INSTITUTO KHIPU

Semestre III

Sesión 09

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

HERRAMIENTAS DE PROGRAMACIÓN – C#

Tema:

ARGUMENTOS CON NOMBRE Y OPCIONALES PALABRAS CLAVE READONLY, CONS

ARGUMENTOS CON NOMBRES, OPCIONALES Y PALABRAS CLAVE READONLY, CONST

Argumentos Opcionales

Los **argumentos opcionales** son una característica poderosa en C# que permite a los desarrolladores definir parámetros en métodos que no siempre necesitan ser proporcionados cuando el método es llamado. Esta flexibilidad mejora la legibilidad y la mantenibilidad del código, facilitando su uso en diversas situaciones. A continuación, exploraremos el concepto de argumentos opcionales, su sintaxis, ejemplos prácticos y ejercicios propuestos para que los estudiantes de nivel básico puedan comprender y aplicar este concepto.

Concepto de Argumentos Opcionales

Cuando definimos un método, normalmente especificamos todos los parámetros que requiere. Sin embargo, a veces queremos que ciertos parámetros tengan valores predeterminados. Estos son los argumentos opcionales. Cuando se llaman a estos métodos, si no se proporciona un argumento opcional, el valor predeterminado se usará automáticamente.

Sintaxis

Para declarar un argumento opcional, se asigna un valor predeterminado al parámetro en la definición del método. La sintaxis es la siguiente:

```
public void NombreDelMetodo(tipo parametro1, tipo parametroOpcional = valorPredeterminado)
{
    // Cuerpo del método
}
```

Ejemplo Práctico

Imaginemos que queremos crear un método que salude a una persona.

Podemos hacer que el saludo tenga un argumento opcional para especificar el idioma.

```
1   using System;
2
3   namespace ConsoleApp1
4   {
        1 referencia
5       internal class Saludo
6       {
            2 referencias
7           public void DecirHola(string nombre, string idioma = "Español")
8           {
9               if (idioma == "Inglés")
10              {
11                  Console.WriteLine($"Hello, {nombre}!");
12              }
13              else
14              {
15                  Console.WriteLine($"¡Hola, {nombre}!");
16              }
17          }
18      }
19  }
20
```

```
using System;

namespace ConsoleApp1
{
    0 referencias
    class Program
    {

        0 referencias
        static void Main(string[] args)
        {
            var saludo = new Saludo();
            saludo.DecirHola("Juan"); // Salida: ¡Hola, Juan!
            saludo.DecirHola("Emily", "Inglés"); // Salida: Hello, Emily!
        }
    }
}

```

Introducción a los Argumentos con Nombres en C#

¿Qué son los Argumentos con Nombres?

En C#, los argumentos con nombres (también conocidos como argumentos nombrados) permiten especificar los parámetros de un método utilizando el nombre del parámetro en lugar de seguir el orden definido en la firma del método. Esto proporciona claridad y flexibilidad, especialmente cuando un método tiene varios parámetros, algunos de los cuales pueden ser opcionales.

Ventajas de Usar Argumentos con Nombres

1. **Claridad:** Hacer explícito el propósito de cada argumento mejora la legibilidad del código.
2. **Flexibilidad:** Puedes omitir argumentos opcionales y solo especificar aquellos que deseas cambiar.
3. **Mantenimiento:** Facilita la actualización de métodos sin afectar el código que los llama.

Sintaxis de los Argumentos con Nombres

Cuando llamas a un método que acepta argumentos con nombres, usas la siguiente sintaxis:

```
NombreDelMetodo(parametro1: valor1, parametro2: valor2);
```

Ejemplo Práctico

Vamos a definir un método que simula la creación de un perfil de usuario. Este método aceptará varios parámetros, y veremos cómo utilizar argumentos con nombres.

```
using System;

namespace ConsoleApp1
{
    2 referencias
    internal class crearPerfil
    {
        3 referencias
        public void CrearPerfil(string nombre, int edad, string ciudad = "Desconocida", string pais = "Desconocido")
        {
            Console.WriteLine($"Nombre: {nombre}");
            Console.WriteLine($"Edad: {edad}");
            Console.WriteLine($"Ciudad: {ciudad}");
            Console.WriteLine($"País: {pais}");
        }
    }
}
```

PROGRAMA PRINCIPAL

```
using System;

namespace ConsoleApp1

{
    0 referencias
    class Program
    {

        0 referencias
        static void Main(string[] args)
        {
            crearPerfil crear = new crearPerfil();
            // Usando argumentos con nombres

            crear.CrearPerfil(nombre: "Alice", edad: 30, ciudad: "Madrid");

            // Omite el parámetro "pais" ya que es opcional
            crear.CrearPerfil(nombre: "Bob", edad: 25);

            // Cambia el orden de los parámetros
            crear.CrearPerfil(pais: "México", ciudad: "Cancún", nombre: "Charlie", edad: 28);
        }
    }
}
```

RESULTADO ESPERADO

```
C:\Windows\system32\cmd.e:
Nombre: Alice
Edad: 30
Ciudad: Madrid
País: Desconocido
Nombre: Bob
Edad: 25
Ciudad: Desconocida
País: Desconocido
Nombre: Charlie
Edad: 28
Ciudad: Cancún
País: México
Presione una tecla para continuar . . .
```

Palabras Clave readonly y const en C#

En C#, las palabras clave readonly y const se utilizan para declarar variables cuyo valor no puede cambiar una vez que se han asignado. Sin embargo, hay diferencias importantes entre ambas, y es crucial comprender cómo y cuándo usar cada una.

1. La Palabra Clave const

–La palabra clave const se utiliza para declarar constantes en C#. Estas son variables cuyo valor es fijo en el momento de la compilación y no puede cambiar en tiempo de ejecución. Cuando declares una constante, debes inicializarla en el momento de su declaración.

Ejemplo de const:

```
using System;

class Program
{
    const double PI = 3.14159;

    static void Main()
    {
        Console.WriteLine("El valor de PI es: " + PI);
        // PI = 3.14; // Esto generará un error de compilación
    }
}
```

Características de const:

- Debe ser inicializada en el momento de la declaración.
- Su valor no puede cambiar después de ser asignado.
- Se utiliza a menudo para valores que son inmutables y conocidos en tiempo de compilación (por ejemplo, constantes matemáticas, límites, etc.).

2. La Palabra Clave readonly

La palabra clave readonly se utiliza para declarar campos de solo lectura en una clase. A diferencia de las constantes, los campos readonly pueden ser inicializados en el constructor de la clase, lo que les permite recibir valores en tiempo de ejecución. Sin embargo, una vez que se asigna un valor, este no puede cambiar.

Ejemplo de readonly:

```
using System;

class Circulo
{
    public readonly double Radio;

    public Circulo(double radio)
    {
        Radio = radio; // Se puede asignar en el constructor
    }

    public double CalcularArea()
    {
        return Math.PI * Radio * Radio;
    }
}

class Program
{
    static void Main()
    {
        Circulo circulo1 = new Circulo(5);
        Console.WriteLine("El área del círculo es: " + circulo1.CalcularArea());
        // circulo1.Radio = 10; // Esto generará un error de compilación
    }
}
```

Características de readonly:

- Puede ser asignada en el momento de la declaración o en el constructor.
- Su valor no puede cambiar después de que se ha establecido.
- Se utiliza comúnmente para almacenar valores que deben permanecer constantes a lo largo de la vida de un objeto.

Diferencias Clave entre const y readonly

Característica	const	readonly
Inicialización	Debe ser inicializada en la declaración	Puede ser inicializada en el constructor o en la declaración
Tiempo de evaluación	En tiempo de compilación	En tiempo de ejecución
Usos comunes	Valores fijos conocidos	Valores que pueden variar en tiempo de ejecución

INSTITUTO

KHIPU