**Ejercicios**

**Instrucciones, excepciones, métodos y parámetros.**

**Excepciones:**

Las excepciones son eventos inesperados o errores que pueden ocurrir durante la ejecución de un programa. Estos errores pueden deberse a diferentes razones, como entradas incorrectas, archivos que no se pueden encontrar o problemas de memoria. Las excepciones proporcionan una forma de manejar estos errores de manera controlada en lugar de que el programa se bloquee o se comporte de manera inesperada.

Cuando ocurre una excepción, el programa puede "lanzar" la excepción, lo que significa que informa sobre el error y puede detener la ejecución normal del programa. Para manejar una excepción, se utiliza un bloque **try-catch**, donde el código potencialmente problemático se coloca en el bloque **try**, y cualquier excepción que ocurra se captura y se maneja en el bloque **catch**. Esto permite que el programa continúe ejecutándose y, si es necesario, tome para resolver el problema o informar al usuario.

**Métodos:**

Los métodos son bloques de código que realizan una tarea específica o una serie de tareas. Los métodos permiten dividir un programa en partes más pequeñas y manejables, lo que hace que el código sea más organizado y más fácil de entender y mantener. Cada método tiene un nombre y puede aceptar datos de entrada llamados parámetros.

Los métodos pueden ser llamados desde otras partes del programa para realizar sus tareas designadas. Esto promueve la reutilización del código, ya que no es necesario volver a escribir la misma lógica una y otra vez. Al llamar a un método, los valores de los parámetros se pasan a la función, que los utiliza para realizar su tarea y, en algunos casos, puede devolver un valor como resultado.

**Parámetros:**

Los parámetros son valores que se pasan a un método cuando se llama. Estos valores son necesarios para que el método realice su tarea correctamente. Los parámetros permiten que los métodos sean flexibles y adaptables, ya que pueden aceptar diferentes datos como entrada.

Los parámetros se definen en la firma del método y se utilizan dentro del cuerpo del método para realizar cálculos o acciones específicas. Pueden ser de diferentes tipos de datos, como números enteros, cadenas de texto o incluso objetos personalizados.

Cuando se llama a un método, los argumentos se proporcionan para cada parámetro en el orden correcto. Estos argumentos son los valores reales que se utilizarán en el método durante su ejecución. Los parámetros son una parte fundamental de cómo los métodos reciben y utilizan datos en un programa.

**Ejemplo de excepciones en C#:**

using System;

class Program

{

static void Main()

{

try

{

int divisor = 0;

int resultado = 10 / divisor; // Esto generará una excepción de división por cero

}

catch (DivideByZeroException ex)

{

Console.WriteLine("Error: " + ex.Message);

}

}

}

En este ejemplo, hemos intentado dividir 10 por cero, lo que genera una excepción de división por cero ( **DivideByZeroException**). El programa utiliza un bloque **try-catch**para capturar y manejar la excepción, mostrando un mensaje de error en la consola.

**Ejemplo de métodos en C#:**

using System;

class Program

{

static void Main()

{

// Llamamos al método Saludar

Saludar("Juan");

}

// Definición de un método llamado Saludar

static void Saludar(string nombre)

{

Console.WriteLine("¡Hola, " + nombre + "!");

}

}

En este ejemplo, hemos definido un método llamado **Saludar** que toma un parámetro de tipo cadena (**nombre**). Luego, llamamos a este método con un nombre específico ("Juan"). El método imprime un saludo personalizado en la consola.

**Ejemplo de parámetros en C#:**

using System;

class Program

{

static void Main()

{

int resultado = Sumar(5, 3); // Llamamos al método Sumar y pasamos dos números como argumentos

Console.WriteLine("La suma es: " + resultado);

}

// Definición de un método llamado Sumar con dos parámetros

static int Sumar(int num1, int num2)

{

int suma = num1 + num2;

return suma;

}

}

En este ejemplo, hemos definido un método llamado **Sumar** que toma dos parámetros (**num1**y **num2**). Luego, llamamos a este método y le pasamos dos números como argumentos. El método suma los números y devuelve el resultado, que se muestra en la consola.

Estos ejemplos simples ilustran cómo se utilizan las excepciones, los métodos y los parámetros en C#. Las excepciones permiten manejar errores, los métodos encapsulan funcionalidad y los parámetros permiten que los métodos acepten datos de entrada.

**Ejercicio 1: Suma de Vectores**

Escribe un método llamado **SumarVectores** que tome dos vectores de enteros como parámetros y devuelva un nuevo vector que sea la suma de los dos vectores. Si los vectores tienen longitudes diferentes, lanza una excepción **ArgumentException** para indicar que los vectores deben ser del mismo tamaño.

**Ejercicio 2: Multiplicación de matrices**

Escribe un método llamado **MultiplicarMatrices** que tome dos matrices como parámetros y devuelva una nueva matriz que sea el resultado de multiplicar las dos matrices. Asegúrese de manejar adecuadamente las dimensiones de las matrices y lanza una excepción si no se pueden multiplicar.

**Ejercicio 3: Ordenar Vector con Excepciones Personalizadas**

Escribe un método llamado **OrdenarVector** que tome un vector de enteros como parámetro y lo ordene en orden ascendente. Si el vector contiene valores no enteros, lanza una excepción de llamada personalizada **ValorNoEnteroException**.

**Ejercicio 4: Matriz Identidad**

Escribe un método llamado **CrearMatrizIdentidad** que tome un entero n como parámetro y devuelva una matriz identidad de tamaño nxn. Si n es menor o igual a 0, lanza una excepción **ArgumentException**.

**Ejercicio 5: Producto Escalar de Vectores**

Escribe un método llamado **ProductoEscalar** que tome dos vectores de enteros y calcula su producto escalar. Asegúrese de manejar correctamente los vectores de diferentes longitudes y lanza una excepción si son de diferente tamaño.