Metodología y programación estructurada.

Paso de arreglos a funciones

Integrantes:

- Slles Abraham Flores Urcuyo
- Silvio Ernesto Mejia Garcia

Docente: Silvia Gigdalia Ticay Lopez

Ejercicio No. 1 (Guia didáctica) Crea un programa que solicite al usuario la base y la altura de un triángulo. El programa debe incluir una función llamada calcular_area_triangulo que reciba como parámetros la base y la altura, y devuelva el área del triángulo. La fórmula para calcular el área es: Área =(base * altura) /2

Estructura General

El código se organiza en dos partes principales:

1. La clase Program, que contiene el método Main, el punto de entrada de la aplicación.
2. La clase Class1, que se encarga de calcular el área del triángulo.

Detalle del Código

1. Importación de Espacios de Nombres

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Clcular_Area_Triagulo_Modelar.Class1;
```

- Se importan varios espacios de nombres que permiten utilizar funcionalidades básicas de C#, como entrada/salida y colecciones.
- La línea using static Clcular_Area_Triagulo_Modelar.Class1; permite acceder a los miembros estáticos de Class1 sin necesidad de prefijar el nombre de la clase.

2. Definición del Namespace

```
namespace Clcular_Area_Triagulo_Modelar
```

- Se define un espacio de nombres llamado Clcular_Area_Triagulo_Modelar, que agrupa las clases relacionadas en un contexto específico.

3. Clase Principal Program

```
internal class Program
{
    static void Main(string[] args)
    {
        // Crear una instancia de la clase Calculadora
        Class1 calculadora = new Class1();
```

- La clase Program es la clase principal que contiene el método Main.
- Se crea una instancia de la clase Class1, que incluye el método para calcular el área del triángulo.

4. Solicitar la Cantidad de Triángulos

```
        // Solicitar al usuario la cantidad de triángulos
        Console.Write("Ingrese la cantidad de triángulos: ");
        int cantidadTriangulos = Convert.ToInt32(Console.ReadLine());
```

- Se solicita al usuario que ingrese la cantidad de triángulos para los cuales desea calcular el área. La entrada se convierte a un entero.

5. Crear Listas para Almacenar Bases y Alturas

```
        // Crear listas para almacenar las bases y alturas
        List<double> bases = new List<double>();
        List<double> alturas = new List<double>();
```

- Se declaran dos listas de tipo double: una para almacenar las bases de los triángulos y otra para las alturas.

6. Solicitar Base y Altura para Cada Triángulo

```
        // Solicitar la base y altura para cada triángulo
        for (int i = 0; i < cantidadTriangulos; i++)
        {
            Console.Write($"Ingrese la base del triángulo {i + 1}: ");
            double baseTriangulo = Convert.ToDouble(Console.ReadLine());
            bases.Add(baseTriangulo);

            Console.Write($"Ingrese la altura del triángulo {i + 1}: ");
            double alturaTriangulo = Convert.ToDouble(Console.ReadLine());
            alturas.Add(alturaTriangulo);
        }
```

- Se utiliza un bucle for para solicitar al usuario la base y la altura de cada triángulo.
- Para cada triángulo, se pide la base y la altura, se convierten a tipo double y se añaden a las listas correspondientes.

7. Calcular y Mostrar el Área de Cada Triángulo

```
        // Calcular y mostrar el área de cada triángulo
        for (int i = 0; i < cantidadTriangulos; i++)
        {
            double area = calculadora.CalcularAreaTriangulo(bases[i], alturas[i]);
            Console.WriteLine($"El área del triángulo {i + 1} es: {area}");
        }
    }
}
```

- Se utiliza otro bucle for para calcular y mostrar el área de cada triángulo.
- Se llama al método CalcularAreaTriangulo de la instancia calculadora, pasando la base y la altura correspondientes.
- Se imprime el área calculada para cada triángulo.

Clase externa Class1

```
internal class Class1
{
    // Método que calcula el área del triángulo
    public double CalcularAreaTriangulo(double baseTriangulo, double alturaTriangulo)
    {
        return (baseTriangulo * alturaTriangulo) / 2;
    }
}
```

- La clase Class1 contiene un método público CalcularAreaTriangulo, que toma como parámetros la base y la altura del triángulo.
- El método calcula el área utilizando la fórmula del área del triángulo: $\frac{base \times altura}{2}$ y devuelve el resultado.

Ejercicio No 4 (Ejercicios modular structs) Crea un programa que use un arreglo estático para almacenar números y una función que calcule el factorial de cada número, el cual es enviado a un segundo arreglo. Muestra los resultados, es decir ambos arreglos
- El número es leído en la función principal Main y es enviado como parámetro a la función que calcula el factorial. Recuerda que el factorial no se calcula para números negativos. Por lo tanto, al arreglo original sólo debes guardar los números positivos o cero.
- El programa se repetirá mientras el usuario lo desea.

Estructura General
El código se organiza en dos partes principales:

1. **La clase Program,** que contiene el método Main, el punto de entrada de la aplicación.
2. **La clase Class1 y la estructura NumeroFactorial,** que se encargan de calcular el factorial y almacenar los resultados.

Detalle del Código

1. Importación de Espacios de Nombres
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Calcular_Factorial_Modular.Class1;
```
- Se importan varios espacios de nombres que permiten utilizar funcionalidades básicas de C#, como entrada/salida y colecciones.
- La línea using static Calcular_Factorial_Modular.Class1; permite acceder a los miembros estáticos de Class1 sin necesidad de prefijar el nombre de la clase.

2. Definición del Namespace
```
namespace Calcular_Factorial_Modular
```
- Se define un espacio de nombres llamado Calcular_Factorial_Modular, que agrupa las clases relacionadas en un contexto específico.

3. Clase Principal Program
```
internal class Program
{
    static void Main(string[] args)
    {
        // Crear una lista para almacenar números y sus factoriales
        List<NumeroFactorial> numerosFactoriales = new List<NumeroFactorial>();
```
- La clase Program es la clase principal que contiene el método Main.
- Se declara una lista llamada numerosFactoriales que almacenará instancias de la estructura NumeroFactorial, que contendrá los números y sus factoriales.

4. Instancia de la Clase Calculadora

```
        // Crear una instancia de la clase Calculadora
        Class1 calculadora = new Class1();
```

- Se crea una instancia de la clase Class1, que incluye el método para calcular el factorial.

5. Bucle para Solicitar Números al Usuario

```
        // Bucle para solicitar números al usuario
        while (true)
        {
            Console.Write("Ingrese un número positivo (o 0 para terminar): ");
            int numero = Convert.ToInt32(Console.ReadLine());
```

- Se inicia un bucle infinito que solicitará números al usuario hasta que este decida terminar.
- Se pide al usuario que ingrese un número positivo o 0 para finalizar. La entrada se convierte a un entero.

6. Condición para Terminar el Bucle

```
            // Condición para terminar el bucle
            if (numero < 0)
            {
                Console.WriteLine("Por favor, ingrese solo números positivos o cero.");
                continue;
            }
```

- Si el número ingresado es negativo, se muestra un mensaje de error y se utiliza continue para volver al inicio del bucle.

7. Almacenar el Número y su Factorial

```
            // Almacenar el número y su factorial en la lista
            if (numero > 0)
            {
                NumeroFactorial nf = new NumeroFactorial
                {
                    Numero = numero,
                    Factorial = calculadora.CalcularFactorial(numero)
                };
                numerosFactoriales.Add(nf);
            }
```

- Si el número es mayor que 0, se crea una nueva instancia de NumeroFactorial y se asignan el número y su factorial (calculado mediante el método CalcularFactorial de la instancia calculadora).
- Esta instancia se agrega a la lista numerosFactoriales.

8. Preguntar al Usuario si Desea Continuar

```
            // Preguntar al usuario si desea continuar
            Console.Write("¿Desea ingresar otro número? (s/n): ");
            string respuesta = Console.ReadLine().ToLower();
            if (respuesta != "s")
```

```
        {
            break;
        }
    }
```

- Después de cada entrada, se pregunta al usuario si desea ingresar otro número. Si la respuesta no es "s", se rompe el bucle.

9. Mostrar los Resultados

```
    // Mostrar los resultados
    Console.WriteLine("\nNúmeros ingresados y sus factoriales:");
    foreach (var item in numerosFactoriales)
    {
        Console.WriteLine($"Número: {item.Numero}, Factorial: {item.Factorial:e}");
    }
  }
}
```

- Al finalizar el bucle, se imprime una lista de los números ingresados y sus respectivos factoriales utilizando un bucle foreach. Se utiliza la notación científica (:e) para mostrar el resultado del factorial.
-

Clase externa Class1 y Estructura NumeroFactorial

```
internal struct NumeroFactorial
{
  public int Numero;
  public double Factorial;
}
```

- La estructura NumeroFactorial contiene dos campos: Numero (un entero que representa el número ingresado) y Factorial (un número de tipo double que representa el factorial de ese número).

```
internal class Class1
{
  // Método que calcula el factorial de un número
  public double CalcularFactorial(int numero)
  {
    if (numero < 0)
    {
            throw new ArgumentException("No se puede calcular el factorial de números negativos.");
    }

    double factorial = 1;
    for (int i = 1; i <= numero; i++)
    {
        factorial *= i;
```

```
        }
        return factorial;
    }
}
```

- La clase Class1 incluye un método público CalcularFactorial, que toma un número entero y calcula su factorial. Si se intenta calcular el factorial de un número negativo, se lanza una excepción (ArgumentException).
- El cálculo se realiza mediante un bucle for que multiplica los números desde 1 hasta el número ingresado.