INSTITUTO KHIPU

Semestre III

Sesión 07

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

HERRAMIENTAS DE PROGRAMACIÓN – C#

Tema:

PARAMETROS DE MÉTODOS-PALABRAS CLAVE PARAMS, IN.

Parámetros de Métodos en C#

Cuando trabajamos con métodos en C#, a veces necesitamos pasar datos a esos métodos. Hay diferentes maneras de hacerlo, y hoy hablaremos sobre dos palabras clave que facilitan esta tarea: params e in.

1. La Palabra Clave params

¿Qué es params?

La palabra clave params permite que un método acepte un número variable de argumentos. Esto significa que no tienes que definir exactamente cuántos parámetros vas a pasar. Puedes enviarle cualquier cantidad de elementos del mismo tipo.

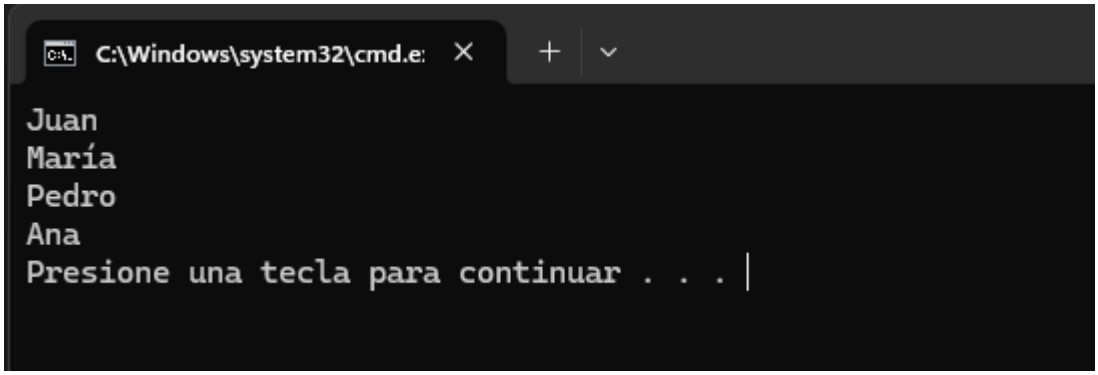
Ejemplo de params

Imagina que queremos crear un método que imprima una lista de nombres. Podríamos usar params para que el método acepte cualquier cantidad de nombres:

```
using System;

namespace ConsoleApp1
{
    0 referencias
    class Program
    {
        3 referencias
        public static void ImprimirNombres(params string[] nombres)
        {
            foreach (var nombre in nombres)
            {
                Console.WriteLine(nombre);
            }
        }

        0 referencias
        static void Main(string[] args)
        {
            ImprimirNombres("Juan", "María", "Pedro"); // Imprime: Juan, María, Pedro
            ImprimirNombres("Ana"); // Imprime: Ana
            ImprimirNombres(); // No imprime nada
        }
    }
}
```



¿Por qué usar params?

- **Flexibilidad**: Puedes llamar al método con diferentes cantidades de argumentos.
- **Código más limpio**: No necesitas crear un arreglo antes de llamar al método.

2. La Palabra Clave in

¿Qué es in?

La palabra clave in se utiliza para pasar un parámetro por referencia, lo que significa que se pasa la dirección de la variable en lugar de su valor. Sin embargo, a diferencia de pasar por referencia normal, con in no se puede modificar el valor de la variable dentro del método.

Ejemplo de in

Ahora, supongamos que queremos trabajar con un rectángulo. Usaremos in para pasar un rectángulo a un método que calcule su área sin permitir que se modifique el rectángulo original.

```
{
    0 referencias
    class Program
    {
        4 referencias
        public struct Rectangulo
        {
            public int Ancho;
            public int Alto;

            1 referencia
            public Rectangulo(int ancho, int alto)
            {
                Ancho = ancho;
                Alto = alto;
            }
        }
        1 referencia
        public static void CalcularArea(in Rectangulo rectangulo)
        {
            int area = rectangulo.Ancho * rectangulo.Alto;
            Console.WriteLine($"El área del rectángulo es: {area}");
        }

        0 referencias
        static void Main(string[] args)
        {
            Rectangulo miRectangulo = new Rectangulo(5, 10);
            CalcularArea(in miRectangulo); // Muestra: El área del rectángulo es: 50

            // Intentar modificar el rectángulo aquí causaría un error
            // rectangulo.Ancho = 7; // Esto no es permitido
        }
    }
}
```

```
C:\Windows\system32\cmd.e:
El área del rectángulo es: 50
Presione una tecla para continuar . . .
```

Explicación:

- En el método CalcularArea, se usa in para asegurar que el rectángulo pasado no puede ser modificado.
- Esto es útil cuando se trabaja con estructuras que pueden contener mucha información, ya que evita la copia innecesaria de datos.

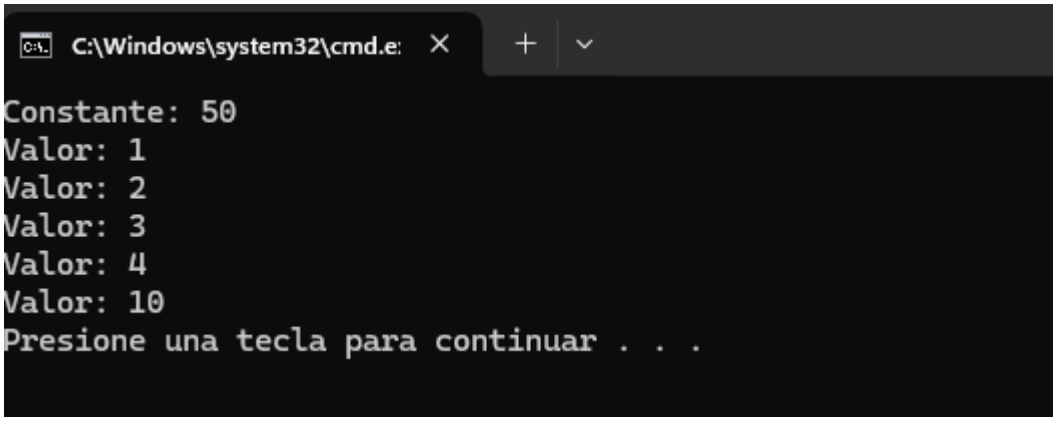
¿Por qué usar in?

- **Eficiencia**: Si tienes estructuras grandes, al usar in evitas la copia de datos innecesaria.
- **Seguridad**: Garantiza que el valor no será cambiado dentro del método, protegiendo tus datos.

3. Ejemplo combinando params e in:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApp1
8   {
        2 referencias
9       internal class ejemploCombinado
10      {
11          // Método que recibe una referencia inmutable y un número variable de parámetros
            1 referencia
12          public void ProcesarDatos(in int constante, params int[] valores)
13          {
14              Console.WriteLine($"Constante: {constante}");
15              foreach (int valor in valores)
16              {
17                  Console.WriteLine($"Valor: {valor}");
18              }
19          }
20      }
21  }
22
```

```
            0 referencias
8           static void Main(string[] args)
9           {
10              int constante = 50;
11              ejemploCombinado ejemplo = new ejemploCombinado();
12              // Llamando al método con la constante y varios parámetros
13              ejemplo.ProcesarDatos(in constante, 1, 2, 3, 4,10); // Salida: Constante: 50 y Valores: 1, 2, 3, 4, 10
14          }
15      }
16  }
17
```



Explicación del Ejemplo:

En este ejemplo, el método ProcesarDatos utiliza tanto in como params. Se pasa una constante por referencia, pero su valor no se modifica, y al mismo tiempo se acepta una cantidad variable de parámetros adicionales usando params.

Conclusión

Las palabras clave params e in en C# son herramientas poderosas que permiten flexibilidad y control sobre cómo se pasan los parámetros a los métodos. Mientras que params permite trabajar con un número indefinido de argumentos, in asegura que los parámetros pasados por referencia no sean modificados. Ambos conceptos pueden mejorar tanto la eficiencia como la legibilidad del código cuando se usan adecuadamente.

EJERCICIOS

1. **Suma de números enteros con params**

   Desarrolla un método llamado SumaNumeros que acepte un número variable de enteros usando params. El método debe sumar todos los números recibidos y retornar el resultado.

2. **Validación de tipos con object[] y params**

   Crea un método llamado MostrarDetalles que acepte un arreglo de objetos de diferentes tipos (enteros, cadenas, booleanos, etc.) usando params object[]. El método debe recorrer los elementos y mostrar el tipo de dato de cada uno.

INSTITUTO

KHIPU