

## INDICE DE CONTENIDO

CAPITULO IV. TÉCNICAS PARA LA FORMULACIÓN DE ALGORITMOS.....	19
4.1 Diagrama de flujo.....	20
4.2 Pseudocodigo.....	21
4.3 Diagrama estructurado (nassi-schneiderman).....	22
CAPITULO V. ESTRUCTURAS ALGORITMICAS.....	23
5.1 Secuenciales.....	24
- Asignación.....	24
- Entrada.....	24
- Salida.....	24
5.2 Condicionales.....	25
- Simples.....	25
- Múltiples.....	25
5.3 Repetición fila condicional.....	39

## CAPITULO IV. TÉCNICAS PARA LA FORMULACIÓN DE ALGORITMOS

- 4.1 Diagrama de flujo
- 4.2 Pseudocodigo
- 4.3 Diagrama estructurado (nassi-schneiderman)

### OBJETIVO EDUCACIONAL:

El alumno:

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

Las dos herramientas utilizadas comúnmente para diseñar algoritmos son:

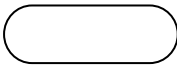
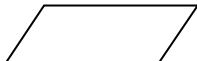
Diagrama de Flujo  
Pseudocodigo

### ***4.1 Diagrama de Flujo***

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

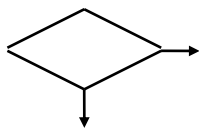
Esta representación gráfica se da cuando varios símbolos (que indican diferentes procesos en la computadora), se relacionan entre si mediante líneas que indican el orden en que se deben ejecutar los procesos.

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

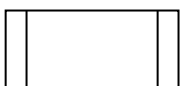
<b><u>SÍMBOLO</u></b>	<b><u>DESCRIPCIÓN</u></b>
	Indica el inicio y el final de nuestro diagrama de flujo.
	Indica la entrada y salida de datos.



Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.



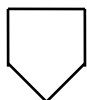
Símbolo de decisión indica la realización de una comparación de valores.



Se utiliza para representar los subprogramas.



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



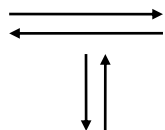
Conector fuera de pagina. Representa la continuidad del diagrama en otra pagina.



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.

## **4.2 Pseudocodigo**

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

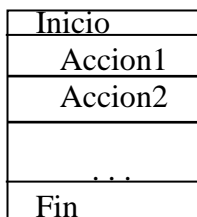
Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocodigo utiliza palabras que indican el proceso a realizar.

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

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

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

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



## CAPITULO V. ESTRUCTURAS ALGORITMICAS

### 5.1 Secuenciales

- Asignación
- Entrada
- Salida

### 5.2 Condicionales

- Simples
- Múltiples

### 5.3 Repetición fila condicional

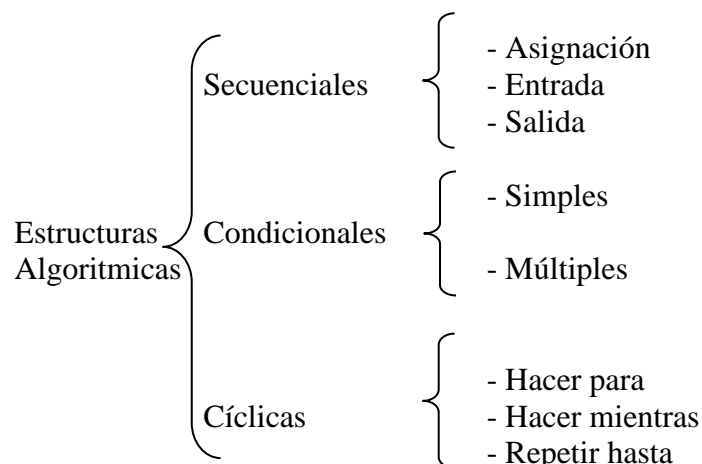
## OBJETIVO EDUCACIONAL:

El alumno:

- Conocerá las diferentes estructuras algorítmicas como componentes básicos de los programas y aplicara la combinación de ellas para el desarrollo de algoritmos mas complejos.

## ESTRUCTURAS ALGORITMICAS

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



### 5.1. Estructuras Secuenciales

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

```
Inicio
  Accion1
  Accion2
  .
  .
  AccionN
Fin
```

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

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

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

```
Leer a, b
```

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

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

```
Escribe “El resultado es:”, R
```

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

## 5.2 Estructuras de Condicionales

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

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

Si <condición> entonces  
    Acción(es)  
Fin-si

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

Si <condición> entonces  
    Acción(es)  
si no  
    Acción(es)  
Fin-si

Donde:

Si .....	Indica el comando de comparación
Condición.....	Indica la condición a evaluar
entonces.....	Precede a las acciones a realizar cuando se cumple la condición
acción(es).....	Son las acciones a realizar cuando se cumple o no la condición
si no.....	Precede a las acciones a realizar cuando no se cumple la condición

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

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

Si <condición> entonces  
    Acción(es)

si no  
Si <condición> entonces  
Acción(es)  
si no  
.  
.  
.  
} Varias condiciones

- *Forma General*

### Casos Variable

Op1: Acción(es)

Op2: Acción(es)

•

•

OpN: acción

## Fin-casos

### 5.3. Estructuras Cíclicas

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

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

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

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

Accion1

## Accion2

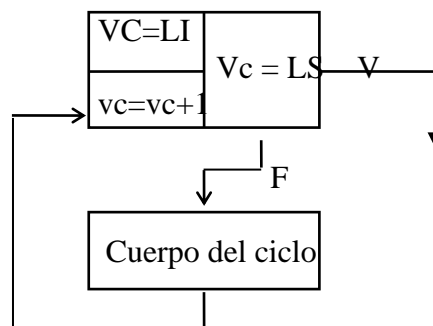
•

•

•

AccionN

Fin-para



Donde:

V.C Variable de control del ciclo



L.I     Limite inferior  
L.S     Limite superior

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

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

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

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

Hacer mientras <condición>  
    Accion1  
    Accion2  
    .  
    .  
    AccionN  
Fin-mientras

