INSTITUTO KHIPU

Semestre III

Sesión 06

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

HERRAMIENTAS DE PROGRAMACIÓN – C#

Tema:

PARAMETROS DE MÉTODOS-PASAR UN ARREGLO DE TIPO OBJECT COMO PARÁMETRO

Concepto de Arreglos de tipo object

En C#, un arreglo de tipo object (object[]) puede contener elementos de cualquier tipo de datos, ya que todos los tipos de C# derivan de la clase base object. Esto hace que los arreglos de tipo object sean extremadamente versátiles, permitiendo la combinación de diferentes tipos en un solo arreglo.

Uso de Arreglos de object

1. **Flexibilidad:**
    - Permite almacenar diferentes tipos de datos en un mismo arreglo, lo que es útil en situaciones donde la homogeneidad de los datos no es estrictamente necesaria.
    - Ejemplos de uso incluyen la manipulación de datos de entrada de usuario, la creación de estructuras de datos dinámicas o cuando se trabaja con bibliotecas de terceros que requieren arreglos de object.
2. **Casting:**
    - Cuando se accede a los elementos de un arreglo de object, es común que necesites convertir (o "cast") el elemento de object a su tipo original. Esto es necesario porque el compilador no conoce el tipo real del elemento en el momento de la compilación.
    - El casting se puede realizar de dos maneras: **casting explícito** o usando el operador as.

Ejemplo Avanzado: Uso de Arreglos de object con Casting

Vamos a ver un ejemplo más avanzado en el que se utiliza un arreglo de object que contiene diferentes tipos, y luego se procesa cada tipo utilizando casting.

Clase Object

```
using System;

namespace ConsoleApp1
{
    1 referencia
    internal class Object
    {
        // Método que acepta un arreglo de tipo object como parámetro
        1 referencia
        public static void ProcesarArreglo(object[] elementos)
        {
            Console.WriteLine("Procesando arreglo de tipo object...");
            foreach (var elemento in elementos)
            {
                // Usando el casting explícito para procesar diferentes tipos
                if (elemento is int)
                {
                    int numero = (int)elemento;
                    Console.WriteLine($"Número entero: {numero}, Doble: {numero * 2}");
                }
                else if (elemento is string)
                {
                    string texto = (string)elemento;
                    Console.WriteLine($"Cadena de texto: {texto}, Longitud: {texto.Length}");
                }
                else if (elemento is bool)
                {
                    bool valorBooleano = (bool)elemento;
                    Console.WriteLine($"Valor booleano: {valorBooleano}, Negado: {!valorBooleano}");
                }
                else if (elemento is double)
                {
                    double decimalNumero = (double)elemento;
                    Console.WriteLine($"Número decimal: {decimalNumero}, Redondeado: {Math.Round(decimalNumero)}");
                }
                else
                {
                    Console.WriteLine("Tipo no reconocido.");
                }
            }
        }
    }
}
```

Llamando a la clase Object desde la clase Program

```
using System;

namespace ConsoleApp1

{
    0 referencias
    class Program
    {

        0 referencias
        static void Main(string[] args)
        {
            // Crear un arreglo de tipo object que contiene diferentes tipos de datos
            object[] arreglo = new object[] { 42, "Hola", true, 3.14, 'A' };

            // Llamar al método pasando el arreglo como parámetro
            Object.ProcesarArreglo(arreglo);
        }
    }
}

```

RESULTADO ESPERADO

```
C:\Windows\system32\cmd.e:
Procesando arreglo de tipo object...
Número entero: 42, Doble: 84
Cadena de texto: Hola, Longitud: 4
Valor booleano: True, Negado: False
Número decimal: 3.14, Redondeado: 3
Tipo no reconocido.
Presione una tecla para continuar . . .
```

Consideraciones al Usar Arreglos de object

1. **Pérdida de Seguridad de Tipos:**
    - Dado que los elementos se almacenan como object, se pierde la comprobación de tipos en tiempo de compilación. Esto puede llevar a errores en tiempo de ejecución si se realiza un casting incorrecto.
    - Siempre es una buena práctica utilizar is o as para verificar el tipo antes de realizar el casting.

2. **Rendimiento:**
    - El uso de object puede tener un impacto en el rendimiento debido a la necesidad de realizar casting y la posible activación del recolector de basura si se utilizan muchos tipos de datos. Esto es especialmente relevante en aplicaciones de alto rendimiento o en situaciones donde se manejan grandes volúmenes de datos.
3. **Alternativas:**
    - En lugar de usar object[], considera usar List<object>, Dictionary<string, object> o clases genéricas si es posible. Estas alternativas permiten un manejo más seguro y flexible de los datos.

Conclusión:

Pasar un arreglo de tipo object como parámetro en C# es una técnica poderosa que permite la manipulación de datos heterogéneos. Sin embargo, es importante ser consciente de las consideraciones relacionadas con la seguridad de tipos, rendimiento y legibilidad del código. Con un manejo cuidadoso y validaciones adecuadas, los arreglos de tipo object pueden ser muy útiles en el desarrollo de software.

INSTITUTO

KHIPU