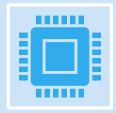
Módulo 1: Fundamentos de la Programación

Curso de Integración de Componentes Software Web

Lógica de Programación



La lógica de programación implica el uso de reglas y secuencias para resolver problemas.

Tipos de operaciones lógicas:

AND (&&): Devuelve verdadero si ambas condiciones son verdaderas.

OR (||): Devuelve verdadero si al menos una condición es verdadera.

NOT (!): Invierte el valor de verdad.

Importancia: Permite crear algoritmos precisos para programas.

Ejemplos:

Tabla de Verdad AND:

True && True = True

True && False = False

Ejemplo en código:

```
if (a > 0 && b > 0) { console.log('Ambos son positivos'); }
```

Estructura de un Programa

- Un programa consta de:
  - Entrada: Datos proporcionados por el usuario.
  - Proceso: Cálculos y operaciones realizadas.
  - Salida: Resultados mostrados al usuario.
- Variables: Espacios de memoria para almacenar datos.
- Estructuras de control: Permiten decisiones (if/else) y repeticiones (for/while).

Ejemplos:

Pseudocódigo:

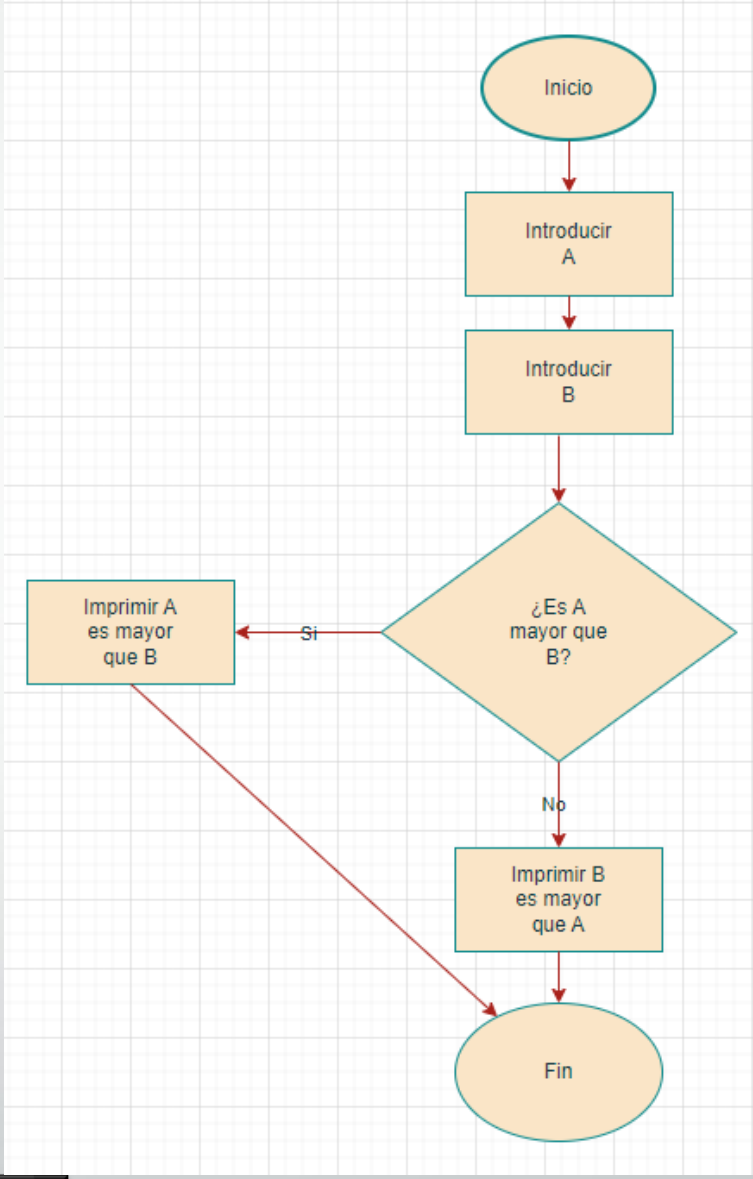
```
Leer A, B
Suma = A + B
Mostrar Suma
```

Código en JS:

```
let nombre = prompt('Ingrese su nombre');
alert('Hola ' + nombre);
```

Diagramas de Flujo

- Son representaciones gráficas de algoritmos, usando símbolos estandarizados.
  - Óvalo: Inicio/Fin
  - Rectángulo: Procesos
  - Rombo: Decisiones
- Ventajas: Facilita la comprensión de la secuencia de pasos en un algoritmo.



Ejemplos:
Ejemplo: Diagrama de flujo para encontrar el mayor de dos números:
Inicio -> Leer A, B -> ¿A > B? -> Sí: Mostrar A, No: Mostrar B -> Fin

Pseudocódigo

Un pseudocódigo usa palabras simples para describir algoritmos.

Estructura:

Inicio/Fin

Leer/Mostrar para E/S

Asignaciones con ←

Decisiones con Si/Entonces

Ejemplos:

```
Inicio
Leer C
F = (C × 9/5) + 32
Mostrar F
Fin
```

Introducción a Objetos

- En POO (Programación Orientada a Objetos), un objeto es una instancia de una clase.
  - Atributos: Características (ej: color, tamaño)
  - Métodos: Acciones (ej: iniciar(), detener())
- Ventajas de la POO:

Clase: "Coche"
- •Atributos: marca, modelo, color, velocidad
- •Métodos: acelerar, frenar, girar

Objeto: "MiCoche"
- •Atributos: marca = "Toyota", modelo = "Corolla", color = "Rojo", velocidad = 0
- •Métodos: puede acelerar, frenar y girar

En resumen:
•La clase es la plantilla, el molde.
•El objeto es la instancia concreta, creada a partir de la plantilla.
•Código en JS:

```
let coche = {marca:'Toyota',
arrancar:function(){alert('Arrancando')}};
```

Reusabilidad: Los objetos se pueden reutilizar en diferentes partes de un programa o en diferentes programas.

Modularidad: Los programas se pueden dividir en módulos más pequeños y fáciles de mantener.

Abstracción: La abstracción permite a los programadores trabajar con objetos de alto nivel **sin** tener que preocuparse por los detalles de implementación.

Encapsulamiento: El encapsulamiento protege los datos internos de los objetos de modificaciones no autorizadas.

Herencia: La herencia permite crear jerarquías de clases que comparten características comunes, lo que reduce la cantidad de código que se debe escribir.

Polimorfismo: El polimorfismo permite a los programadores escribir código que puede trabajar con objetos de diferentes clases de manera uniforme.

Ejemplos de Código

- Mostrar ejemplos en:
  - C (Estructurado): printf('Hola');
  - JavaScript (Scripting): console.log('Hola');
  - Python (POO): print('Hola')

Ejemplos:

Practicar escribiendo 'Hola Mundo' en cada lenguaje.
Ejercicio: Crear un programa para sumar dos números en cada lenguaje.

C:

```
#include <stdio.h>

int main() {
  printf("HolaMundo\n");
  return 0;
}
```

JavaScript:

```
console.log("HolaMundo");
```

Python:

```
print("HolaMundo")
```

Prácticas del Módulo 1

- Actividades sugeridas:
  - Crear tablas de verdad para operaciones lógicas.
  - Escribir pseudocódigos para cálculos simples.
  - Dibujar diagramas de flujo para tareas diarias.
  - Crear objetos en pseudocódigo.