Diferencias entre Funciones y Métodos

Función: Es un bloque de código con instrucciones cuyo objetivo es realizar una función específica y retornar un valor como resultado.

Procedimiento: Es un bloque de código con instrucciones que cumple una tarea específica y no retorna un valor como resultado. En C# no existe explícitamente una palabra reservada para implementar un procedimiento, se implementa como una función que no retorna ningún valor (**void** que significa vacío).

Clase: Una clase define un tipo de objeto, pero no es un objeto en sí. Una clase es un molde o plantilla o un template. Un objeto es una entidad concreta basada en una clase y, a veces, se lo conoce como la instancia de una clase. En programación orientada a objetos (POO), es una forma de encapsular funcionalidad, contiene atributos que representan a los datos del objeto instanciado y métodos que son la funcionalidad del objeto.

Método: Un método en programación orientada a objetos en C#, está implementado con una función que retorna o no un resultado, y puede tener parámetros que son valores que ingresan al método cuando se lo llama.

```
public  class ConsoleTest
{
        public void Saludo()
        {
                Console.WriteLine("¡Hola!");
        }

        public int Suma(int a, int b)
        {
                return a + b;
        }
}
```

En el ejemplo anterior,

- **ConsoleTest** es una clase que puede instanciarse en un objeto
- **Saludo** es un método y un procedimiento. Como no devuelve un valor es tipo **void**
- **Suma** es una función y devuelve int

Parámetros

Un parámetro es un dato que ingresa a un método a través de su llamada. Todos los tipos de C# son tipos de valor o tipos de referencia (para más detalles consulte el siguiente link Tipos y variables. De forma predeterminada, los tipos de valor y los tipos de referencia se pasan a un método por valor.

Pasar parámetros por valor

Cuando un tipo de valor se pasa a un método por valor, se pasa una copia del objeto y no el propio objeto. Por lo tanto, los cambios realizados en el objeto en el método llamado no tienen ningún efecto en el objeto original cuando el control vuelve al autor de la llamada.

En el ejemplo siguiente se pasa un tipo de valor a un método por valor, y el método llamado intenta cambiar el valor del tipo de valor. Define una variable de tipo int, que es un tipo de valor, inicializa su valor en 20 y lo pasa a un método denominado **ModificarValor** que cambia el valor de la variable a 30. Pero cuando el método vuelve, el valor de la variable no cambia.

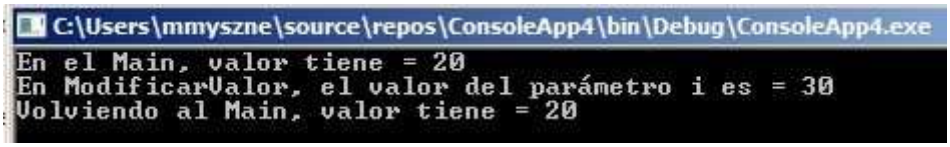
```
using System;

public class Ejemplo
{
        public static void Main()

        {

        int valor = 20;
        Console.WriteLine("En el Main, valor tiene = {0}", valor);
        ModificarValor(valor);
        Console.WriteLine("Volviendo al Main, valor tiene = {0}", valor);
        }

        static void ModificarValor(int i)
        {
        i = 30;
        Console.WriteLine("En ModificarValor, el valor del parámetro i es = {0}"   , i);
        return;
        }
}
```



Cuando un objeto de un tipo de referencia se pasa a un método por valor, se pasa por valor una referencia al objeto. Es decir, el método no recibe el objeto concreto, sino un argumento que indica la ubicación del objeto. Si cambia un miembro del objeto mediante esta referencia, el cambio se reflejará en el objeto cuando el control vuelva al método de llamada. Pero el reemplazo del objeto pasado al método no tendrá ningún efecto en el objeto original cuando el control vuelva al autor de la llamada.

En el ejemplo siguiente se define una clase (que es un tipo de referencia) denominada **EjemploRefType**. Crea una instancia de un objeto **EjemploRefType**, asigna 44 a su campo value y pasa el objeto al método **ModificarObjeto**. Fundamentalmente, este ejemplo hace lo mismo que el ejemplo anterior: pasa un argumento por valor a un método. Pero, debido a que se usa un tipo de referencia, el resultado es diferente. La modificación que se lleva a cabo en **ModificarObjeto** para el campo **obj.value** cambia también el campo value del argumento **rt**, en el método **Main** a 33, tal y como muestra el resultado del ejemplo.

```
using System;

public class EjemploRefType
{
        public int value;
}

public class Example
{
        public static void Main()
        {
        var rt = new EjemploRefType();
        rt.value = 44;
        ModificarObjeto(rt);
        Console.WriteLine(rt.value);
        }

        static void ModificarObjeto(EjemploRefType obj)
        {
        obj.value = 33;
        }
}
```

Pasar parámetros por referencia

Pase un parámetro por referencia cuando quiera cambiar el valor de un argumento en un método y reflejar ese cambio cuando el control vuelva al método de llamada. Para pasar un parámetro por referencia, use la palabra clave ref o out. También puede pasar un valor por referencia para evitar la copia, pero impedir modificaciones igualmente usando la palabra clave in.

El ejemplo siguiente es idéntico al anterior, salvo que el valor se pasa por referencia al método **ModificarValor**. Cuando se modifica el valor del parámetro en el método **ModificarValor**, el cambio del valor se refleja cuando el control vuelve al autor de la llamada.

```
using System;

public class Ejemplo
{
        public static void Main()
        {
                int valor = 20;
                Console.WriteLine("En el Main, valor tiene = {0}", valor);
                ModificarValor(ref valor);
                Console.WriteLine("Volviendo al Main, valor tiene = {0}", valor);
        }
        static void ModificarValor(ref int i)
        {
        i = 30;
        Console.WriteLine("En ModificarValor, el valor del parámetro i es = {0}", i);
        return;
        }
}
```