

UNIVERSIDAD PRIVADA DEL NORTE**FACULTAD DE INGENIERÍA**

CARRERA: Ingeniería de sistemas computacionales

CURSO: Introducción a la ingeniería de sistemas computacionales

DOCENTE: Ing. José Castillo Zumarán

TEMA: Introducción a los algoritmos, representación de algoritmos (DFD, pseudocódigo y código), operadores.

Índice de contenidos

Capítulo 1.	Introducción a los algoritmos.....	(pag. 3)
Capítulo 2.	Representación de algoritmos.....	(pag. 3-5)
Capítulo 3.	Operadores.....	(pag. 6-8)
Capítulo 4.	Sílabo	(pag. 9)

Representación de algoritmos

La representación de algoritmos permite planificar y comunicar la lógica de una solución antes de implementarla en código. Las representaciones comunes incluyen los Diagramas de Flujo de Datos (DFD), el pseudocódigo y el código en un lenguaje de programación. Cada uno tiene un propósito específico y se usa en distintas etapas del desarrollo para facilitar la comprensión y la implementación de la solución.

Diagrama de Flujo de Datos (DFD)

1. Definición

El Diagrama de Flujo de Datos (DFD) es una representación gráfica de cómo la información fluye a través de un sistema. Describe las entradas, procesos, salidas y almacenamiento de datos en un sistema. Los DFD se organizan en niveles (0, 1, 2...) para mostrar un mayor o menor detalle.

2. Componentes

- > **Entidades Externas**: Representan las fuentes de entrada y salida de datos del sistema.
- > **Procesos**: Transforman los datos de entrada en salida.
- > **Almacenes de Datos**: Lugares donde se almacenan los datos.
- > **Flujos de Datos**: Líneas que muestran el movimiento de datos entre entidades, procesos y almacenes.

3. Ejemplo de DFD

Un DFD de nivel 0 de un sistema de gestión de pedidos podría incluir:

- > Entidad Externa: Cliente
- > Proceso: Procesar Pedido
- > Almacén de Datos: Base de Datos de Inventario
- > Flujo de Datos: Información del Pedido, Confirmación

Representación de los elementos dentro de un diagrama de flujo:

1. Inicio y Fin (Óvalo)

Representado por un óvalo, marca el comienzo y el final del diagrama de flujo.

Indica el punto donde el proceso inicia y, cuando se usa al final, muestra dónde termina.

Generalmente contiene palabras como "Inicio" o "Fin".

2. Proceso (Rectángulo)

Representado por un rectángulo, este elemento indica una acción o instrucción dentro del proceso, como cálculos, asignaciones o transformaciones de datos.

Cada rectángulo describe un paso único del proceso.

3. Decisión (Rombo)

Representado por un rombo, define un punto de decisión en el flujo del proceso, donde se debe escoger una dirección basada en una condición.

Tiene al menos dos salidas (por ejemplo, "Sí" o "No") que muestran los caminos posibles.

Las decisiones típicas incluyen comparaciones o verificaciones, como "¿El valor es mayor que 10?".

4. Entrada y Salida (Paralelogramo)

Representado por un paralelogramo, indica operaciones de entrada o salida.

Ejemplos de entrada pueden ser la lectura de datos por parte del usuario, y ejemplos de salida incluyen la presentación de resultados en pantalla.

5. Conector (Círculo o punto pequeño)

Representado por un pequeño círculo, conecta partes del diagrama cuando el flujo de operaciones continúa en otra sección de la página o hacia una parte lejana.

Se utiliza para mantener el diagrama ordenado y fácil de leer cuando es demasiado largo o complicado.

6. Flechas (Líneas de Flujo)

Indican la dirección en la que el proceso fluye de un paso a otro.

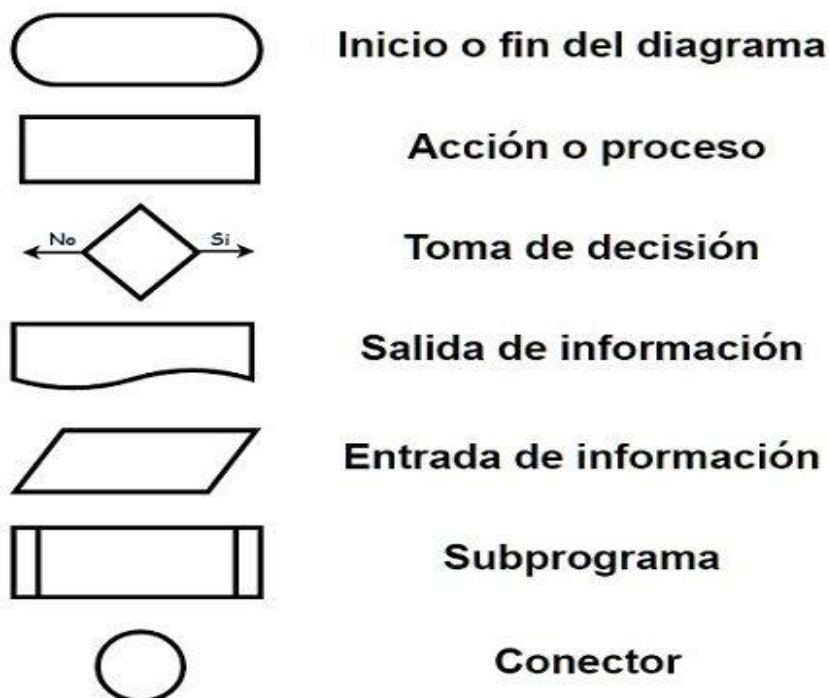
Conectan los elementos en el orden en que deben ser ejecutados.

Son fundamentales para guiar al lector y entender la secuencia de los pasos.

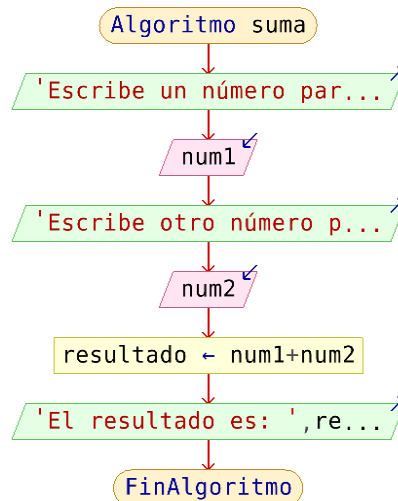
7. Subproceso o Función (Rectángulo con líneas dobles en los extremos)

Representa un proceso que se detalla en otro diagrama o un conjunto de pasos que pueden definirse independientemente.

Se utiliza cuando un paso específico es complejo y debe explicarse en un diagrama de flujo separado, manteniendo el diagrama principal simple y claro



Ejemplo de un diagrama de flujo:



Descripción del Diagrama de Flujo - Algoritmo "suma"

1. **Inicio** (Algoritmo suma)

El algoritmo inicia. Esto se representa en la parte superior con una elipse que dice "Algoritmo suma".

2. **Solicitar el primer número**

Se muestra un mensaje al usuario:

'Escribe un número par...'

Aquí el programa está pidiendo al usuario que introduzca un número par. El valor que el usuario ingresa se almacena en la variable num1.

3. **Solicitar el segundo número**

Luego, el programa muestra otro mensaje:

'Escribe otro número p...'

El usuario debe escribir otro número par, que se guarda en la variable num2.

4. **Calcular la suma**

El algoritmo realiza la suma:

'resultado ← num1 + num2'

Es decir, toma los dos números introducidos por el usuario (num1 y num2) y los suma, guardando el resultado en la variable resultado.

5. **Mostrar el resultado**

Finalmente, el algoritmo muestra el mensaje:

'El resultado es: ', resultado

Este mensaje incluye el valor calculado en el paso anterior, es decir, la suma de los dos números

6. **Fin del Algoritmo**

El proceso termina, lo cual está representado por una elipse con el texto "FinAlgoritmo".

Pseudocódigo

1. Definición

El pseudocódigo es una descripción detallada de un algoritmo usando un lenguaje intermedio entre el lenguaje humano y el lenguaje de programación. Se escribe en una sintaxis que no pertenece a ningún lenguaje en específico, pero permite comprender la lógica del algoritmo de manera clara y sencilla.

2. Ventajas

- > Comprensibilidad: Facilita la comprensión de la lógica sin necesidad de entender un lenguaje específico.
- > Flexibilidad: Se puede ajustar fácilmente.
- > Independencia de Plataforma: Útil en la fase de diseño, sin depender del lenguaje de programación.

3. Ejemplo de Pseudocódigo

Para un algoritmo de cálculo de área de un rectángulo:

```
1  Algoritmo AreaRectangulo
2      Definir Largo Como Entero
3      Definir Ancho Como Entero
4      Definir Area Como Entero
5      Escribir "Ingrese el largo del rectangulo"
6      Leer Largo
7      Escribir "Ingrese el ancho del rectangulo"
8      Leer Ancho
9      Area ← Largo * Ancho
10     Escribir "La area del rectangulo es:", Area
11 FinAlgoritmo
12
```

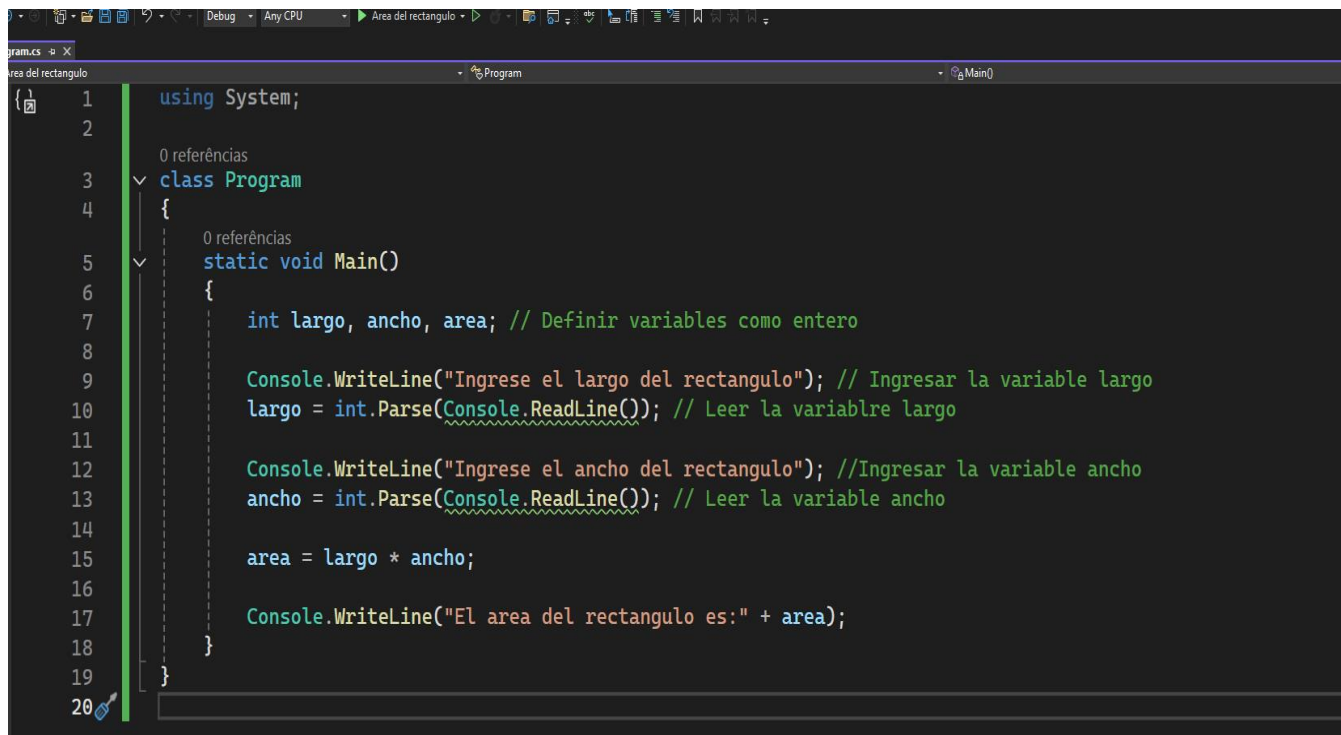
Código

1. Definición

El código es la traducción del pseudocódigo a un lenguaje de programación específico como C++, Python, Java, entre otros. Es la representación más detallada y estricta del algoritmo, ya que debe seguir la sintaxis y semántica del lenguaje de programación en cuestión.

2. Ejemplo de Código en C#

Siguiendo el pseudocódigo anterior para calcular el área de un rectángulo, el código en C# sería:



```
1 using System;
2
3 0 referencias
4 class Program
5 {
6     0 referencias
7     static void Main()
8     {
9         int largo, ancho, area; // Definir variables como entero
10
11         Console.WriteLine("Ingrese el largo del rectangulo"); // Ingresar la variable largo
12         largo = int.Parse(Console.ReadLine()); // Leer la variable largo
13
14         Console.WriteLine("Ingrese el ancho del rectangulo"); //Ingresar la variable ancho
15         ancho = int.Parse(Console.ReadLine()); // Leer la variable ancho
16
17         area = largo * ancho;
18
19         Console.WriteLine("El area del rectangulo es:" + area);
20     }
21 }
```

Operadores

1. Operadores Lógicos

Los operadores lógicos son fundamentales en programación, especialmente en estructuras de control, como las condiciones (if, while) y las evaluaciones booleanas. Se usan para realizar comparaciones entre expresiones y devolver un valor booleano (true o false), lo cual permite la toma de decisiones. Los operadores lógicos principales son:

- **AND (& o and):** Devuelve true solo si ambas expresiones son verdaderas.
 - Ejemplo: (a > 5 && b < 10) devuelve true si a es mayor a 5 y b es menor a 10.
- **OR (|| o or):** Devuelve true si al menos una de las expresiones es verdadera.
 - Ejemplo: (a > 5 || b < 10) devuelve true si a es mayor a 5 o b es menor a 10.
- **NOT (! o not):** Invierte el valor de la expresión. Si es true, devuelve false, y viceversa.
 - Ejemplo: !(a > 5) devuelve true solo si a no es mayor a 5.

Estos operadores son utilizados en expresiones complejas para evaluar múltiples condiciones a la vez, como en `if` anidados o en bucles que dependen de varias condiciones.

2. Operadores Matemáticos

Los operadores matemáticos son aquellos que se usan en álgebra y matemáticas más avanzadas. A diferencia de los operadores aritméticos, se aplican principalmente en algoritmos y cálculos complejos. Algunos operadores matemáticos comunes son:

- **Exponenciación (^ o **):** Eleva un número a una potencia.
 - Ejemplo: 2 ^ 3 o 2 ** 3 devuelve 8.
- **Raíz cuadrada (sqrt):** Calcula la raíz cuadrada de un número. En muchos lenguajes, se utiliza una función como sqrt().
 - Ejemplo: sqrt(9) devuelve 3.
- **Logaritmo (log):** Calcula el logaritmo de un número, siendo base 10 o base e (logaritmo natural).
 - Ejemplo: log(100) devuelve 2 en base 10.
- **Módulo (MOD) (%):** Calcula el residuo de una división.
 - Ejemplo: 7 % 3 devuelve 1 porque el residuo de 7 / 3 es 1.
- **Div (DIV) (/):** Calcula el resultado de una división.
 - Ejemplo: 5 DIV 2 = 2 porque el resultado de 5/2 es 2

- **Suma (+):** Agrega dos números o concatena cadenas de texto.
 - Ejemplo: $5 + 3$ devuelve 8.
- **Resta (-):** Resta un número de otro.
 - Ejemplo: $5 - 3$ devuelve 2.
- **Multipliación (*):** Multiplica dos números.
 - Ejemplo: $5 * 3$ devuelve 15.
- **Incremento (++) y Decremento (--):** Aumentan o disminuyen el valor de una variable en 1. En algunos lenguajes, estos operadores pueden ser **pre-incremento (++x)** o **post-incremento (x++)**. Mas utilizados para contadores en programas.
 - Ejemplo: $x++$ incrementa x en 1 después de usar su valor actual.

3. Operadores de Comparación

Los operadores de comparación son herramientas fundamentales en la programación, ya que permiten **comparar dos valores y tomar decisiones en función del resultado de esa comparación**. Estos operadores siempre **devuelven un resultado booleano: es decir, verdadero (true) o falso (false)**.

Los operadores de comparación más comunes son:

- Igual a (==): Compara si dos valores son iguales.
- Distinto de (!=): Comprueba si dos valores son diferentes.
- Mayor que (>): Verifica si un valor es mayor que otro.
- Menor que (<): Verifica si un valor es menor que otro.
- Mayor o igual que (>=): Comprueba si un valor es mayor o igual que otro.
- Menor o igual que (<=): Comprueba si un valor es menor o igual que otro.

Por ejemplo, si tienes dos números $a = 5$ y $b = 3$, las comparaciones darían los siguientes resultados:

- $a == b$ es falso (porque 5 no es igual a 3).
- $a != b$ es verdadero (porque 5 es diferente de 3).
- $a > b$ es verdadero (porque 5 es mayor que 3).
- $a < b$ es falso (porque 5 no es menor que 3).
- $a >= b$ es verdadero (porque 5 es mayor o igual que 3).
- $a <= b$ es falso (porque 5 no es menor o igual que 3).

 Resumen de los 3 tipos de operadores en tablas:

Operadores Aritmético

Símbolo	Nombre	Ejemplo	Resultado
**	Potencia	3 ** 3	9
*	Multiplicación	3 * 3	9
/	División	3 / 2	1.5
MOD	Módulo (resto)	5 MOD 2	1
DIV	División entera	5 DIV 2	2
+	Suma	5 + 2	7
-	Resta	5 - 2	3

Operadores Lógicos

Operador	Nombre	Resultado
And	Conjunción	F (falso)
Or	Disyunción	V (verdadero)
Not	Negación	V (verdadero)

Orden de Prioridad de Operadores en una Expresión

Prioridad	Operadores	Descripción
1°	()	Paréntesis
2°	**	Potencia
3°	*, /, MOD, DIV	Multiplicación, División, Módulo
4°	+, -	Suma, Resta
5°	>, <, >=, <=, =, ≠	Comparaciones
6°	NOT	Negación lógica
7°	AND	Conjunción lógica
8°	OR	Disyunción lógica

Bibliografía

PPT Semana 5 (: Introducción a los algoritmos. Representación de algoritmos (DFD, pseudocódigo y código)

Universidad Autónoma de Aguascalientes. *Problemario de Algoritmos Resueltos con Diagramas de Flujo y Pseudocódigo.*

<https://editorial.uaa.mx/docs/algoritmos.pdf>