Programación orientada a objetos.

Actividad 0

Equipo #5

Emilio Gómez Breschi

Fátima Lisette Hernández Barajas

Ruth Regina Mendoza Ochoa

Paradigma Procedimental

Programacion procedimental

Es un paradigma de Programacion y su origen parte como una evolución de la programación estructurada (que se basa principalmente en dividir un problema en partes pequeñas).

¿Como aborda problemas?

La programación procedimental también se conoce como programación imperativa y se centra en mejorar la claridad y tiempo de desarrollo de los programas.

Este paradigma descompone el problema en subproblemas y organiza el código en procedimientos también conocidos como subrutinas las cuales contienen una secuencia de instrucciones con orden lógico para realizar tareas específicas.

cada procedimiento puede ser llamado en cualquier momento durante la ejecución del programa, promueve la reutilización del código y la modularidad, ya que los procedimientos pueden ser definidos una vez y utilizados múltiples veces a lo largo del programa. Sin embargo, también puede llevar a un diseño de software menos flexible y más difícil de mantener, especialmente en programas grandes y complejos.

Ejemplo:

c++

```
#include <iostream>

using namespace std;

void saludar() {

    cout << "¡Hola, mundo!" << endl;

}

int main() {
```

```
    saludar();
    return 0;
}
```

En el ejemplo anterior, hemos definido un procedimiento llamado `saludar` que imprime un mensaje en la consola. Luego, en la función `main`, llamamos a este procedimiento para ejecutar su código. Este es un ejemplo básico de cómo se estructura un programa en el paradigma procedimental.

Características del Paradigma Procedimental

1. **Modularidad**: Los programas se dividen en módulos o procedimientos, lo que facilita la comprensión y el mantenimiento del código.

2. **Reutilización de Código**: Los procedimientos pueden ser reutilizados en diferentes partes del programa o incluso en otros programas, lo que reduce la duplicación de código.

3. **Secuencialidad**: La ejecución del programa sigue una secuencia de pasos, lo que facilita el seguimiento del flujo de control.

4. **Abstracción**: Los detalles de implementación de un procedimiento se ocultan detrás de su interfaz, lo que permite a los programadores utilizar procedimientos sin necesidad de comprender su funcionamiento interno.

5. **Control de Flujo:** El paradigma procedimental proporciona estructuras de control (como bucles y condicionales) que permiten a los programadores dirigir el flujo de ejecución del programa de manera efectiva.

Ventajas Del Paradigma Procedimental

1. Muchos lenguajes de programación de propósito general lo admiten.
2. Simplifica el código, lo hace más comprensible.
3. Implementar algoritmos es fácil.
4. Aumenta la reutilización de código.
5. el programa fluye en dirección final lo que lo hace fácil de seguir.

Desventajas del Paradigma Procedimental

1. **Dificultad para Manejar la Complejidad:** A medida que los programas crecen en tamaño y complejidad, la gestión de múltiples procedimientos y su interacción

puede volverse complicada, lo que dificulta la comprensión del sistema en su conjunto.

2. **Dependencia de Estado Global:** Los procedimientos pueden depender de variables globales, lo que puede llevar a efectos secundarios inesperados y dificultar la comprensión del flujo de datos en el programa.

3. **Falta de Encapsulamiento:** A diferencia de los paradigmas orientados a objetos, el paradigma procedimental no proporciona un mecanismo para encapsular datos y comportamientos relacionados, lo que puede llevar a un diseño menos organizado y más propenso a errores.

4. **Dificultad para la Prueba y Validación**: La naturaleza secuencial y la falta de encapsulamiento pueden dificultar la prueba y validación de procedimientos individuales, lo que puede llevar a errores no detectados en el programa.

5. **Rigidez**: La estructura del paradigma procedimental puede hacer que los programas sean menos flexibles y más difíciles de modificar, ya que los cambios en un procedimiento pueden tener efectos en cadena en otras partes del programa.

Problemas de la clase y sus códigos

Tablas de multiplicar

```
#include <iostream>

using namespace std;

void multiplicar(int numero)

{

  for (int i = 1; i <= 10; i++)

  {

    cout << numero << " x " << i << " = " << numero * i << endl;

  }

}

int main()

{

  int numero;
```

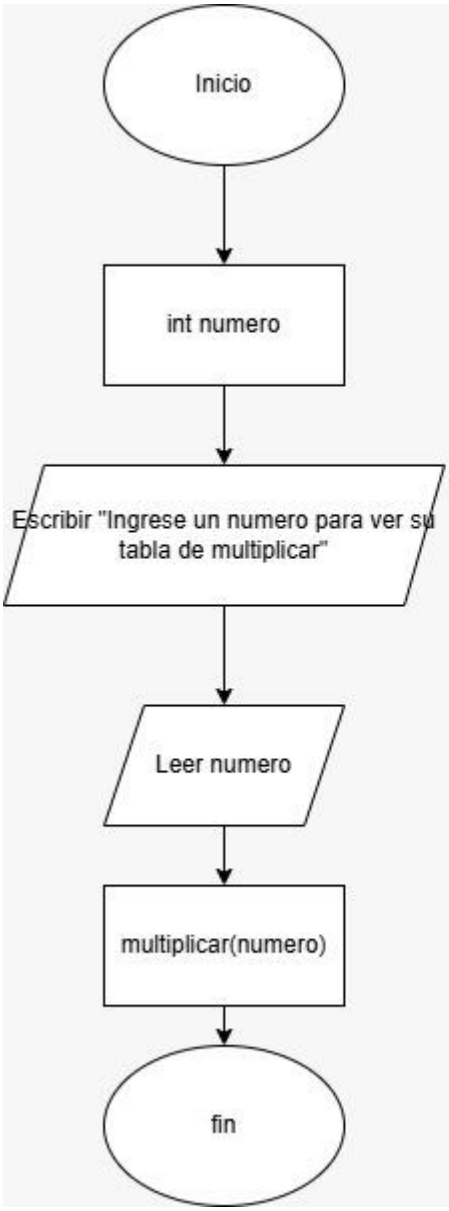
```
  cout << "Ingrese un número para ver su tabla de multiplicar: ";
  cin >> numero;
  multiplicar(numero);
  return 0;
}
```

Suma números del 1-100

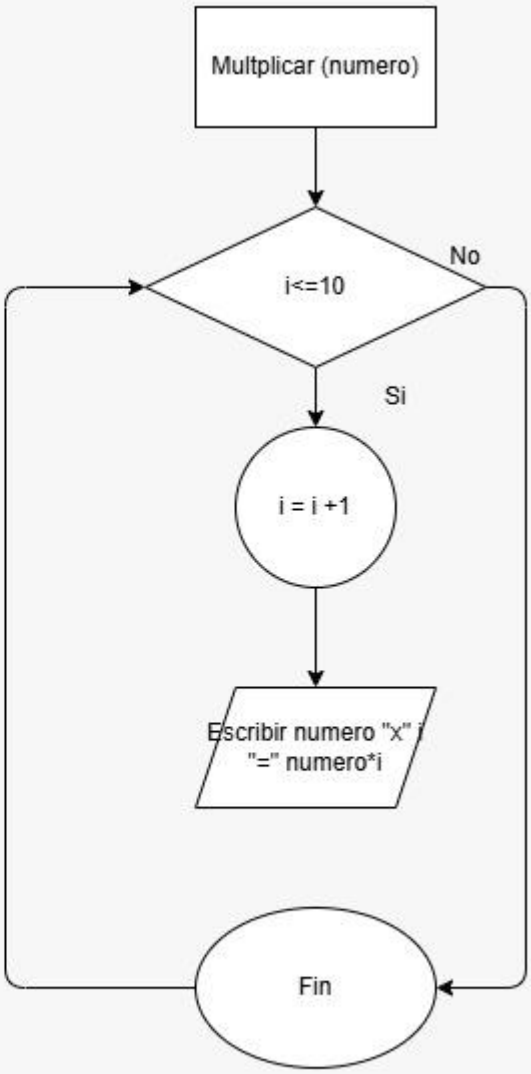
```
#include <iostream>
using namespace std;
int suma()
{
  int total = 0;
  for (int i = 1; i <= 100; i++)
  {
    total += i;
  }
  return total;
}
int main()
{
  cout << "La suma de los números del 1 al 100 es: " << suma() << endl;
  return 0;
}
```

Diagramas:

tablas



Suma numeros 1-100



En los dos problemas aplicamos la abstracción al crear los procedimientos multiplicar() y suma() dentro de los cuales se desarrolla la lógica utilizando ciclos for en ambos casos para obtener los resultados, cuando se necesite hacer cualquiera de estas dos operaciones no será necesario volver a escribir toda la lógica si no que bastara con llamar al procedimiento correspondiente.

conclusión

El paradigma procedimental ha sido fundamental en la evolución de la programación, proporcionando un enfoque claro y estructurado para el desarrollo de software. A pesar de sus desventajas, como la dificultad para manejar la complejidad y la falta de encapsulamiento, sigue siendo ampliamente utilizado en la práctica. La comprensión de este paradigma es esencial para los programadores, ya que sienta las bases para el aprendizaje de enfoques más avanzados, como la programación orientada a objetos.

Bibliografía

1.1.2 Programación procedimental. (s. f.). http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro32/112_programacin_procedimental.html

Spasojevic, A. (2024, 10 abril). ¿Qué es la programación procedimental? phoenixNAP IT Glossary. https://phoenixnap.mx/glosario/programaci%C3%B3n-procesal

Procedural Programming - Definition, Advantages, and Disadvantages. (s. f.). techgeekbuzz.com.https://www.techgeekbuzz.com/blog/procedural-programming/