

# **INTRODUCCION**

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

Estos apuntes servirán de apoyo, 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 practicas 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

## 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.

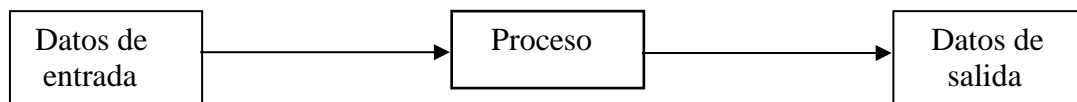
El diseño de soluciones a la medida de nuestros problemas, requiere como en otras disciplinas una metodología que nos enseñe de manera gradual, la forma de llegar a estas soluciones.

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.

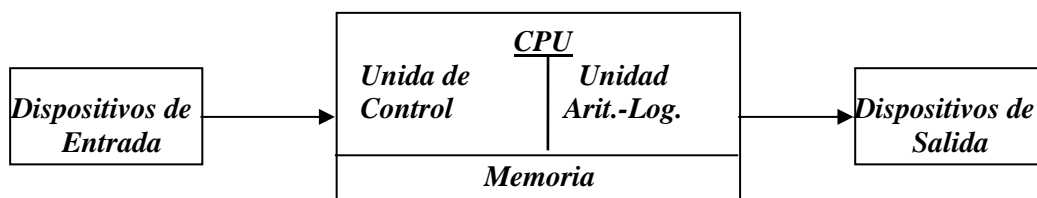
**Computadora:** Es un dispositivo electrónico utilizado para procesar información y obtener resultados. Los datos y la información se pueden introducir en la computadora como entrada (input) y a continuación se procesan para producir una salida (output).

### *Proceso de información en la computadora*



**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*



**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, scanners (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:

- Unidad de control
- 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:

- Memoria Central o Interna
- 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 (Random 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 mas 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:

- \* Los mensajes deben correr en un sentido a la vez.
- \* 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:

- **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. Las instrucciones en lenguaje maquina se expresan en términos de la unidad de memoria más pequeña el bit (dígito binario 0 o 1).
- **Lenguaje de Bajo Nivel (Ensamblador):** En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos para las operaciones y direcciones simbólicas.
- **Lenguaje de Alto Nivel:** Los lenguajes de programación de alto nivel (BASIC, pascal, cobol, fortran, 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.

### ***1.3 Definición de Algoritmo***

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhwarizmi, 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***

- **Cualitativos:** Son aquellos en los que se describen los pasos utilizando palabras.

➤ **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***

➤ **Gráficos:** Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo).

➤ **No Gráficos:** Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocódigo).

## ***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.

### ***1.8 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.

### ***1.9 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 pseudocódigo), 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 en 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

➤ **Documentación Interna**: Son los comentarios o mensaje que se añaden al código fuente para hacer más claro el entendimiento de un proceso.

➤ **Documentación Externa**: Se define en un documento escrito los siguientes puntos:

Descripción del Problema  
Nombre del Autor  
Algoritmo (diagrama de flujo o pseudocódigo)  
Diccionario de Datos

## Código Fuente (programa)

- Manual del Usuario: Describe paso a paso la manera como 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.



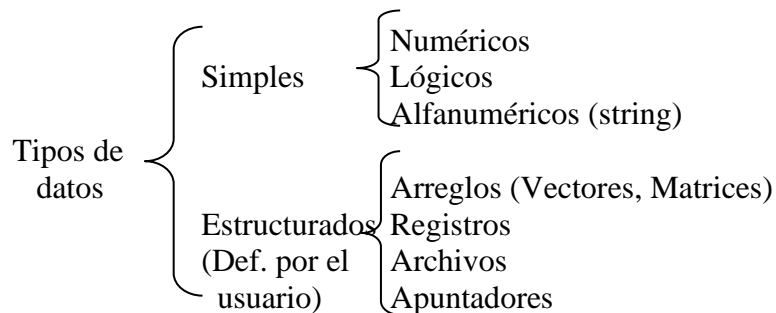
## CAPITULO II.

# 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

### 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.



#### *Tipos de Datos Simples*

- **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.
- **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).
- **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:  
"2005"

### 2.2 Expresiones

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

$$a+(b + 3)/c$$

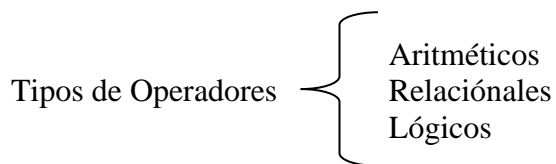
Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

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

- Aritméticas
- Relacionales
- Lógicas

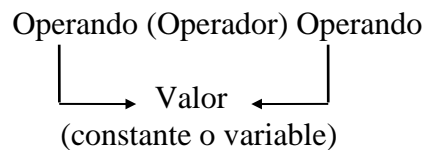
## 2.3 Operadores y Operandos

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



➤ **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.



### **Operadores Aritméticos**

+	Suma
-	Resta
*	Multiplicación
/	División
Mod	Modulo (residuo de la división entera)

Ejemplos:

Expresión	Resultado
-----------	-----------

$$\begin{array}{ll} 7 / 2 & 3.5 \\ 12 \bmod 7 & 5 \\ 4 + 2 * 5 & 14 \end{array}$$

### ***Prioridad de los Operadores Aritméticos***

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

- 1.- ^ Exponenciación
- 2.- \*, /, mod Multiplicación, división, modulo.
- 3.- +, - Suma y resta.

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

Ejemplos:

$$\begin{array}{ll} 4 + 2 * 5 = 14 & \\ 23 * 2 / 5 = 9.2 & 46 / 5 = 9.2 \\ 3 + 5 * (10 - (2 + 4)) = 23 & 3 + 5 * (10 - 6) = 3 + 5 * 4 = 3 + 20 = 23 \\ 3.5 + 5.09 - 14.0 / 40 = 5.09 & 3.5 + 5.09 - 3.5 = 8.59 - 3.5 = 5.09 \\ 2.1 * (1.5 + 3.0 * 4.1) = 28.98 & 2.1 * (1.5 + 12.3) = 2.1 * 13.8 = \\ 28.98 & \end{array}$$

### ***➤ Operadores Relacionales:***

- Se utilizan para establecer una relación entre dos valores.
- Compara estos valores entre si y esta comparación produce un resultado de certeza o falsedad (verdadero o falso).
- Los operadores relacionales comparan valores del mismo tipo (numéricos o cadenas)
- Tienen el mismo nivel de prioridad en su evaluación.
- Los operadores relacionales tiene menor prioridad que los aritméticos.

### ***Operadores Relacionales***

- |    |                   |
|----|-------------------|
| >  | Mayor que         |
| <  | Menor que         |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| <> | Diferente         |
| =  | Igual             |

Ejemplos:

$$\text{Si } a = 10 \quad b = 20 \quad c = 30$$

$a + b > c$	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)

### ➤ **Operadores Lógicos:**

- η Estos operadores se utilizan para establecer relaciones entre valores lógicos.
- η Estos valores pueden ser resultado de una expresión relacional.

#### **Operadores Lógicos**

And	Y
Or	O
Not	Negación

#### **Operador And**

<i>Operando1</i>	<i>Operador</i>	<i>Operando2</i>	<i>Resultado</i>
T	AND	T	T
T		F	F
F		T	F
F		F	F

#### **Operador Or**

<i>Operando1</i>	<i>Operador</i>	<i>Operando2</i>	<i>Resultado</i>
T	OR	T	T
T		F	T
F		T	T
F		F	F

#### **Operador Not**

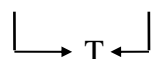
<i>Operando</i>	<i>Resultado</i>
T	F
F	T

Ejemplos:

$$(a < b) \text{ and } (b < c)$$

$$(10 < 20) \text{ and } (20 < 30)$$

$$T \quad \text{and} \quad T$$



### ***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 acceder a su contenido.

Ejemplo:      Nombre  
                 Num\_hrs  
                 Calif2

### ***Reglas para formar un Identificador***

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

### **Constantes y Variables**

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

Ejemplo:

$\pi = 3.1416$

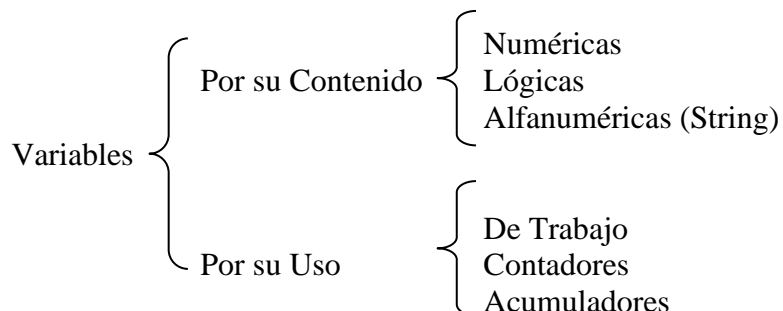
- **Variable:** Es un espacio en la memoria de la computadora que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambiar durante la ejecución del programa. Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con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

Ejemplo:

$\text{área} = \pi * \text{radio}^2$

Las variables son: el radio, el área y la constante es  $\pi$

### **Clasificación de las Variables**



### **Por su Contenido**

- **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:

$\text{iva}=0.15$        $\pi=3.1416$        $\text{costo}=2500$

- **Variables Lógicas:** Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.

- **Variables Alfanuméricas:** Está formada por caracteres alfanuméricos (letras, números y caracteres especiales). Ejemplo:

letra='a'      apellido='lopez'      direccion='Av. Libertad #190'

**Por su Uso**

➤ **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

➤ **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.

➤ **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 PARA LA FORMULACIÓN DE ALGORITMOS

- 3.1 Diagrama de flujo
- 3.2 Pseudocódigo
- 3.3 Diagrama estructurado (nassi-schneiderman)

### OBJETIVO EDUCACIONAL:

El aprendiz:

- 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:


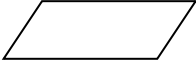

Diagrama de Flujo  
Pseudocódigo

### *3.1 Diagrama de Flujo*

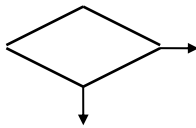
Un diagrama de flujo es la representación gráfica de un algoritmo. También se puede decir que es la representación detallada en forma gráfica de cómo deben realizarse los pasos en la computadora para producir resultados.

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

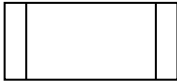
Los símbolos utilizados han sido normalizados por el instituto norteamericano de normalización (ANSI).

<u><b>SÍMBOLO</b></u>	<u><b>DESCRIPCIÓN</b></u>
	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.





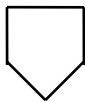
Símbolo de decisión indica la realización de una comparación de valores.



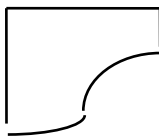
Se utiliza para representar los subprogramas.



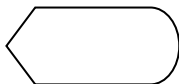
Conector dentro de página. Representa la continuidad del diagrama dentro de la misma página.



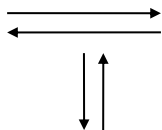
Conector fuera de página. Representa la continuidad del diagrama en otra página.



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***

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

### 3.2 Pseudocódigo

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 pseudocódigo 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 pseudocódigo utiliza palabras que indican el proceso a realizar.

#### *Ventajas de utilizar un Pseudocódigo a un Diagrama de Flujo*

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

### 3.3 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:

