resultado en un tiempo finito. Los métodos que utilizan algoritmos se denominan métodos algorítmicos, en oposición a los métodos que implican algún juicio o interpretación que se denominan métodos heurísticos. Los métodos algorítmicos se pueden implementar en computadoras; sin embargo, los procesos heurísticos no han sido convertidos fácilmente en las computadoras. En los últimos años las técnicas de inteligencia artificial han hecho posible la implementación del proceso heurístico en computadoras.

Ejemplos de algoritmos son:

- * Instrucciones para montar en una bicicleta.
- * Hacer una receta de cocina.
- * Obtener el máximo común divisor, etc;

Los algoritmos se pueden expresar por formulas, diagramas de flujo o en N - S y SEUDO códigos. Esta última representación es la más utilizada en lenguajes estructurados como Turbo Pascal.



Para la **construcción** de un algoritmo se realizan **tres módulos**:

Entrada de Datos	Procesamiento de Datos	Salida de Resultados
Acción u operación que permite el ingreso de datos del problema	Operación u operaciones secuenciales, lógicas y organizadas, cuyo objetivo es obtener al procesar los datos de entrada.	Operación ó conjunto de operaciones que permiten al exterior los resultados alcanzados.

Si se aplican estos módulos al ejemplo del algoritmo de la cerradura tendrías:

• Entrada de Datos	📖 Estado de Puerta = CERRADO
• Procesamiento de Datos	<ul style="list-style-type: none"> ✓ Introducir la llave ✓ Giro de la llave a la derecha ✓ Le damos la vuelta ✓ Se abre la puerta
• Salida de Datos	📖 Estado de Puerta = ABIERTO

El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos profundos de lo que se requiere resolver, si consideras este algoritmo de una forma más compleja la construcción de estos módulos sería:

<ul style="list-style-type: none"> • Entrada de Datos 	 Estado de puerta = CERRADO
<ul style="list-style-type: none"> • Procesamiento de Datos 	<ul style="list-style-type: none"> ✓ Introducir la llave correcta ✓ Giro de la llave a la derecha una vez ✓ Otro giro a la derecha <p>En caso contrario</p> <ul style="list-style-type: none"> ✓ Giro a la izquierda ✓ Otro giro a la izquierda ✓ Se abre la puerta si el pasador cede <p>En caso contrario</p> <ul style="list-style-type: none"> ✓ Empujar para subir
<ul style="list-style-type: none"> • Salida de Datos 	 Estado de Puerta = ABIERTO

Como puedes observar, un algoritmo te permite describir la solución de un problema por medio de pasos lógicos y sistemáticos, precisos, determinísticos y finitos, se apoya para su construcción en tres módulos que son entrada de datos, procesamiento y salida de resultados.

1.3 TÉCNICAS DE SOLUCIÓN DE PROBLEMAS

Para lograr una solución satisfactoria, existen tres técnicas que nos permiten resolver un problema planteando: lineal, estructurada, y orientada a objetos.

1.3.1 Técnica Lineal

Consiste en el desarrollo de instrucciones que se ejecutan secuencialmente, una tras otra, actualmente ha perdido terreno, debido a que al programar es muy difícil adecuar un programa únicamente con instrucciones secuenciales, ya que ocasionaba limitantes en el desarrollo de los programas.

¿Cómo quedaría con la técnica lineal, el ejemplo de la cerradura?

1. Colocamos la llave de la cerradura
2. Giramos la llave a la derecha
3. Le damos una vuelta
4. El pasador cede
5. La puerta se abre

Como observas, se va ejecutando instrucción secuencialmente, es decir, de la uno a la cinco.

Cabe preguntarnos, ¿Qué tan funcional es? Comenta la respuesta con tu profesor y compañeros.

1.3.2 Técnica Estructurada

Se crea por la necesidad de solucionar los problemas de la técnica lineal y replantearse la manera de elaboración de programas, aumentando considerablemente la calidad del programa, a Edsger W. Dijkstra se le considera como el padre de la programación estructurada, también llamada sistemática.

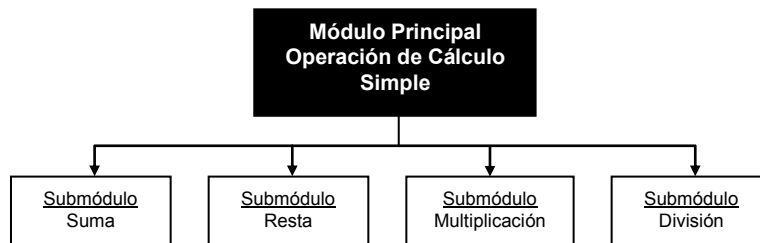
Esta técnica se basa en la creación de programas auxiliándose en la creación de submódulos, en el que cada submódulo se divide en partes independientes, cada una de las cuales ejecuta una única actividad o tarea y se describe o codifica independientemente de otros submódulos.

Ejemplo:

Se requiere elaborar un programa que realice las cuatro **operaciones de cálculo simple** como la **suma, resta, multiplicación**, división de dos números y obtener su resultado.

El programa debe contener un módulo llamado **módulo principal o programa principal** el cual tiene la función de llevar el control de lo que sucede, pasando el control a otros submódulo o subprogramas de modo que ejecuten sus funciones, devolviendo el control al programa principal cuando se termine la tarea o actividad.

ESQUEMA DE UN PROGRAMA

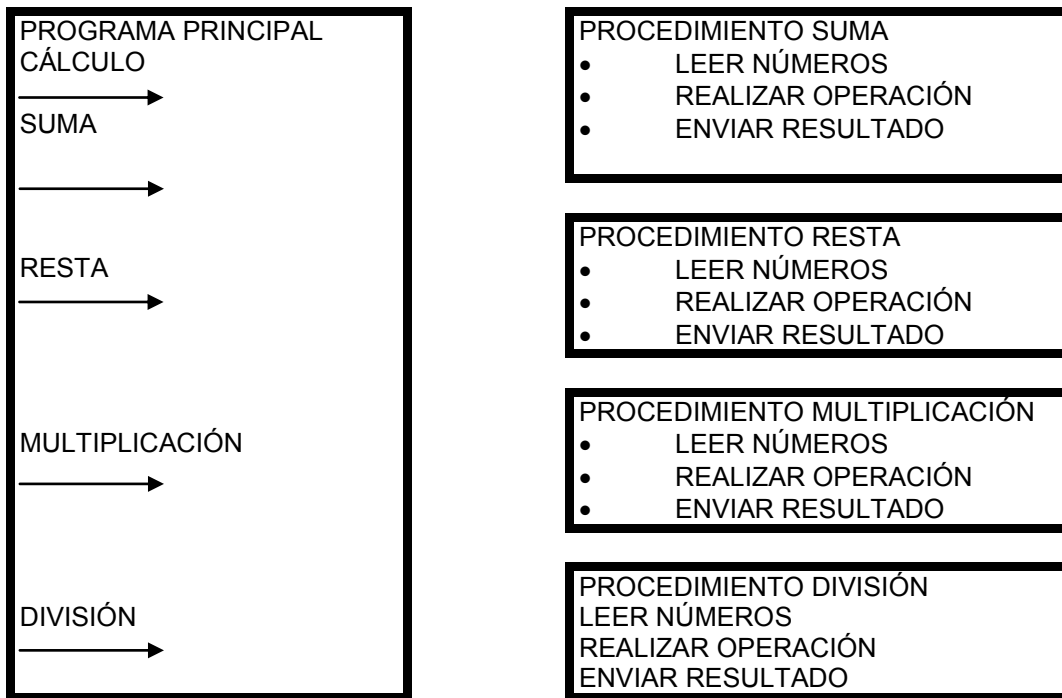


Si la tarea o actividad encargada al submódulo es muy compleja se recomienda crear otros submódulos dentro de ella, de manera que cada submódulo pueda ejecutar la tarea encomendada. El submódulo se crea para una tarea específica, regresando resultados.

Los submódulos toman diferentes nombres según el lenguaje de programación en el que se desarrolle el programa, en el caso de **Pascal** que es un **lenguaje de programación estructurado** con el que vamos a trabajar en esta asignatura, se les llama **procedimientos o funciones**.

En el lenguaje algorítmico, un submódulo se denomina **subalgoritmo**.

En el ejemplo del cálculo de operaciones, tenemos en el programa principal el llamado a cuatro submódulos o procedimientos: suma, resta, multiplicación y división, en donde las operaciones que realice dentro de cada una de ellas, son independientes de las otras, pero que en conjunto forman el programa principal.



Programa Principal y Procedimientos

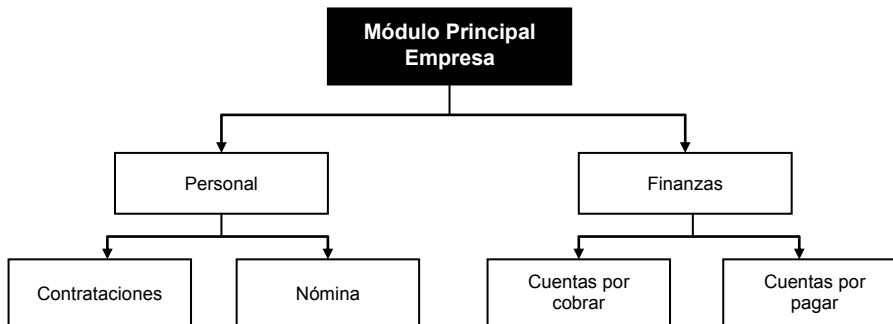
Al ser independientes los submódulos, es factible dividirse el trabajo, de manera que se avance en el desarrollo del programa y se reduzca el tiempo de proceso y elaboración.

La **técnica estructurada** esta compuesta de un conjunto de tres técnicas:

A) Diseño Descendente (Top-Down): Consiste en descomponer en niveles jerárquicos el problema.

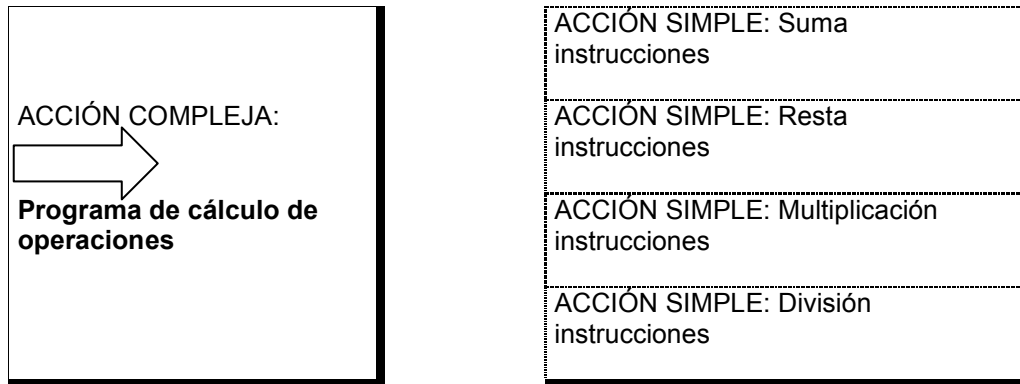
Ejemplo:

Se requiere elaborar un programa que automatice en una empresa el departamento de personal y de finanzas, representándolos en niveles jerárquicos quedaría así:



Se pueden descomponer en cuantos niveles se necesite. Esto permite que los programadores se puedan repartir los submódulos y unirlos ya que estén listos para formar el sistema.

B) Recursos Abstractos: Descompone una acción compleja en un número de acciones más simples que se ejecutan en la computadora y se convierten en instrucciones.



C) Estructuras básicas Algorítmicas: Compuestas de tres estructuras

1. **Secuencial:** Es aquella en la que una instrucción o acción sigue a otra es decir la salida de una es la entrada de la otra.
2. **Selectiva:** De un número determinado de alternativas o acciones se selecciona una o vanas condiciones lógicas que se apegan al problema.
3. **Repetitiva:** Repite una o varias acciones un número determinado de veces, llamado bucle y se llama interacción al hecho de repetir la ejecución de una secuencia de acciones.

Para el manejo de los programas en lenguaje Pascal la programación estructurada ha tenido éxito, y es en la actualidad como se trabaja este lenguaje, el inconveniente que existe es la dependencia entre los módulos, de tal forma que la revisión o mejoramiento de alguno de ellos requiere la modificación de otras partes del programa. Es por ello que surge una nueva técnica de resolución de problemas llamada:

1.3.3 Técnica Orientada a Objetos

Es una nueva técnica, la cual permite representar un avance mucho mayor hacia la modularización, utiliza estructuras llamadas objetos que unen procedimientos y funciones llamados encapsulados que forman la unidad.

Un programa orientado al objeto tiene un solo tipo de identidad para representar tanto la información como su manipulación. El objeto representa a ambos.

La programación orientada al objeto proporciona una metodología de programación soportada por los nuevos lenguajes (Visual Basic, Delphi, Power Builder) y herramientas para mejorar su productividad.

Con esta técnica se enseña a resolver problemas, usando como herramienta la computadora, por ello un programador de sistemas debe ser eficiente manejando problemas en forma rigurosa y sistemática. La noción central de esta metodología es el algoritmo.



ACTIVIDAD DE REGULACIÓN

Para que confirmes e integres los conocimientos que has alcanzado hasta este momento. Realiza la siguiente actividad.

1. Realiza un Algoritmo que nos represente el cambio de una llanta para un automóvil.



COLEGIO DE BACHILLERES

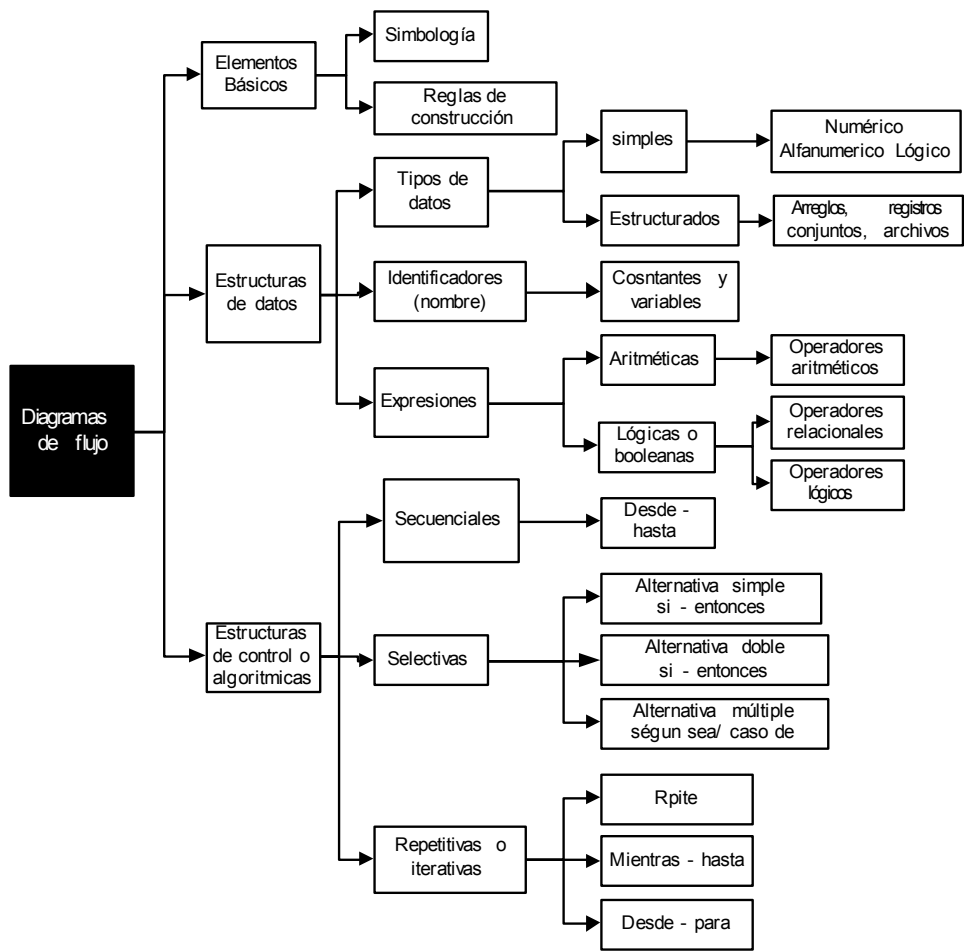
**LÓGICA COMPUTACIONAL
Y
PROGRAMACIÓN**

FASCÍCULO 2. DIAGRAMAS DE FLUJO

DIAGRAMAS DE FLUJO

OBJETIVO: Elaborarás diagramas de flujo, utilizando su simbología, las estructuras de datos y las estructuras algorítmicas; lo que te permitirá representar gráficamente la solución a un problema determinado.

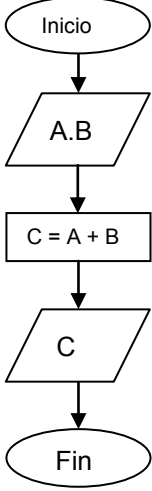
Este fascículo se encuentra organizado de la siguiente forma:



2.1 ELEMENTOS BÁSICOS



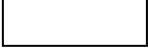

Un **diagrama de flujo** es una técnica que te permite la representación gráfica de un algoritmo, muestra los pasos o procesos para alcanzar la solución a un problema, utiliza símbolos estandarizados y normalizados por ANSI (American National Standards Institute), en donde cada símbolo representa el tipo de operación a ejecutar.

Elaboremos el algoritmo y el diagrama de flujo de la suma de dos números


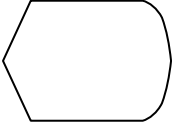
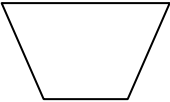
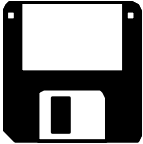
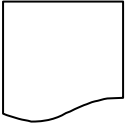

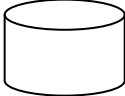

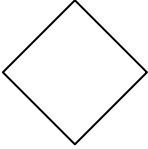

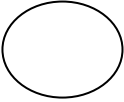
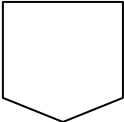

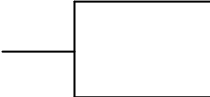
Algoritmo	Diagrama de Flujo
<ol style="list-style-type: none">1. Iniciar.2. Leer los números A y B.3. Realizar la suma de los números A+B el resultado asignarlo a C.4. Escribir el resultado "C".5. Salir.	 <pre>graph TD; Inicio([Inicio]) --> AB[/A.B/]; AB --> Suma[C = A + B]; Suma --> C[/C/]; C --> Fin([Fin]);</pre>

2.2 SIMBOLOGÍA

Con base en lo anterior se aprecian los principales símbolos:

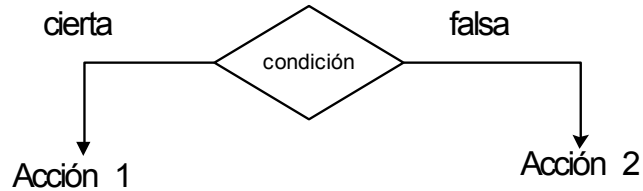
* El diagrama de flujo debe de tener un inicio y un fin , representado por:	
* El símbolo de lectura y escritura de datos o también llamado entrada/salida de datos:	
* El símbolo de proceso automático que nos permite hacer cálculos y asignar valores:	
* El símbolo de flujo de información , nos indica la dirección que sigue la información:	

Otros símbolos que se utilizan son:

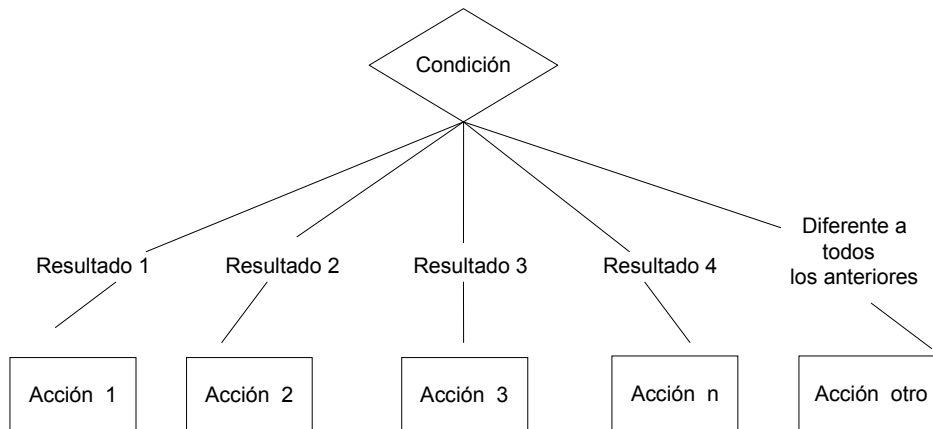
Teclado		Pantalla	
Proceso manual		Disco flexible	
Salida en papel		Unidad CD	
Disco magnético		Almacenamiento de acceso secuencial	
Decisión		Cassette	
Conector dentro de una misma página		Conector entre diferentes páginas	
Subrutina submódulo		Comentarios	

Cuando se utiliza el símbolo de decisión, puede ser simple o múltiple:

Decisión simple: Se evalúa una condición en su interior, y dependiendo del resultado, se selecciona uno de los caminos establecidos. Este símbolo también es conocido como **estructura algorítmica (o de control) selectiva sí entonces y sí entonces / sino**.



Decisión múltiple: En su interior se evalúa una condición, y a partir del valor del resultado, se selecciona una de las acciones planteadas, en caso no existir una opción equivalente al resultado, se podrá implementar una acción para una respuesta diferente a las esperadas. También se le llama estructura **algorítmica selectiva múltiple**.



2.3 REGLAS DE CONSTRUCCIÓN

Algunas reglas que debes de tener en cuenta **al construir los diagramas de flujo** son:

1. Debe ser de arriba hacia abajo (top - down) y de izquierda a derecha (left to right).
2. Debe tener un inicio y un fin.
3. Las líneas de flujo para indicar la dirección del flujo de información deben ser rectas, verticales y horizontales.
4. Todas las líneas de flujo que se utilizan para indicar la dirección deben estar conectadas a algún símbolo.
5. La notación utilizada debe ser independiente del lenguaje de programación.
6. No debe de llegar más de una línea a un símbolo.

2.4 ESTRUCTURA DE DATOS

Al elaborar un diagrama de flujo utilizamos su simbología y reglas de construcción, pero también existe una parte fundamental e indispensable que son los **tipos de datos** que maneja, estos pueden ser **simples o estructurados** y están agrupados así:

	Concepto	Ejemplo
SIMPLES	Hace referencia a un único valor, ocupa un espacio de memoria.	Numéricos, carácter, lógicos, enumerados y subrango.
ESTRUCTURADOS	Tiene varios componentes, los cuales pueden ser un dato simple ó estructurado, ocupa un grupo de espacios de memoria.	Arreglos, cadena de caracteres, registros, conjuntos y archivos.

Identificadores.

Las celdas de memoria (constantes o variables) tienen un nombre que nos permite identificarla. Un identificador es una secuencia de letras y dígitos, aunque el primer carácter debe ser una letra. La selección del identificador es una parte importante de una buena programación, ya que si el identificador es bien seleccionado hace que los programas sean más fáciles de leer y comprender.

Ejemplo; SUMA JUANA rosa datos_de_entrada

Constantes y Variables.

Para el manejo de datos en un programa es necesario establecer si los valores van a cambiar o no, con la intención de poder declararlos o definirlos de la forma más adecuada. Si el dato **no va a cambiar** durante el programa, entonces el valor podemos declararlo como **constante**. Si los datos **van a cambiar**, entonces es necesario definir una variable.

Expresiones y Operadores.

A la combinación de constantes, variables, símbolos de operación, paréntesis y nombre de funciones especiales se les llama **expresiones**. Cada expresión toma un valor, que se determina cuando los valores de las constantes y las variables implicadas, son evaluadas en la expresión.

Ejemplo:

8+9-6 c+r+(w-5)+t/6 h/a-q (a*b)+g

Una expresión consta de **operandos y operadores**. En el ejemplo anterior el valor de 8, 9 y 6 se le llama operandos, se le llama operadores aritméticos a los símbolos +, -, *, /, **, div (división entera) y mod (residuo).

Ejemplo: 21 div 8 el resultado toma el valor 2
 21 mod 8 el resultado arrojaría 5

Para realizar operaciones necesitamos de operadores aritméticos y lógicos:

- **Operaciones Aritméticas:** Nos permiten elaborar operaciones aritméticas entre operandos. El resultado de la expresión es de tipo numérico.

OPERACIONES ARITMÉTICAS			
Operador Aritmético	Operación	Ejemplo	Resultados Obtenidos
**	Exponenciación	2**3	8
*	Multiplicación	8.45*6	50.7
/	División	17/6	2.83
+	Suma	25.36+11.098	36.458
-	Resta	78.09-13.2	64.89
mod	Módulo (residuo)	13 mod 2	1
div	División entera	19 div 3	6

Para evaluar las expresiones que tienen operadores aritméticos, es indispensable que se lleven a cabo algunas jerarquías de orden de aplicación o reglas de prioridad, primero se ejecuta el operador de jerarquía mayor prioridad al de menor.

Jerarquía de los Operadores Aritméticos

Operación	Operador	Jerarquía
Exponenciación	**	del mayor
Multiplicación, división, módulo, división entera	*,/,mod,div	al
Suma, resta	+,-	menor

Para resolver las expresiones aritméticas, cuando tienen uno o más operandos se establecen algunas **reglas**, las cuales nos permitan determinar el orden de las operaciones:

- 1) Las operaciones que están encerradas entre paréntesis se elaboran primero. Si existieran otros paréntesis interiores (anidados: uno dentro de otro) las expresiones más internas se evalúan primero.
- 2) Los operadores aritméticos se aplican tomando en cuenta las jerarquías y de izquierda a derecha.
- 3) Si coincidieran varios operadores de igual prioridad en una expresión el orden de prioridad será de izquierda a derecha.

> **Expresiones Booleanas o lógicas:** Se forman combinando constantes lógicas variables lógicas y otras expresiones lógicas, utiliza los operadores lógicos (not and y or) y los operadores relacionales. El valor que pueden tomar estas expresiones es el de Verdadero o Falso, se ocupa en las estructuras algorítmica selectivas dependiendo del resultado de la evaluación se toma por un determinado camino.

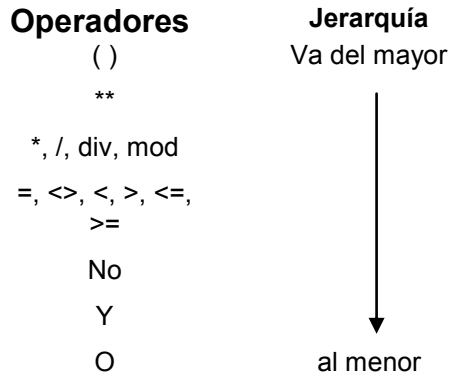
Operadores Relacionales: Son operadores que nos permiten comparar dos operandos, de tipo numérico o carácter. El resultado de una expresión con operadores relacionales es verdadero o falso.

Operador Relacional	Operación
=	Igual que
<>	Diferente a
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Operadores Lógicos: Nos permiten formular condiciones complejas a partir de condiciones simples. Los operadores son:

Operador Lógico	Operación
No	Negación
Y	Conjunción
O	Disyunción

La jerarquía de todos los operadores (aritméticos, relacionales y lógicos) es:



Asignación.

La asignación se utiliza para destinar valores o expresiones a una variable, si antes de la asignación la variable tenía un valor almacenado, éste se pierde y conserva el nuevo valor.

Para asignar valores a una variable se debe tomar en cuenta la siguiente sintaxis:

Variable = valor o expresión (nota: consideramos que la expresión puede ser aritmética o lógica, una variable o una constante)

Ejemplo:

1 = 0

ACUM = 1

CAR = a

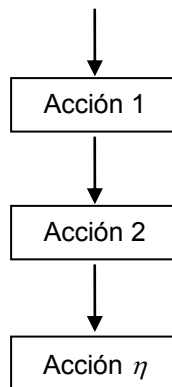
2.5 ESTRUCTURAS ALGORÍMICAS O DE CONTROL

Son otros símbolos que se utilizan en el diagrama de flujo, estas estructuras **algorítmicas** están conformadas en tres grupos:

- a) Secuenciales.
- b) Selección.
- c) Repetitivas o iterativas.

➤ **Estructura Algorítmica Secuencial.**

Se utiliza cuando queremos mostrar que de una acción le sigue otra y así sucesivamente, hasta el final del proceso. Para indicar el proceso utilizamos el rectángulo y en la parte interna la descripción de la acción.



Esta estructura tiene una entrada y una salida.

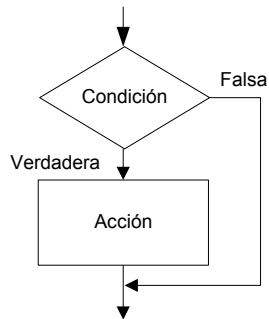
➤ **Estructura Algorítmica Selectiva.**

Se utiliza cuando se quiere tomar una decisión lógica, por ello también se le llama estructura de decisión o alternativa. En estas estructuras se evalúa una condición utilizando expresiones lógicas y dependiendo del resultado de ésta, se realiza una opción u otra. Su representación gráfica se expresa utilizando un rombo.

Las estructuras selectivas pueden ser:

a) *SI entonces o alternativa simple.*

Se ejecuta una acción cuando se cumple una condición. Esta estructura en inglés se conoce como (If then), su representación es:

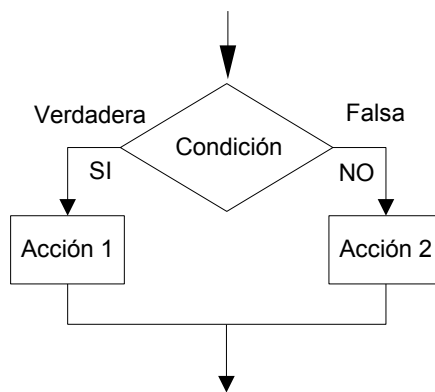


Si al evaluar la condición:

- * Es verdadera, **entonces** se ejecuta la acción o acciones.
- * Pero si la condición es falsa, entonces no se hace ninguna acción.

b) Si - entonces - sino o alternativa doble.

Se permite elegir entre dos opciones o alternativas posibles, de tal forma que si el resultado es verdadero se sigue por el camino correspondiente y se procede a ejecutar la(s) acción(es); pero si el resultado de la condición es falso entonces se sigue el otro camino y se procede a ejecutar la(s) otra(s) acción(s). Esta estructura en inglés se conoce como (If then else).



Independientemente de cuál camino se ejecute y qué condición siga, al término de las acciones, se continúa con la secuencia normal.

Se le llama **estructura algorítmica selectiva en cascada o anidada** cuando al tomar una decisión y seleccionar un camino, es necesario tomar otra decisión un número finito de veces.

c) Según_sea, caso de o alternativa múltiple.

Permite que el flujo del diagrama se divida por varios caminos en el punto de la toma de decisión, dependiendo el valor que tome el resultado es la acción que se ejecutará. Si el resultado toma el valor de 1, se ejecutará la acción 1, si el resultado toma el valor n, se ejecutara la acción n.

En el caso de que se tome un valor que no esté entre 1 y n, se continuará con el flujo normal pero se realizará la acción correspondiente al caso, donde la respuesta no se encuentre en los valores anteriores.

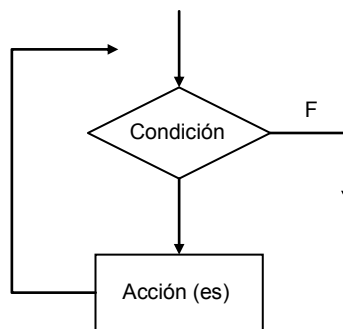
➤ **Estructura algorítmica Repetitiva o Iterativa.**

Llamada así porque permite repetir una o varias acciones un número determinado de veces. A las estructuras que repiten una secuencia de instrucciones, un número determinado de veces, se denominan **bucles**, y se le llama **iteración** al hecho de repetir la ejecución de una secuencia de acciones.

Este tipo de estructura esta conformada en tres partes:

a) Estructura mientras (en inglés While o dowhile).

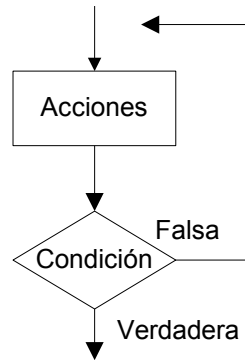
Es aquella que se ejecuta mientras la condición evaluada resulte **verdadera**, esta estructura es adecuada para utilizar un ciclo que se aplica un número determinado de veces, repitiendo una operación, acción, o tarea. Si la condición es falsa de entrada, no se realizará ninguna de las acciones asignadas



b) Estructura repetir (en inglés repeat).

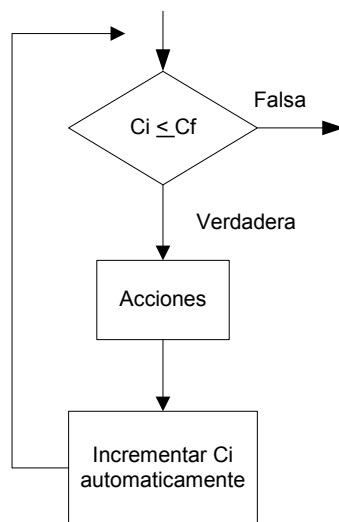
Se ejecuta un número definido de veces (se conoce cuántas veces tenemos que repetir una determinada acción o acciones) y se ejecuta hasta que se cumpla una condición determinada, que se comprueba al final del bucle. En esta estructura, el cuerpo del bucle se ejecuta al menos una vez.

Cuando la instrucción **repetir** se ejecuta, se hace la ejecución del bucle y se evalúa la expresión booleana resultante de la condición, **si es falsa** el cuerpo del bucle **se repite** y la expresión se vuelve a evaluar, después de cada iteración del cuerpo del bucle, se evalúa, si resulta **verdadera** el bucle termina y pasa a la siguiente instrucción.



c) Estructura desde / para (en inglés for).

Se utiliza cuando se conoce el número de veces que se desea ejecutar las acciones de un bucle, en el que el número de iteraciones es fijo. Comienza con un valor inicial índice (**contador inicial**) y las acciones especificadas se ejecutan si el valor inicial es menor que el valor final (**contador final**). La variable índice se **incrementa** en uno y si el valor no excede al final, se ejecuta de nuevo las acciones.

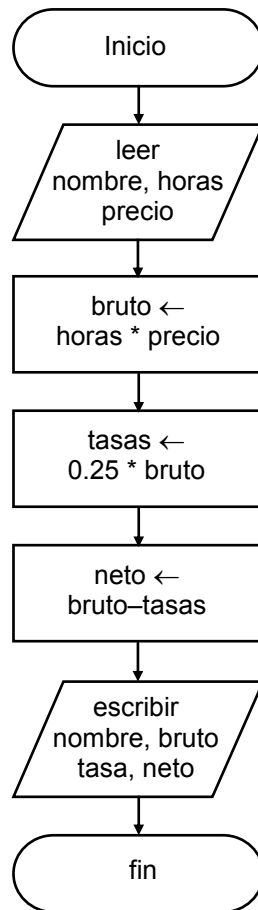


Hasta aquí hemos revisado los elementos básicos de un diagrama de flujo, su simbología y estructuras algorítmicas. Ahora retomaremos los elementos anteriores para construir algunos ejemplos.

Recuerda que cada Diagrama de Flujo debe ir precedido por su algoritmo para facilitarnos el trabajo.

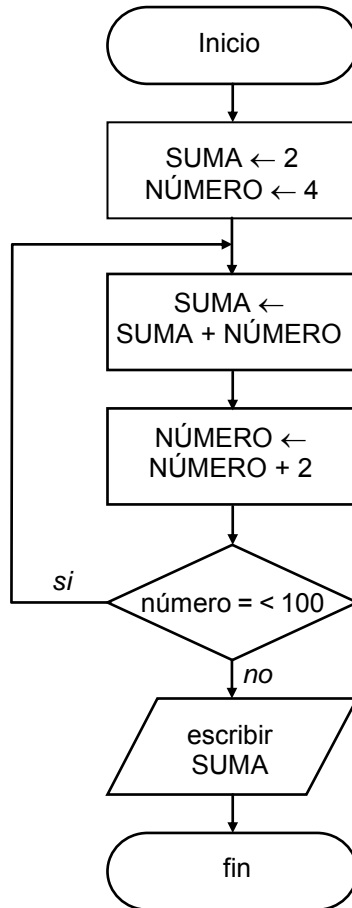
Ejemplo 1:

La siguiente figura es un diagrama de flujo básico, que representa la resolución de un programa que deduce el salario neto de un trabajador a partir de la lectura del nombre, horas trabajadas, precio de la hora, y sabiendo que los impuestos aplicados son el 25 por 100 sobre el salario bruto.



Ejemplo 2:

Analiza el siguiente diagrama de flujo y trata de determinar cuál es su función.



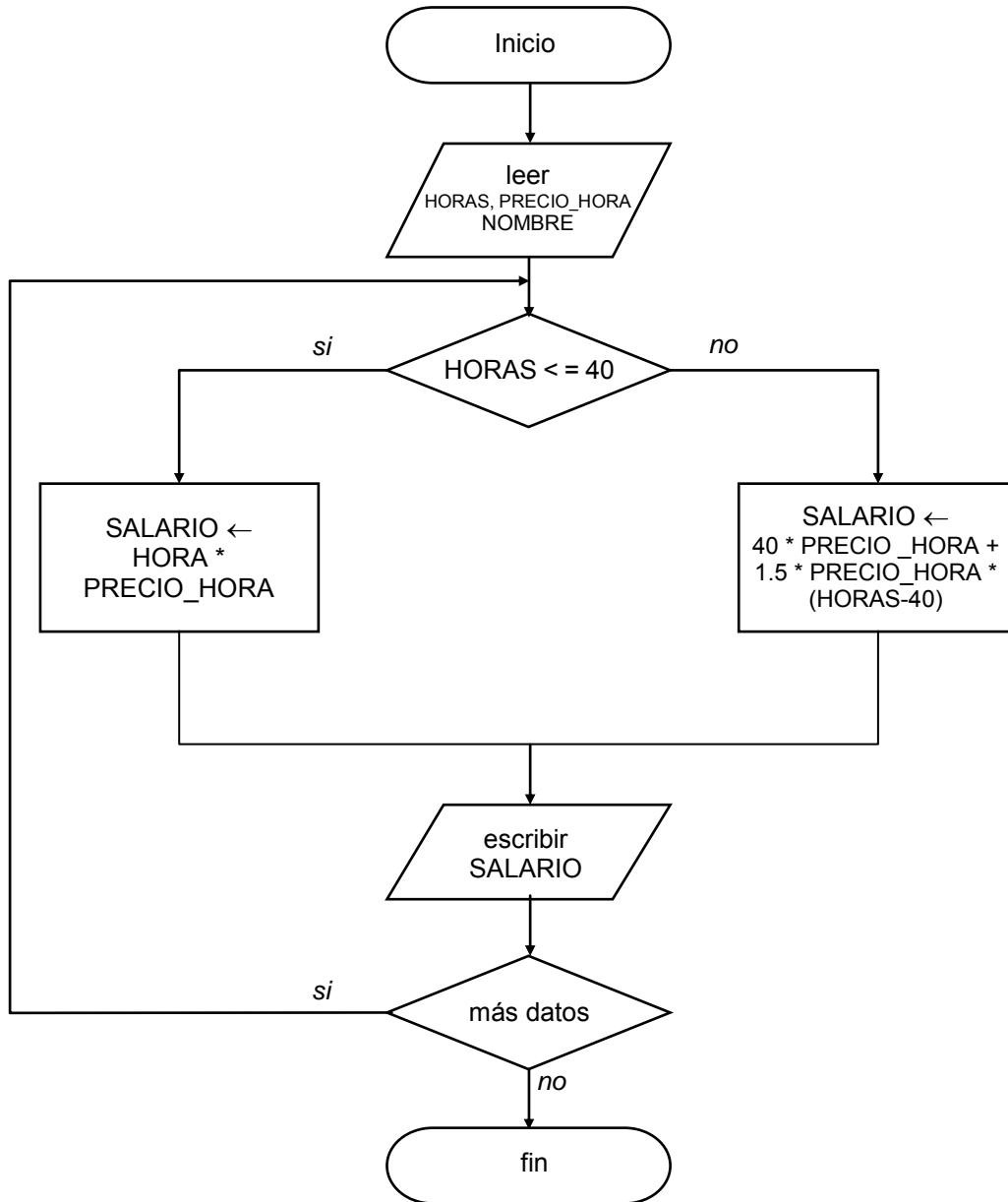
¿Lo lograste?

En efecto, el diagrama de flujo anterior suma números pares comprendidos entre 2 y 100.

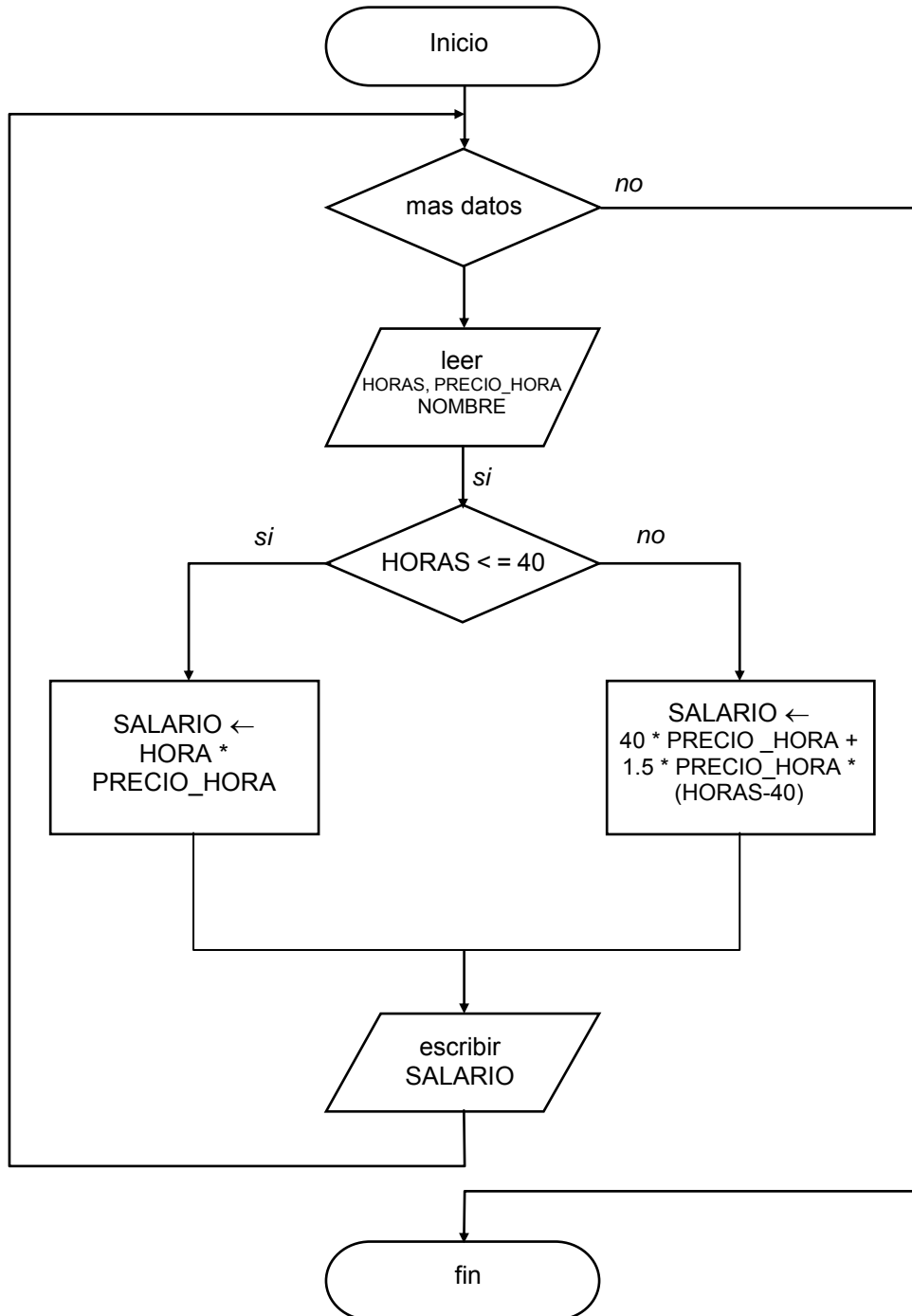
Ejemplo 3:

Se desea realizar el diagrama de flujo (algoritmo) que resuelve el siguiente problema.

Cálculo de los salarios mensuales de los empleados de una empresa, sabiendo que éstos se calculan en base a las horas semanales trabajadas y de acuerdo a un precio especificado por hora. Si se pasan de 40 horas semanales, las horas extraordinarias se pagarán a razón de 1.5 veces la hora ordinaria.



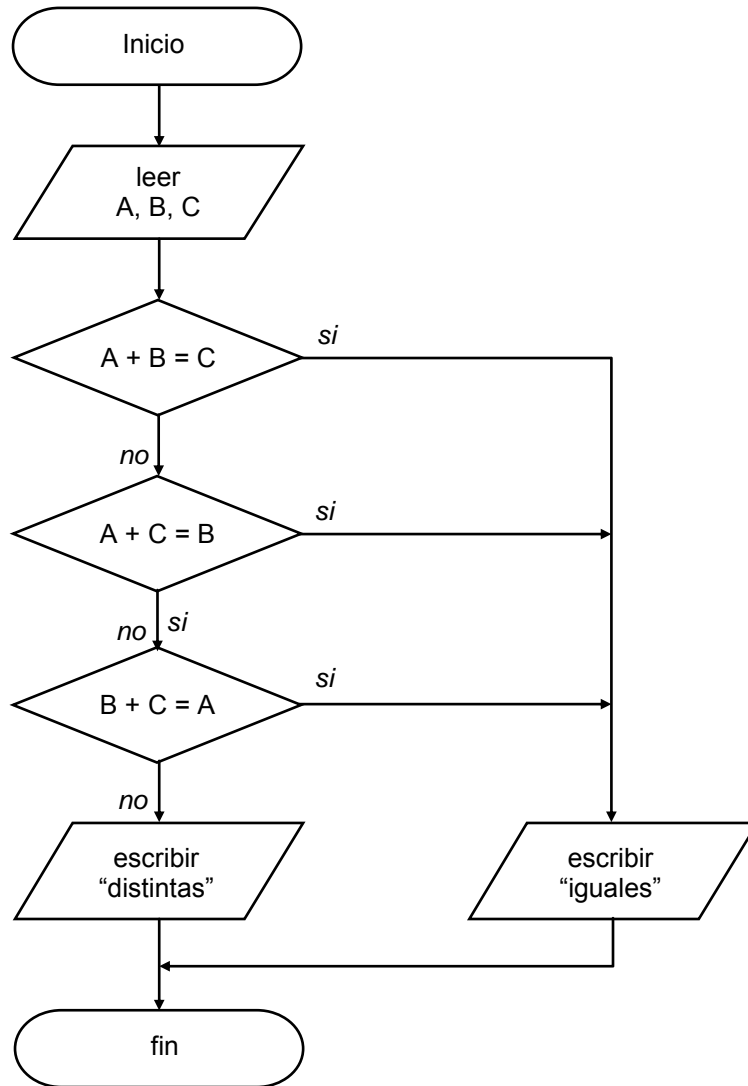
El siguiente diagrama de flujo tiene la misma función que el anterior.



Ejemplo 4:

El diagrama de flujo que se presenta a continuación, realiza lo siguiente.

Dados 3 números, determina si la suma de cualquier pareja de ellos es igual al tercer número. Si se cumple esta condición, escribe "iguales" y, en caso contrario, escribe "distintas".



ACTIVIDAD DE REGULACIÓN

Para que confirmes e integres los conocimientos que has alcanzado hasta este momento. Realiza la siguiente actividad. Para la solución de este ejercicio puedes retomar como modelos, los ejemplos que acabas de estudiar.

1. Realiza el Diagrama de Flujo que represente el cambio de una llanta para un automóvil.



COLEGIO DE BACHILLERES

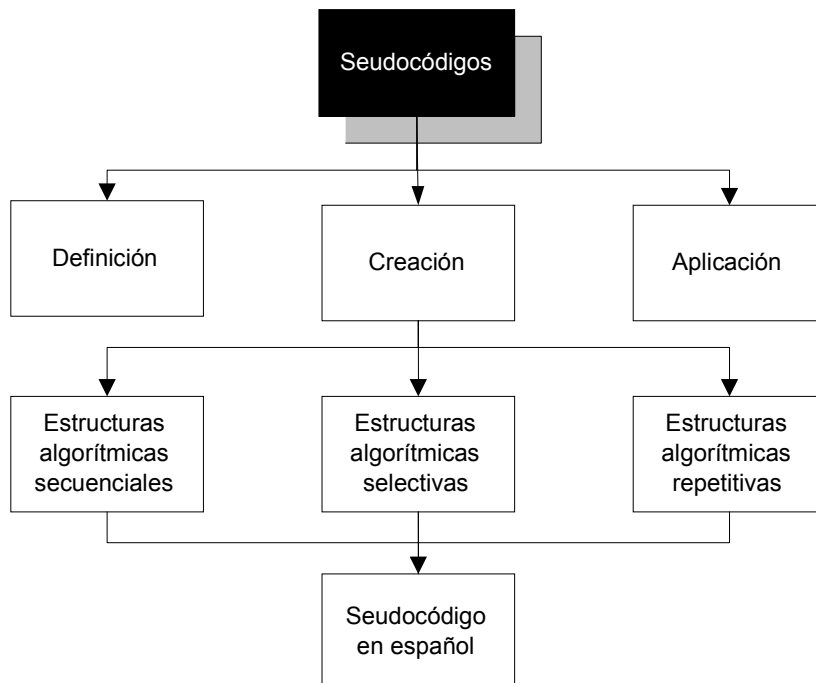
**LÓGICA COMPUTACIONAL
Y
PROGRAMACIÓN**

FASCÍCULO 3. SEUDOCÓDIGOS

SEUDOCÓDIGOS

OBJETIVO: Elaborarás pseudocódigos, mediante su diseño y construcción, lo que te permitirá establecer las bases para programar en cualquier lenguaje.

Este fascículo se encuentra organizado de la siguiente forma:



3.1 DEFINICIÓN.

El **seudocódigo** es un lenguaje de especificación, es decir de la descripción de algoritmos y diagramas de flujo, permitiendo pasar a la traducción de un lenguaje de programación.

El seudocódigo nació como un lenguaje similar al inglés y es un medio de representar las estructuras de control, es un primer intento para llegar a un lenguaje de programación, sin embargo, actualmente se utiliza en el idioma español.

Un seudocódigo está compuesto de frases escritas en lenguaje natural, también llamado español estructurado.

En la conversión (transformación) de los diagramas de flujo a seudocódigos es fundamental el uso de las llamadas estructuras de control o algorítmicas, las cuales son manejadas dentro de la técnica de programación estructurada.

El seudocódigo traduce todos los símbolos y funciones representados en el diagrama de flujo, para ser traducidos finalmente en un lenguaje de alto nivel.

Con base en esto tenemos que:

- > El seudocódigo es compacto.
- > Puede modificarse fácilmente.
- > Se utilizan palabras claves en mayúsculas en español, que identifican las estructuras algorítmicas secuenciales, selectivas y repetitivas.
- > Se construye de manera estructurada (de arriba hacia abajo).
- > No existen reglas estándar para utilizarlo.
- > No se tiene una representación gráfica de la lógica del programa,
- > No puede ser representado en una computadora.

La ventaja de realizar el seudocódigo, es que el programador se concentra en la lógica y en las estructuras de control, no así en las reglas de un lenguaje específico.

3.2 TRANSFORMACIÓN DE ESTRUCTURAS ALGORÍTMICAS A SEUDOCÓDIGOS EN CASTELLANO

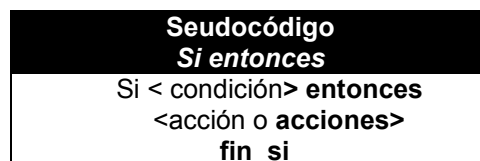
Seudocódigos de las estructuras algorítmicas:

- 1) **Estructura Algorítmica Secuencial:** Muestra que de una acción le sigue otra, y así sucesivamente:

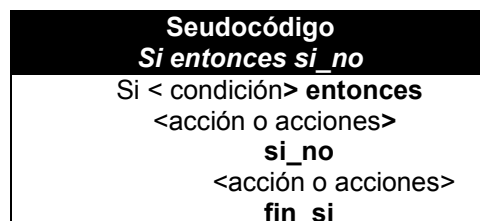


- 2) **Estructura Algorítmica Selectiva:** Usada cuando se quiere tomar una decisión lógica.

- a) **Si entonces (if then) o alternativa simple.** Se ejecuta una acción cuando se cumple una condición. Si al evaluar la es verdadera, entonces se ejecuta la acción o acciones.



- b) **Si – entonces – sí – no (it then else) o alternativa doble.** Permite elegir entre dos opciones o alternativas posibles, si al elegir la(s) condición(es) el resultado es verdadero, se sigue por el camino especificado y se procede a ejecutar la(s) acción(es). Pero si el resultado de la condición es falso, entonces se sigue el otro camino y se procede a ejecutar la(s) otra(s) acción(es).



- c) **Múltiple (según sea, caso de / case).** Permite que el flujo del diagrama se divida por varios caminos en el punto de la toma de decisión(es), dependiendo el valor que tome el selector, es la acción que se ejecutará. Si el selector toma el valor de 1 se ejecutará la acción 1, así si el selector toma el valor n, se ejecutará la acción n.

Seudocódigo <i>Caso</i>
Caso variable (C) C1: <acción o acciones> C2: <acción o acciones> Cn: <acción o acciones> si_no <acción o acciones> fin_caso

3) **Estructura Algorítmica Repetitiva:**

- a) **La estructura repetitiva mientras (en ingles while o dowhile).** Es aquella que se ejecutará mientras la condición evaluada resulte verdadera.

Seudocódigo <i>Mientras</i>
Mientras <condición>haz <acción 1> <acción 2> : <acción Sn> fin_mientras

- b) **La estructura repetitiva repetir (en inglés repeat).** Se ejecuta un número definido de veces, y se va a ejecutar hasta que se cumpla una condición determinada que se comprueba al final del bucle.

Seudocódigo <i>Repetir</i>
Repite <acción o acciones> hasta <condición>

- c) **La estructura desde/para (en inglés for).** Se utiliza cuando se conoce el número de veces que se desea ejecutar las acciones de un bucle, en el que el número de iteraciones es fijo:

Seudocódigo		
<i>Desde</i>		
Para v:=v, hasta v, (incremento Decremento <valor<)		
<acción o acciones>		
fin_para		
donde v: variable control	v,: valor inicial	v,: valor final

3.3 APLICACIÓN DE SEUDOCÓDIGOS

Seudocódigo de la suma de dos números

Comienza

Lee (N1)

Lee (N2)

Suma = N1 + N2

Escribe (Suma)

Termina

- Ω **Usando la estructura algorítmica selectiva:** Seudocódigo para calcular la media aritmética de una serie de números positivos.

Seudocódigo de la media aritmética

Comienza

s = 0

c = 0

datos:

Leer (X)

Si X < 0 entonces

Ir_ a media

Si_no

C = c + 1

S = s + x

Ir_ a datos

Fin_si

Media:

m = s / c

Escribir (m)

// Cálculo de la media

Termina

Nota:

Sea

s (suma)

c (contador de números)

M (media)

Ω Usando la estructura algorítmica selectiva:

Dado un número del 1 al 7, decir qué día de la semana es, partiendo de que el lunes es 1.

Seudocódigo, leída una fecha decir el día de la semana

Comienza

Escribe ('día')
Lee día

Nota:
Consideramos que día es un número entero

Caso día:

1: Escribe ('Lunes')
2: Escribe ('Martes')
3: Escribe ('Miércoles')
4: Escribe ('Jueves')
5: Escribe ('Viernes')
6: Escribe ('Sábado')
7: Escribe ('Domingo')

Fin_caso

Termina

Ω Usando la estructura algorítmica repetitiva: Seudocódigo para obtener la suma de los números pares hasta 700 inclusive.

Seudocódigo para obtener la suma de los números pares hasta 700 inclusive, utilizando la estructura Mientras.

Comienza

Suma=0
Num=2

Nota: con mientras
Consideramos que Num y Suma son números reales.

Mientras Num <=700 haz

Suma = Suma + Num
Nun = Num + 2

Fin_mientras

Termina

ACTIVIDAD DE REGULACIÓN

Para que confirmes e integres los conocimientos que has alcanzado hasta el momento. Realiza la siguiente actividad.

1. Realiza el Seudocódigo que nos represente el cambio de una llanta de automóvil.



COLEGIO DE BACHILLERES

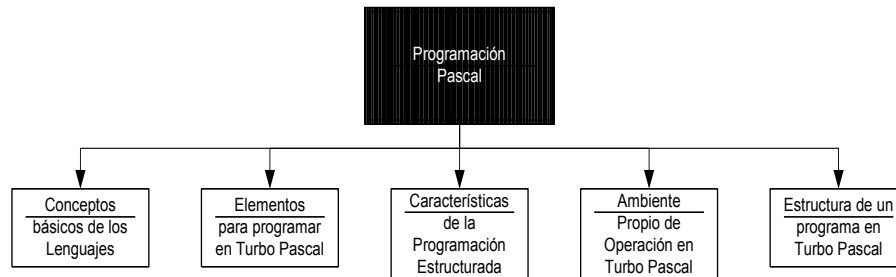
**LÓGICA COMPUTACIONAL
Y
PROGRAMACIÓN**

FASCÍCULO 4. PROGRAMACIÓN PASCAL

PROGRAMACIÓN PASCAL

OBJETIVO: Aplicarás los elementos básicos de la programación en el lenguaje Pascal, construyendo programas elementales; lo que te permitirá el desarrollo de habilidades de razonamiento lógico en la solución de problemas.

A continuación te presentamos la forma en que se encuentra organizado este fascículo:



4.1 CONCEPTOS BÁSICOS DE LOS LENGUAJES

Cuando se aplica cualquier metodología para la resolución de problemas, con la intención de presentar un programa que se pueda ejecutar en la computadora, forzosamente tendremos que elegir el lenguaje de programación a utilizar, en nuestro caso, un lenguaje de cuarta generación como lo es Pascal.

Es importante aclarar que la computadora tiene su propio idioma, de tal forma que tendremos que considerar los diferentes lenguajes que existen, tales como:

Lenguaje máquina. Serie de instrucciones directamente entendibles por la computadora, pero de muy difícil manejo para cualquier programador.

Lenguaje de bajo nivel. Conjunto de instrucciones específicas entendibles por la computadora. Las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos.

Lenguaje de alto nivel. Serie de instrucciones escritas con palabras muy similares al idioma propio, esto implica que la computadora requerirá de un traductor para entender convertirlo al lenguaje máquina, el cual es el único idioma que interpreta la computadora.

Lo anterior requiere que manejes los elementos mínimos necesarios para manipular el ambiente, lo que implica el uso de menús, operaciones con los bloques de instrucciones, así como la forma de imprimir tus códigos y pantallas de ejecución.

Pascal es por naturaleza un lenguaje de programación estructurado, por lo cual en el podrás codificar programas modulares con una metodología de programación estructurada.

En algunos casos se requiere que la información procesada dentro del programa se almacene para su posterior tratamiento, por lo cual se tendrán que definir estructuras de datos especialmente para esto.

Los archivos que revisarás son los de tipo texto y los estructurados, donde en el primero la forma de almacenamiento es mediante código ASCII, mientras que en el segundo es mediante el uso de registros.

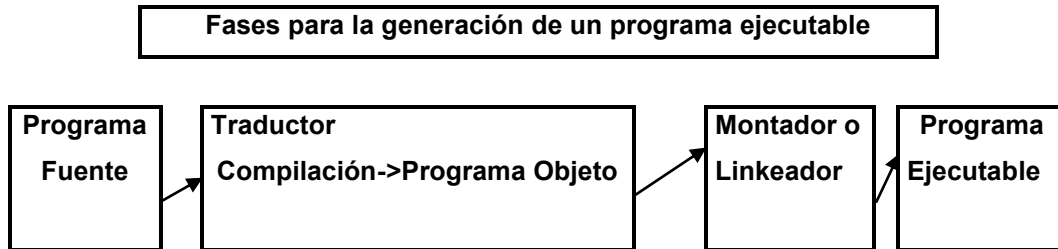
El Lenguaje Pascal fue creado por el matemático Suizo Nicklaus Wirth en 1970 es adecuado e innovador para la enseñanza de los conceptos, métodos y técnicas de programación, utilizado en todo tipo de aplicaciones, posee grandes facilidades para la programación de sistemas y diseño gráfico. Asimismo utiliza los conceptos de tipo de datos, programación estructurada y diseño descendente, entre otros.

Cuando ya se tienen los pseudocódigos el siguiente paso es codificar (trasladar a una escritura en un lenguaje de programación de alto nivel) y elaborar un **programa**, el cual es una serie de instrucciones de cómputo estructuradas y ordenadas de tal manera que al ser ejecutadas, hace que la computadora efectúe una función determinada.

La estructura de un programa se determina al realizar un proceso sobre unos datos de entrada para obtener unos resultados que den respuesta al problema planteado. Es

decir, que las instrucciones que forman el programa afectan a la entrada de datos, a su proceso o a la salida de resultados.

Para la elaboración y la explotación de programa se deben seguir algunos pasos como son:



- 1) **Programa fuente**; es el código original de un Programa. Aquí se transcribe el pseudocódigo al lenguaje de alto nivel Pascal, en la computadora (creación del programa), guardándolo en la memoria auxiliar (disco) a través del editor de programa.
- 2) **Traductor**: es un programa que traduce las palabras de un lenguaje de alto nivel (programa fuente) al lenguaje de máquina (sistema binario), existen dos tipos de traductores. Los intérpretes y los **compiladores**, siendo éste último el que utiliza Pascal. Los compiladores traducen todo el programa fuente (si es que no existen errores de sintaxis) generando otro **programa** llamado **objeto**, el cual es la traducción al lenguaje máquina línea por línea.
- 3) **Montador**: Es un programa que complementa al traductor se utiliza cuando el programa objeto necesita de una preparación y agregar algunas rutinas de una preparación y agregar algunas rutinas del propio lenguaje, enlazando todas las líneas generadas en dicho programa.
- 4) **Programa ejecutable**: Programa independiente del lenguaje de programación, es decir, el paso real para hacer un programa funcional es enlazarlo (linkearlo/linker) y convertirlo a un **archivo ejecutable** con extensión **.EXE** o **.COM**. En el caso del lenguaje Pascal la compilación no genera un programa objeto en disco y el "linkeamiento" se realiza al enlazar todas las líneas generadas en memoria principal por compilación. Esta tarea se realiza de forma automática mediante el **compilador** del lenguaje, el cual requiere que la traducción sea a disco para generar el ejecutable

INSTALACIÓN DE TURBO PASCAL

Para la instalación del software (Turbo Pascal) en nuestra máquina, implica copiar en el disco duro los archivos de los programas y datos que nos proporciona el distribuidor generalmente el software de distribuye en CD ROM (si es complejo y ocupa mucho espacio) o discos flexibles de 3.5".

Instalar Turbo Pascal es fácil, sólo necesitamos colocar el disco de instalación en la unidad correspondiente, teclear install o instalar y presionar Enter; a continuación aparecerán las instrucciones del programa de instalación de Turbo Pascal, las cuales tendremos que seguir al pie de la letra para lograr una instalación correcta. Se requiere un espacio mínimo de 3.2Mb de disco libre para instalar el programa completo, si se requiere por algún motivo que la instalación sea más básica, podemos decirle al programa- que no instale los ejemplos, con esto se logra un ahorro de 700K de espacio en disco. Además se puede quitar de la instalación los archivos que se instalaron en el directorio BTURB03 si no se va a trabajar con Turbo Pascal 3, el cual se incluye para que exista compatibilidad con esta versión o anteriores.

Si se está instalando en Windows, se selecciona la barra de tareas en INICIO y se va a Ejecutar, de ahí se escribe la unidad donde esta el disco y el nombre del archivo a instalar, A: o B:\instalar (A: o B:\install, si el programa está en inglés).

ESTRUCTURA GENERAL DE UN PROGRAMA EN PASCAL

¿Cómo se compone un Programa en Pascal?

Un programa escrito en un lenguaje de programación requiere de dos elementos estructuras de datos e instrucciones.



Turbo Pascal es un lenguaje de programación estructurado, por lo cual se pueden transcribir pseudocódigos que fueron descritos con la programación modular y estructurada, esta tendencia facilita la escritura y lectura de programas, además de que este lenguaje es de los más flexibles.

4.2 ELEMENTOS PARA PROGRAMAR EN TURBO PASCAL

- Un programa en Turbo Pascal requiere de la definición de varios módulos que pueden ser o no ser omitidos en la codificación, los cuales son los siguientes:
- Al inicio de cada programa se debe de incluir la sentencia **PROGRAM** <Identificador>, lo cual indicará el nombre que se le dará al programa (Puede ser omitida).
- Declaración de **USES <librerías>**, sirve para poner en uso algunas librerías o bibliotecas, las cuales tienen funciones predefinidas que permiten ahorrar código (puede ser omitida).
- Modulo de **CONST**, donde se definen las constantes (puede ser omitida, pero no es recomendable).
- Bloque de **VAR**, donde se declaran las variables a utilizar, con toda certeza se podría decir que es un bloque que siempre deberá de incluirse en cualquier programa.
- Declaración de **FUNCIONES Y PROCEDIMIENTOS** (puede ser omitida).

Por último, un programa debe estar con un cuerpo de instrucciones principales, las cuales, no se omiten. En Turbo Pascal, este cuerpo principal de Instrucciones se encuentra agrupado por las palabras **BEJÍN** y **END**.

El lenguaje de programación Pascal utiliza **palabras reservadas**, esto es, son palabras que ya cuentan con un significado predeterminado, y por lo tanto, no debemos utilizar.

Algunas consideraciones e instrucciones que nos permitirán construir nuestro primer programa son:

Zona de Declaración de Programas o Encabezado

La primera línea del programa es el encabezado, el cual debe iniciar con la **palabra reservada PROGRAM** seguida de un **nombre** que es elegido por el programador, y nos debe indicar cuál es la función del programa, este nombre consta de uno a ocho caracteres, siendo el primero una letra y los restantes letras/dígitos.

Ejemplo:

```
PROGRAM SUMA.
```

En la Zona de declaraciones.

A partir de la **segunda línea se declaran las variables**, las cuales le indican al compilador el tipo de datos que van a ser usados en el programa. Siempre estarán en mayúsculas. Aquí se decide que tipo de valor tomarán las variables: enteras, reales, booleanas, de tipo carácter, etc., esto depende de las necesidades que se tengan. Cualquier nombre que se utilice dentro de un programa se le llama identificador.

El nombre de la variable o identificador debe empezar con una letra y puede contener solamente letras, dígitos numéricos y el signo (_). La longitud del nombre no debe ser mayor de 63 caracteres, ya que el compilador sólo respetará los 63 primeros.

Ejemplo de nombres de variables: pulgada, metros, iva5, prom_calif, p34

Ejemplos no válidos de nombre de variables: 2000conta, prom,calif, calif total.

Esta zona tiene cinco secciones que deben aparecer en la secuencia:

LABEL (declaración de etiquetas)

CONST (declaración de constantes)

TYPE (declaración de escalares)

VAR (declaración de variables)
(declaración de subprogramas)

En donde LABEL, CONST, TYPE, VAR son palabras reservadas.

Si no se ocupan las cinco secciones pueden quitarse las que no sean necesarias en el programa, pero deberá conservarse el orden dado.

Ejemplo:

```
TYPE
        Semana=(LUN,MAR.MIER,JUE,SAB,DOM);
        Dialab=LUN..VIE;
```

```
VAR      Suma:Integer;
```

En la Zona de Declaración de Procedimientos ¹

Un procedimiento es un subprograma, es decir un programa-pequeño dentro de otro programa, donde el programa hace un llamado a éste, él cual tiene la función de realizar una tarea específica. Al ser un procedimiento un conjunto de instrucciones utilizadas muy comúnmente en nuestro programa, es conveniente agruparlas con un nombre, de manera que si se necesita posteriormente sólo será necesario hacer la llamada al procedimiento correspondiente mediante su nombre.

```
PROCEDURE NOMPROC (<lista Parámetros>); {comentarios}
    <DECLARACIONES VAR, TYPE, USES, ETC>
BEGIN
    <CUERPO DEL PROCEDIMIENTO>
END;
```

NOMPROC el nombre del procedimiento, <lista Parámetros> pueden ser por valor, variable o ambos, pero deben corresponder en número y tipo a los parámetros de la sentencia que hace la llamada al procedimiento. En esta zona de declaración de variables <DECLARACIONES VAR, TYPE, USES, ETC> se deben declarar las constantes, tipos variables, las unidades uses, en forma análoga al programa principal.

Parámetros: En la declaración de un procedimiento y función se debe especificar una lista de parámetros (que no son más que datos de entrada o de salida del módulo) Los tipos principales de parámetros son:

a) **Por Valor:** Son datos de entrada a un módulo (función o procedimiento) que pueden variar dentro de éste, pero al terminar su tarea, los parámetros regresaran a su valor de entrada, de tal forma que no se ven afectados por su ejecución se caracterizan por **NO** ir precedidos de la palabra **VAR** en la lista de parámetros del procedimiento.

Ejemplo:

```
Procedure suma1(a,b:integer);
Var
    Tem:integer;
```

b) **Por referencia:** Son datos de entrada y salida de un módulo (función o procedimiento), los cuales pueden variar dentro de éste, y al terminar su tarea los parámetros conservarán los nuevos valores adquiridos, de tal forma que afecten al módulo y/o programa que realizó la llamada, se caracterizan por ir precedidos de la palabra **VAR** en la lista de parámetros del procedimiento.

Ejemplo:

```
Procedure suma2(var a,b:integer);
Var
    Tem:integer
```

¹ Algunas veces en un programa no se necesita la zona de declaración de procedimientos y funciones, por ello decimos que algunos programas están compuestos de 3 zonas. En cada zona no hay número específico de líneas, eso dependerá de lo que se necesite. Lo que se debe de considerar es el orden en que se presentan las zonas, para evitar errores al momento de compilar el programa.

En la Zona de Declaración de Funciones

Una función es un subprograma que nos permite calcular y regresar un sólo resultado, basado en uno o más valores que se le pasan. Si quisiéramos evaluar constantemente una operación matemática compleja que sea de tipo entera o real, es conveniente hacer uso de éstas. Existen en Pascal varias funciones como son: `sqr`, `sqrt`, `abs`, `round`, `ord`, etc, pero también se pueden definir de acuerdo con las necesidades del usuario.

FUNCTION NOMFUNC(<lista Parámetros>):TIPFUNC; {encabezado}

<DECLARACIONES VAR, TYPE, USES ETC>

BEGIN

<CUERPO DE LA FUNCIÓN> {calcular resultado}

NOMFUNC:=<VARIABLE>; {asignar resultado a nomfunc}

END;

En el encabezado debe empezar con la palabra **FUNCTION**, seguida del nombre de la función, una lista de parámetros formales y la definición del tipo de dato que es la función. En la zona de declaración de variables <DECLARACIONES VAR, TYPE, USES, ETC> se deben declarar las constantes, tipos, variables, las unidades uses, en forma análoga al programa principal. El cuerpo de una función debe calcular el resultado y asignar dicho resultado al nombre de la función (nomfunc).

Llamado a Función: Una función calcula y devuelve un valor único, un llamado a función puede aparecer cada vez que sea permitido utilizar una expresión.

Existen tres aplicaciones de llamado a función:

- a) A la derecha de un enunciado de asignación.
- b) Como un elemento dentro de la orden `writeln`.
- c) Como parte de una condición booleana.

En la Zona de Programa Principal o Cuerpo del Programa.

Aquí ponemos las instrucciones que son necesarias para solucionar el problema, las cuales pueden ser asignaciones hasta llamadas a procedimientos y/o funciones, procurando que esta sección tenga la menor parte de instrucciones. En esta zona debe empezar con la palabra reservada **BEGIN** y continúa con instrucciones, que contienen las acciones a realizar por el programa, al terminar la serie de instrucciones con la palabra reservada **END**.

Ejemplo:

Begin

X:=7;

End.

Esta zona es la parte activa, ya que todas las secciones anteriores aquí se ponen a trabajar. Es recomendable que el programa principal contenga pocas líneas y esto se logra si fue bien diseñado.

ALGUNAS INSTRUCCIONES USADAS EN LA PROGRAMACIÓN PASCAL

Unidades Predefinidas

Las Unidades son funciones predefinidas que realizan una tarea específica las cuales pueden ser llamadas desde cualquier programa.

Turbo Pascal utiliza seis unidades que se encuentran en el archivo turbo.tpl, las cuales son:

- a) **System**: Nos permite enlazar a todos los programas y contiene todos los procedimientos y funciones.*
- b) **Crt**: Nos permite activar declaraciones de entrada y salida, y la usamos para direccionar y limpiar datos en pantalla.
- c) **Printer**: Nos permite direccionar las salidas a la impresora mediante la declaración Ist.
- d) **Graph3**: En esta unidad, contiene las funciones específicas para el manejo de gráficos, el tres se refiere la versión tres de Turbo Pascal.
- e) **Dos**: Contiene los procedimientos, funciones y tipos de datos para poder hacer programas de bajo nivel con MS-DOS.
- f) **Turbo3**: Contiene las variables y subprogramas de turbo 3.

Estas unidades se pueden utilizar en el programa, se declaran en la zona de declaración de unidades o librerías, usando la palabra reservada `uses` seguida de nombre de la unidad o unidades a usar.

Ejemplo:

```
Uses printer;
```

O bien

```
Uses printer, crt;
```

Normalmente la palabra **uses** sigue inmediatamente después del encabezado del programa.

En el cuerpo del programa se hace el llamado a esa unidad.

- En el caso de la unidad **System** no es necesario declararla, porque ésta se enlaza a todos los programas.
- Si se requiere compilar un programa que maneje gráficos es necesario que la unidad `graph` sea declarada en los `uses`.

Funciones predefinidas en la unidad Crt.

Clrscr	Limpia la pantalla de salida y coloca el cursor en la esquina superior! derecha del monitor.
GotoXY(a.b)	<p>Permite posicionar el cursor en la columna y renglón requerido. Si la pantalla de salida contiene 80 columnas y 25 renglones, podemos posicionar el cursor donde se requiera. Esta instrucción toma dos valores el de la columna (a) y el renglón (b).</p> <p>Ejemplo: GotoXY(50,18); Write('HOLA');</p> <p>Con estas instrucciones se moverá el cursor al renglón 18 y a la columna 50 y desplegará la palabra Hola*.</p> <p>*ver instrucción writein.</p>
Variable:=X	<p>Llamado enunciado de asignación (:=), permite asignar un valor a una variable. La forma general de un enunciado de asignación es:</p> <p style="text-align: center;">Variable:=expresión</p> <p>En el enunciado de asignación se evalúa la expresión de la derecha y se le asigna el valor del resultado a la variable que se encuentra a la izquierda del símbolo. La expresión puede ser una constante, una variable, o una expresión aritmética.</p> <p>Para el uso de las expresiones aritméticas en Pascal, la multiplicación utiliza el símbolo * y para la división el símbolo /.</p>
Punto y coma	<p>Define el término de una sentencia o instrucción. Solamente en las siguientes instrucciones se omite el punto y coma:</p> <ul style="list-style-type: none"> a) Begin b) El último End del programa principal (debe llevar punto) c) Los comentarios (agrupados por los siguientes caracteres (*Comentario*) o {comentario}) d) La línea anterior a cualquier end (opcional)
Comas	Se utilizan en el programa para separar los elementos como: las variables, constantes, mensajes a pantalla o impresora, operaciones con archivos, entre otros.
Punto	<p>En Pascal es importante la puntuación que se utiliza en el programa.</p> <p>Debe existir un punto final en el último End, indicando el final del programa.</p>

Writeln o write	<p>Con el writeln la computadora despliega lo que se encuentra entre paréntesis y se coloca al inicio de la siguiente línea en la pantalla se puede utilizar write o writeln, la diferencia es que write despliega lo que se encuentra entre paréntesis, dejando el cursor al final de éste.</p> <p>También:</p> <p>a) Nos permite enviar información a la pantalla, impresora o un archivo.</p> <p>b) Se puede utilizar para desplegar un mensaje, este se encierra entre apóstrofes, para abrir y cerrar el mensaje. Writeln('Prueba')</p> <p>c) Despliega el valor de una variable o expresión Writeln(a) considerando a:=5 entonces la salida es: 5 Writeln (a+9) la salida es 14</p> <p>d) Puede combinar mensajes, variables y expresiones en cualquier orden. Se necesita utilizar una coma para separar cada una de las partes. Writeln('El valor de la suma es ',suma)</p> <p>e) Cuando el writeln se utiliza solo, nos permite llevar el cursor a la primera columna de la siguiente línea. Esto es que provocará que el cursor salte una línea.</p> <p>f) Si queremos que la salida de un programa se dé en papel deberá utilizar la unidad printer (declararía en la zona de variables) y en el cuerpo del programa enunciarla así: Write(1st,...) o bien writeln(1st,...)</p> <p>Para llevar el carro de impresión al inicio de la siguiente línea es indispensable utilizar writeln en blanco Writeln;</p>
------------------------	--

Readln(x);	<p>Esta instrucción nos permite leer datos del teclado, pide un dato y te posiciona en la primera columna del siguiente renglón. Estos datos se guardan temporalmente en la variable que está dentro del paréntesis</p>
-------------------	---

Comentarios	<p>Se utilizan para dar mensajes en el programa, en ellos se puede" expresar el nombre del programador, la fecha, el objetivo, el tipo de archivo o cualquier otro tipo de comentario. En el programa se utiliza un par de llaves {} ó (* *). Dentro de ellas se pone el comentario no es necesario que lleve punto y coma al final, porque el compilador no traduce la información contenida en los comentarios.</p>
--------------------	---

<p>Espacios en blanco</p>	<p>Se pueden utilizar los espacios en blanco para separar dos identificadores en el programa utilizando una coma, pero no deben colocarse dentro del identificador, ya que el compilador los identificará como dos nombres diferentes.</p> <p>Se recomienda dar espacios en forma de sangría para que el programa tenga más presentación, es decir en el cuerpo del programa, después de la palabra Begin, recorrer las instrucción de manera que se distingan.</p> <p>Si se usan estructuras algorítmicas de control, recorrer la primera instrucción de la estructura de inicio y poner a esa misma distancia su instrucción final, en la parte interna de estas instrucciones, volver a dar otra sangría.</p>
<p>Unidad GRAPH;TPU Y el archivo TURBO:TPL</p>	<p>Si se requiere compilar un programa que maneje gráficos es necesario tener la unidad Graph.tpu y el archivo Turbo.tpl. También es necesario tener un adaptador gráfico (VGA, EGA, CGA o Hércules) y el controlador de Borland BGI (Borland Graphic Interface) los cuales deberán estar en el mismo compilador de Turbo Pascal. Todos los programas gráficos deben declarar la unidad Graph en su sección uses.</p>
<p>InitGraph</p>	<p>Este procedimiento y detect pertenecen a la unidad graph, y son las instrucciones de inicio de un programa gráfico.</p> <pre>ControlGr:=detect; LnitGraph(ControlGr,ModoGr,э э)</pre> <p>Estas instrucciones nos permiten inicializar en Turbo pascal en el modo gráfico de más alta resolución para el tipo de adaptador que se tenga.</p> <p>Para la salida de un programa gráfico se utilizan las dos instrucciones siguientes:</p> <pre>Readin; Closegraph;</pre> <p>El readin nos permite mantener la gráfica en la pantalla hasta que el usuario presiona enter. Closegraph regresa a la computadora al modo de texto.</p>

FASES EN EL DISEÑO DE UN PROGRAMA

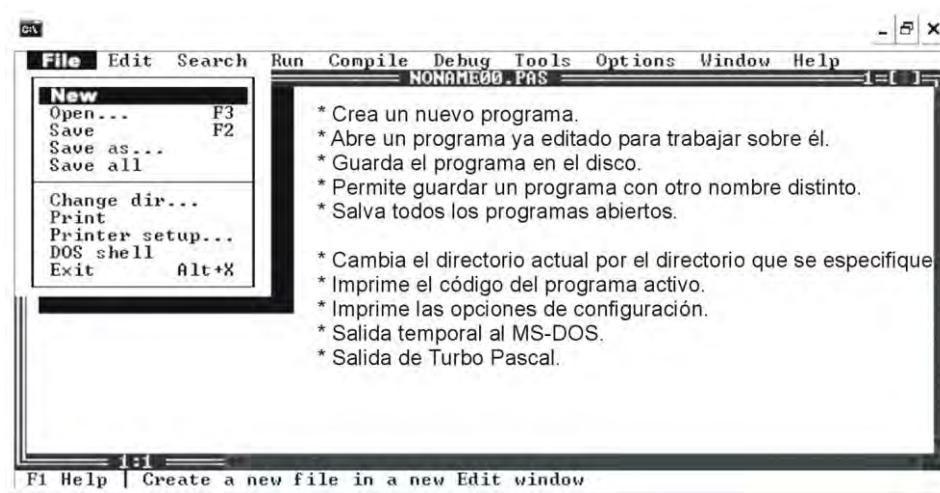
Al elaborar un programa como producto final de un proceso debe seguir ciertos pasos:

- 1) **Diseño de Sistemas:** Se lleva a cabo el **análisis del problema** con el objeto de obtener una representación mental de los elementos del mismo y el **desarrollo de la solución** dividiendo estos elementos, en partes más simples hasta llegar a un nivel de gran detalle.
- 2) **Codificación:** De la fase anterior surge un programa abstracto, el cual se traduce o se implementa en un lenguaje de programación. El proceso de codificar un programa en un lenguaje de programación es totalmente directo.
- 3) **Prueba:** Ya implementada la solución presentada, es necesario que sea probada para asegurarse que resolverá correctamente el problema original.
- 4) **Mantenimiento:** Aquí se realizan las modificaciones que son necesarias cuando cambian algunas características del problema que no implican un cambio en el diseño original del programa. Este mantenimiento se debe llevar a cabo durante toda la vida útil del sistema.

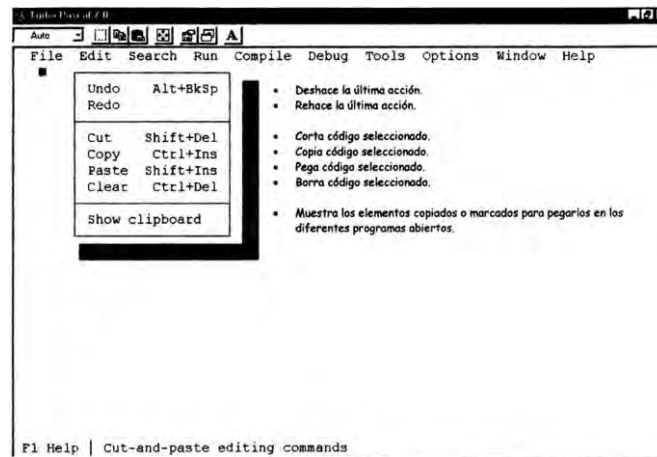
4.3 AMBIENTE PROPIO DE OPERACIÓN EN TURBO PASCAL

Para empezar a construir nuestro primer programa, necesitamos entrar al editor de Turbo Pascal. Las principales opciones que ofrece el ambiente Turbo Pascal que más comúnmente utilizaremos son:

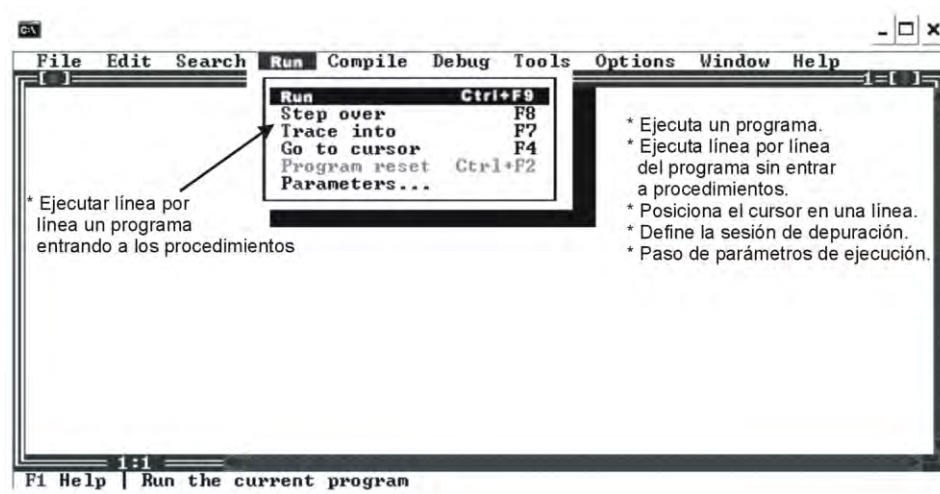
MENÚ FILE



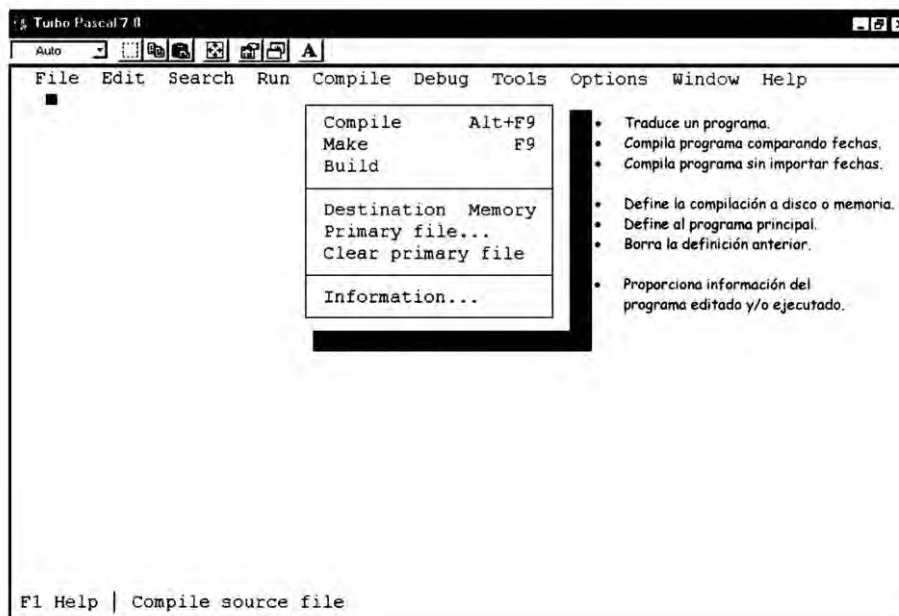
MENÚ EDIT:



MENÚ RUN:



MENÚ COMPILE:



4.4 ESTRUCTURA DE UN PROGRAMA EN TURBO PASCAL

Analizaremos en detalle las reglas de estructura que se deben seguir siempre que se escriben programas en turbo pascal.

La estructura de un programa en turbo pascal es la siguiente:

Program	identificador	[cabecera del programa]
Uses	identificadores	
Label	lista de etiquetas	[sección de etiquetas]
Const	definición de constantes.	
Type	declaración de tipos de datos	definidos por el usuario.
Var	declaración de variables.	
Procedure	definición de procedimientos	
Function	definición de funciones	
Begin	Sentencias	[cuerpo del programa]
End.		

Programa Demo.

Program DemoPrimero. (cabecera)

(Este es nuestro primer programa en turbo pascal)

Uses

Crt, Dos; (declaraciones)

Const

Pi= 3.141592;
Iva= 12;

Type

Palabra = string [20];
Iva = 12;

Var

Salario : real;
Numero : integer;

Begin

```
CirScr; {borra o limpia la pantalla}
WriteLn ('¿Cual es su primer apellido?');
ReadLn(Apellido);
WriteLn(' Escriba un numero Sr: ', Apellido);
ReadLn(Numero);
WriteLn(' El cuadrado del numero es ', numero * numero);
End. {fin del cuerpo del programa}
```

Ejemplo:

Dado tres números enteros, determinar cual es el mayor (los tres números se suponen distintos)

Program El_numero_Mayor;

Var

```
A, B, C: integer;
mayor : integer;
```

Begin

```
CirScr;
```

```
{introducción de datos en memoria}
```

```
WriteLn(' Introduzca los tres números enteros ');
ReadLn(A,B,C);
```

```
{determinar el número mayor}
```

```
IF A>B THEN
```

```
  IF A>C THEN
```

```
    mayor := A
```

```
  ELSE
```

```
    mayor := C;
```

```
ELSE
```

```
  IF B>C THEN
```

```
    mayor := B
```

```
  ELSE
```

```
    mayor := C;
```

```
{visualización del número mayor}
```

```
WriteLn ('El número mayor es:', mayor);
```

End.

```

Program Pascua;

Uses Crt;
Var
    Anio : integer;
    A,B,C,D,E : integer;
    Día : integer;

Begin

ClrScr;
Write ('¿Cual es el año?');
Readln (Anio);

A: = Anio mod 19;
B: = Anio mod 4;
C: = Anio mod 7;
D: = (19*A+24) mod 30;
E: = (2*B+4*C+6*D+5) mod 7;

Día := 22+ D+ E;

Write ('La fecha del domingo de pascua de', Anio, 'es :');

IF día <= 31
    THEN
        WriteLn (Día, ' de marzo')
    ELSE
        WriteLn (Día - 31,' de abril ')

End.

```

ACTIVIDAD DE REGULACIÓN

Para que confirmes e integres los conocimientos que has alcanzado hasta el momento. Realiza la siguiente actividad:

1. Realiza el programa en Pascal del siguiente problema:
 - a) Convierte los metros en centímetros utilizando los bucles While y Repeat.

ACTIVIDADES DE CONSOLIDACIÓN

Con la finalidad de que apliques e integres los conocimientos adquiridos con el estudio de este material. A continuación realiza lo que se te pide:

1. Coloca en la línea de la izquierda el(los) concepto(s) que correspondan a los enunciados de la derecha:

- | | |
|----------|--|
| a) _____ | Expresa el algoritmo como un programa en un lenguaje de programación. |
| b) _____ | Esta etapa es medular y se establece como una secuencia ordenada de pasos sin ambigüedades, que conducen a la solución de un problema. |
| c) _____ | En esta característica del algoritmo, dado un conjunto de datos de entrada, deberán arrojar los mismos resultados. |
| d) _____ | Son las características de un algoritmo. |
| e) _____ | En esta etapa se ejecuta y se valida el programa por la computadora. |
| f) _____ | Serie de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema. |

2. Relaciona los pasos para la construcción del algoritmo que se encuentran en la izquierda con sus definiciones de la derecha.

- | | | |
|--------|---|---------------------|
| a) () | Operación u operaciones secuenciales, cuyo objetivo es obtener la solución al problema | 1. Datos de entrada |
| b) () | Acción o operación que permite el ingreso de los datos del problema. | 2. Resultados |
| c) () | Acción que permite el diseño de un algoritmo. | 3. Procesamiento |
| d) () | Operación o conjunto de operaciones que permiten comunicar al exterior los resultados alcanzados. | |

3. Construye el algoritmo del siguiente problema, haciendo uso de los tres módulos para su construcción. Dado los datos enteros A y B, escribe el resultado de la siguiente expresión:

$$C = \frac{(A + B)}{3}$$

a) Escribe el algoritmo correspondiente, marcando cada uno de sus pasos.

--

- b) Aplica los tres módulos para su construcción:

• Entrada de Datos	
• Procesamiento de Datos	
• Salida de Datos	

4. Relaciona los conceptos de la izquierda con sus definiciones de la derecha, colocando el número en el paréntesis que corresponda.

1. Sistemática	a) Creador de la programación estructurada.	()
2. Procedimientos	b) Estos se crean para una tarea específica regresando Resultados.	()
3. Recursos Humanos	c) Con qué otro nombre se les conoce a Submódulos, funciones o....	()
4. Submódulos	d) Llamada bucle y repite la ejecución de una secuencia de acciones.	()
5. Selectiva	e) Utiliza estructuras llamadas objetos que unen procedimientos y funciones, llamados encapsulados que forman la unidad.	()
6. Orientada a objetos	f) De un número determinado de alternativas objetos acciones se selecciona una o varias condiciones lógicas.	()
7. Repetitiva	g) Descompone una acción compleja en un número de acciones más simples que se ejecutan en la computadora.	()
8. Secuencial	h) El diseño descendente es una técnica.	()

9. Estructurada	i) En esta técnica una instrucción o acción sigue a otra, es decir la salida de una es la entrada de la otra.	()
10. Edsger W. Dijkstra		
11. Algoritmo	j) Con que otro nombre se le conoce a la programación estructurada.	()

5. ¿Cuáles son las técnicas de la etapa de análisis, que nos permiten resolver un problema?

a)
b)
c)

6. ¿Qué es un programa?

7. Menciona los tres principales problemas que se tienen con la programación lineal:

a)
b)
c)

8. ¿A quién se le atribuye ser el creador de la programación estructurada?

--

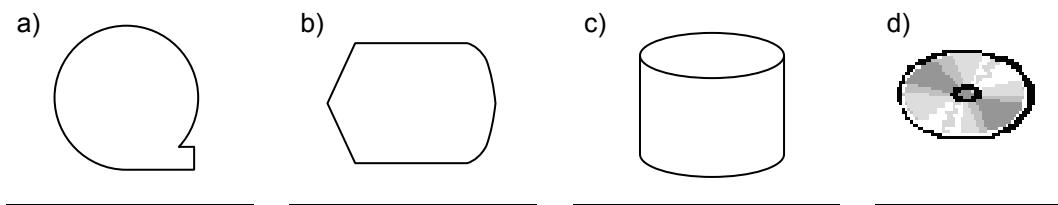
9. Construye el algoritmo del cálculo de la suma de los números del 1 a 999.

--

10. Relaciona ambas columnas, anotando en el paréntesis de la izquierda, la letra que corresponda:

- | | | |
|-----------|---|-------------------------------------|
| 1. . () | Nos permite hacer cálculos y asignar valores. | a) Decisión |
| 2. . () | Permite lectura y escritura de datos. | b) Submódulo |
| 3. . () | Indica la dirección que sigue la información | c) Conector misma página |
| 4. . () | Se evalúa una condición en su interior, y dependiendo del resultado de la selección, se sigue uno de los caminos establecidos. | d) Comentarios |
| 5. . () | Se utiliza para concertar dos partes de un mismo diagrama en una misma hoja. | e) Impresora |
| 6. . () | También llamado estructura algorítmica selectiva múltiple. | f) Conector entre diferente página. |
| 7. . () | Se usan para añadir notas clasificadoras a otros símbolos del diagrama de flujo | g) Flujo de información |
| 8. . () | Se da la salida de los datos. | h) Proceso |
| 9. . () | Conecta dos partes de un mismo diagrama que se encuentran separadas en diferentes páginas. | i) Entrada/salida |
| 10. . () | Es un módulo independiente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y al terminar, regresa al programa principal. | j) Decisión múltiple |
| | | k) Toma de decisión |

11. Identifica la función de las siguientes figuras



12. Relaciona las siguientes columnas, colocando en el paréntesis de la izquierda, la letra que corresponda:

- | | | | |
|--------|----------------|----|---|
| 1. () | Carácter. | a) | Es una secuencia de letras y dígitos, aunque el primer carácter debe ser una letra. |
| 2. () | Datos Enteros. | b) | Podemos asignarle a este tipo de variables un conjunto de caracteres. |
| 3. () | Datos lógicos. | c) | Son aquellos números que no tienen parte decimal. |

- 4. () Constantes. d) Son números que tienen parte decimal, se les puede asignar valores enteros como valores de punto flotante.
- 5. () Identificador. e) Este tipo de identificadores pueden cambiar de valor durante la ejecución del programa.
- 6. () Datos Reales. f) Dato cuyo contenido puede ser una letra del abecedario, un número o un símbolo especial.
- 7. () Cadena de caracteres g) Sólo podrá adquirir dos valores TRUE o FALSE.
- 8. () Variables. h) En este tipo de identificador no cambia su valor durante el programa.
- 9. () Datos Alfanuméricos i) Combinación de constantes, variables, símbolos de operación, paréntesis y nombre de funciones especiales.
- 10. () Tipo de datos simple. j) Compuesto por carácter (simple) y cadena de carácter (compuesto ó estructurado).
- 11. () Expresiones. k) Contiene a los datos numéricos, carácter y lógico.

13. Obtén el resultado de las siguientes expresiones aplicando sus prioridades.

a) $(B^{**2}) > (C*6)$ si $B=5$ y $C=4$	b) $5*6^{**2}/60 \text{ div } 3$	c) $-6 + 7 + (6 \text{ mod } 4)$
---	----------------------------------	----------------------------------

14. Contesta lo que se te pide:

a) ¿Cuándo se utiliza una estructura algorítmica secuencial?

b) ¿Cuándo se utiliza una estructura algorítmica selectiva?

c) ¿Cuándo se utiliza una estructura algorítmica repetitiva?

15. Realiza el diagrama de flujo que se requiere para calcular el aumento que tendrá en su sueldo, un trabajador de la empresa Charito Mexicano, considerando los siguientes criterios:

- Sueldo $< \$1,200$ tendrá un aumento del 24%
- $\$1,200 \leq \text{sueldo} \leq \$1,700$ tendrá un aumento del 20%
- Sueldo $> \$1,700$ tendrá un aumento del 15%

16. ¿Qué es un pseudocódigo?

17. Enuncia cinco ventajas de utilizar un pseudocódigo.

18. Realiza el pseudocódigo para obtener la suma de los números impares hasta 701 inclusive, utilizando la estructura algorítmica repetitiva **“REPETIR”**.

19. Coloca en el paréntesis la letra que corresponda.

1. () ¿Quién fue el creador de lenguaje Pascal?

- a) Pascal.
- b) Norton.
- c) Hennefeld.
- d) Nicklaus Wirth.

2. () ¿Serie de instrucciones de cómputo estructuradas y ordenadas, para hacer una tarea determinada?
- a) Codificar.
 - b) Programa.
 - c) Compilador.
 - d) Lenguaje ensamblador.
3. () Algunos pasos son obtener programa fuente, programa objeto y programa ejecutable.
- a) Montador.
 - b) Ciclo de vida.
 - c) Ciclo moderno.
 - d) Fases de Compilación.
4. () Es un programa que complementa al traductor, se utiliza cuando el programa objeto necesita de una preparación y agregar algunas rutinas del propio lenguaje.
- a) Objeto.
 - b) Traductor.
 - c) Montador.
 - d) Compilador.
5. () Hace la traducción, analiza la sintaxis del programa. Detecta errores de escritura y los corrige si son necesarios.
- a) Objeto.
 - b) Fuente.
 - c) Traductor.
 - d) Compilador

20. Menciona las fases que se siguen para lograr un programa ejecutable.

21. ¿Qué características necesita una computadora para instalar Turbo Pascal?

a)
b)
c)

22. Relaciona ambas columnas

- | | | | |
|--------|---------------|----|---|
| 1. () | Cirscr | a) | Permite mandar a la impresora la salida con la declaración Ist. |
| 2. () | Asignación | b) | Permite enlazar a todos los programas y contiene todos los procedimientos y funciones. |
| 3. () | System | c) | Permite posicionar el cursor en la columna y renglón requerido. |
| 4. () | GotoXY(a.b) | d) | Nos permite separar dos enunciados consecutivos en su programa. |
| 5. () | Printer | e) | Limpia la pantalla de salida y coloca el cursor en la esquina superior derecha de la pantalla |
| 6. () | Writeln | f) | Se evalúa la expresión de la derecha y se le asigna el valor del resultado a la variable que se encuentra a la izquierda del símbolo. |
| 7. () | Punto y coma. | g) | Despliega lo que se encuentra entre paréntesis y se coloca al inicio de la siguiente línea en la pantalla. |
| 8. () | Readln | h) | Nos permite leer datos del teclado, pide un dato y te posiciona en la primera columna del siguiente renglón. |
| | | i) | Se utilizan para dar mensajes en el programa. |

23. Realiza un programa que calcule el promedio de un conjunto de números reales.

AUTOEVALUACIÓN

A continuación te presentamos las posibles respuestas en relación a las Actividades de Consolidación. Si se te presenta alguna duda, acude con tu asesor de Informática.

1.

- a) Construcción.
- b) Análisis.
- c) Determinístico
- d) Características
- e) Verificación
- f) Algoritmo.

2.

- a) (3)
- b) (1)
- c) (2)
- d) (-)

3.

- a)
 - 1. Inicio
 - 2. Leer A,B
 - 3. Calcula la expresión $C=(A+B)**2/3$
 - 4. Escribir C
 - 5. Fin

b)

Entrada de Datos	Leer A,B
Procesamiento de Datos	$C=(A+B)**2/3$
Salida de Datos	Escribir C

4.

- a) (10)
- b) (4)
- c) (2)
- d) (7)
- e) (6)
- f) (5)
- g) (3)
- h) (9)
- i) (8)
- j) (1)

5.

- a) Lineal.
- b) Estructurada.
- c) Orientada a Objetos.

6.

Conjunto de órdenes, instrucciones que se le dan a la computadora para que realice un proceso determinado.

7.

- a) El programador no se atenía a un método específico
- b) Altos costos de desarrollo y mantenimiento.
- c) Atraso en la explotación del sistema.

8.

Edsger W. Dijkstra

9.

1. Inicio.
2. la variable 1 toma el valor de 1
3. la variable suma toma el valor de 0
4. Mientras $1 \leq 999$ hacer lo siguiente
 - 4.1 a suma sumar suma más 1
 - 4.2 a l sumar l sumar 1fin de mientras
5. escribir suma
6. fin.

10.

1. . (h)
2. . (i)
3. (g)
4. (a)
5. (c)
6. (j)
7. (d)
8. (e)
9. (f)
10. (b)

11.

- a) Almacenamiento de acceso secuencial
- b) Pantalla
- c) Disco Magnético
- d) Unidad de CD

12.

1. . (f)
2. . (c)
3. (g)
4. (h)
5. (a)
6. (d)
7. (b)
8. (e)
9. (j)
10. (k)
11. (i)

13.

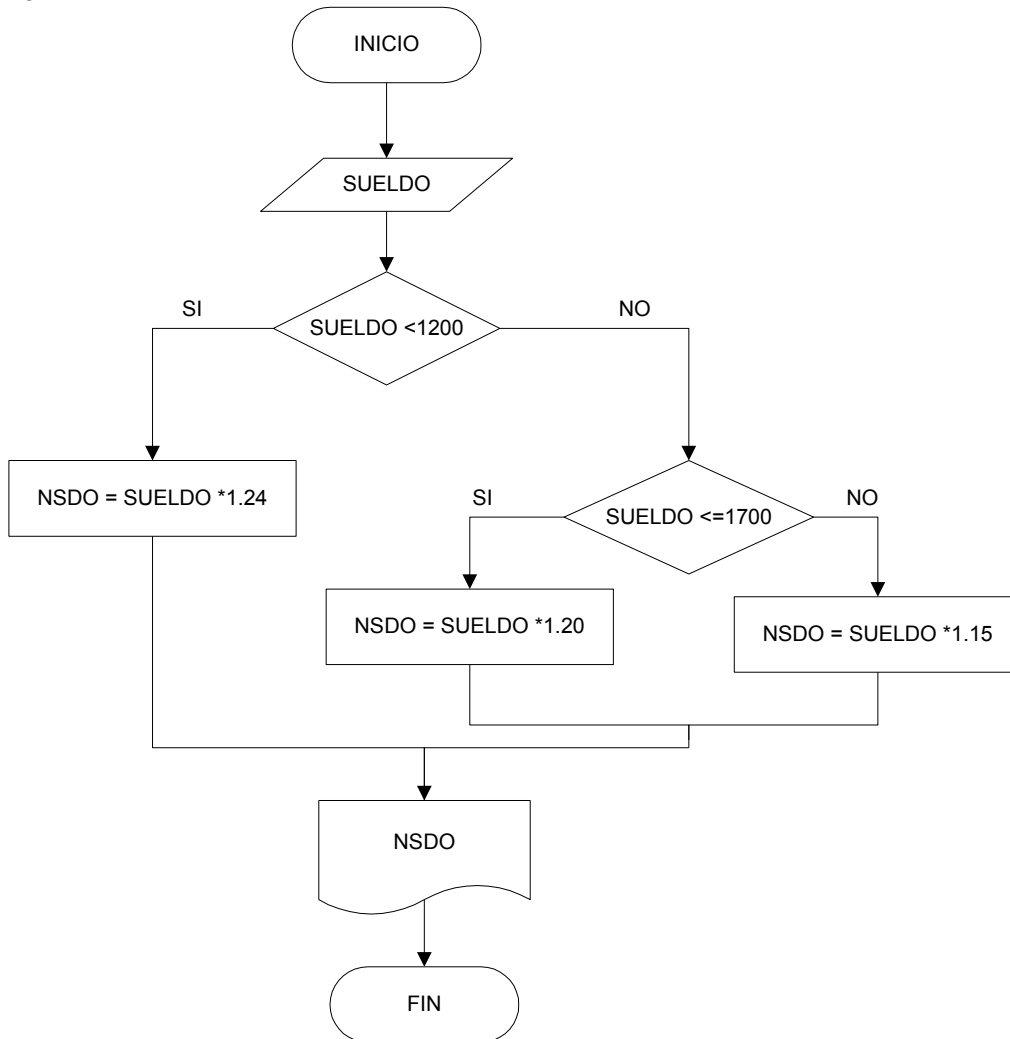
a) $(B^{**2}) > (C*6)$ si $B=5$ y $C =4$ $(5^{**2}) > (4*6)$ $(25) > (24)$ Verdadero	b) $5*6^{**2}/60 \text{ div } 3$ $5*36/60 \text{ div } 3$ $180/60 \text{ div } 3$ $3 \text{ div } 3$ 1	c) $-6 + 7(6 \text{ mod } 4)$ $-6 + 7 + 12$ $-6 + 9$ 3
--	--	---

14.

- a) Se utiliza cuando queremos mostrar que de una acción le sigue otra y así sucesivamente, hasta el final del proceso.
- b) Se utiliza cuando se quiere tomar una decisión lógica por ello también se le llama estructura de decisión o alternativa.

- c) Cuando permite repetir una o varias acciones un número determinado de veces. A las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan **bucles** y se llama **iteración** al hecho de repetir la ejecución de una secuencia de acciones.

15.



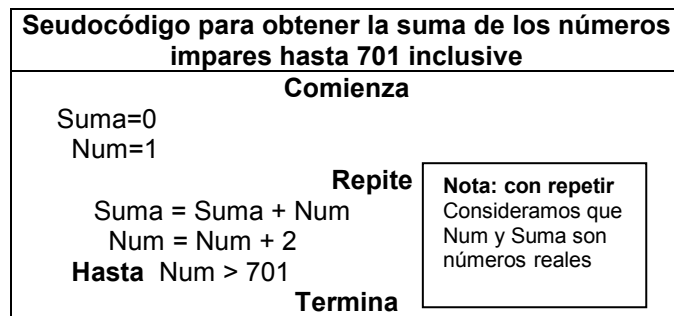
16. ¿Qué es un pseudocódigo?

El pseudocódigo es un lenguaje similar al español y es un medio de representar las estructuras algorítmicas o de control, es un primer intento para llegar a un lenguaje de programación. Es un lenguaje de especificación, es decir la descripción de algoritmos y los diagramas de flujo, permitiendo pasar a la traducción de un lenguaje de programación. Traduce todos los símbolos y funciones representados en el diagrama de flujo, para ser traducidos finalmente en un lenguaje de alto nivel.

17. Enuncia cinco ventajas de utilizar un pseudocódigo.

- Es compacto.
- Puede modificarse fácilmente.
- Se utilizan palabras claves en mayúsculas en español, que identifican las estructuras algorítmicas secuenciales, selectivas y repetitivas.
- Se puede construir de manera estructurada (de arriba hacia abajo).
- No existen reglas estándar para utilizarlo.
- No se tiene una representación gráfica de la lógica del programa.

18. Realiza el pseudocódigo para obtener la suma de los números pares hasta 700 inclusive, utilizando la estructura algorítmica repetitiva "REPETIR"



19.

- 1 (d)
- 2 (b)
- 3 (d)
- 4 (c)
- 5 (d)

20.

Programa fuente.
Traductor (Compilación → Programa objeto).
Montador o linkeador.
Programa ejecutable.

21.

- a) Colocar en la unidad y teclear el programa ejecutable install o instalar presione Enter.
- b) Por lo menos 3.2 Mb de espacio libre para instalar el programa completo.
- c) Para instalación básica, podemos no instalar los ejemplos existe.

22.

Relaciona las columnas

1	e
2	f
3	b
4	c
5	a
6	g
7	d
8	h

23.

```
Program Promedio_de_numeros;
```

```
Uses Crt;
```

```
Var
```

```
    Valor,Suma,Promedio,real;
```

```
    Cuenta:integer;
```

```
    Fin:Char;
```

```
Begin
```

```
    Clrscr
```

```
    Suma:=0;
```

```
    Cuenta:=0;
```

```
    Writeln('Teclea varios números y termina la entrada con 0');
```

```
    Readln(Valor);
```

```
    While valor<> 0 do
```

```
    Begin
```

```
        Suma:=Suma+Valor;
```

```
        cuenta:=Cuenta+1;
```

```
        Readln(Valor)
```

```
    End;
```

```
    If Cuenta>0 then
```

```
    Begin
```

```
        Promedio:=Suma/Cuenta;
```

```
        Writeln(Cuenta,' valores leídos, Promedio =', Promedio:5:2)
```

```
    End
```

```
    Else writeln ('ningún valor leído');
```

```
    Readln;
```

```
End.
```

GLOSARIO

- Algoritmo.** Es una serie de pasos, procedimientos o acciones que llevan una secuencia lógica y sistemática que permiten alcanzar un resultado o resolver un problema.
- Análisis.** Conjunto de trabajos que comprenden el estudio detallado de un problema, la concepción de un método que permita resolverlo y la definición precisa del tratamiento correspondiente en el ordenador.
- Constantes.** es un valor que no puede cambiar durante la ejecución de un programa.
- Diagrama de Flujo (Flow Chart).** Es una representación gráfica de un algoritmo. Los símbolos utilizados han sido normalizados por el Instituto Norteamericano (ANSÍ).
- Expresión.** Es un conjunto de datos o funciones unidas por operadores aritméticos.
- Seudocódigo.** Es una herramienta de programación en las que las instrucciones se escriben en palabras similares al inglés o español, que facilitan la escritura y lectura de programas.
- Variables.** Son objetos de un programa cuyo valor puede variar por operadores aritméticos

BIBLIOGRAFÍA CONSULTADA

- CAIRO** Oswaido: Metodología de la programación Tomo 1, Alfaomega, México, 1995.
- JOYANES** Aguilar L.: Fundamentos de Programación Segunda Edición, Me Graw – Hill, México, 1996.
- PALMER** Scott D.: Domine Turbo Pascal Ventura ediciones, México, 1992.
- HENNETEID** Jolien: Turbo Pasca! con Aplicaciones 4.0 - 6.0 Segunda Edición, Iberoamericana, México, 1992.
- SCHEID** Francis: Introducción a las Ciencias de las Computadoras Segunda Edición, Schaum, Mc Graw – Hill. s.l. s.f.

DIRECTORIO

Dr. Roberto Castañón Romo
Director General

Mtro. Luis Miguel Samperio Sánchez
Secretario Académico

Lic. Filiberto Aguayo Chuc
Coordinador Sectorial Norte

Lic. Rafael Torres Jiménez
Coordinador Sectorial Centro

Biol. Elideé Echeverría Valencia
Coordinadora Sectorial Sur

Dr. Héctor Robledo Galván
**Coordinador de Administración Escolar
y del Sistema Abierto**

Lic. José Noel Pablo Tenorio
Director de Asuntos Jurídicos

Mtro. Jorge González Isassi
Director de Servicios Académicos

C.P. Juan Antonio Rosas Mejía
Director de Programación

Lic. Miguel Ángel Báez López
Director de Planeación Académica

M.A. Roberto Paz Neri
Director Administrativo

Lic. Manuel Tello Acosta
Director de Recursos Financieros

Lic. Pablo Salcedo Castro
Unidad de Producción Editorial