



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

Métodos

Una "función", se describe como un **conjunto de instrucciones** que llevan a cabo una subtarea dentro de un programa más complejo. Un método es una función que pertenece a una clase en particular. Los métodos pueden o no necesitar datos de entrada y pueden o no retornar valores.

La sintaxis general de definición de métodos es la siguiente:

```
<accesibilidad> <tipo_retorno> <nombre_metodo> ( <lista_parámetros> )
{
    <declaración_de_variables_locales> ;
    <instrucciones> ;
    return (<valor_retorno>);
}
```

<accesibilidad> se refiere a la visibilidad que tienen otras partes de un programa para ejecutar el método.

Las posibilidades son public, private, protected, lo cual define la accesibilidad que se tiene a este método desde otras clases.

<tipo_retorno> es el tipo del valor que devolverá la función. **Es posible definir funciones que no devuelven ningún valor** (para por ejemplo definir funciones que solo imprimen valores o ciertos mensajes en pantalla, para los que no es necesario que devuelva valor alguno). Para definirlos o declararlos, se utiliza como tipo de retorno la palabra reservada **void**. A estas funciones se las conoce también como **procedimientos**, porque no devuelven ningún valor.

<nombre_metodo> debe ser sustituido por un identificador que corresponde al nombre con el que se invocará al método posteriormente.

<lista_parámetros> corresponde con una lista de variables denotadas por identificadores asociados a un tipo, cada uno representa un parámetro de entrada. La lista de parámetros puede ser vacía pero **los paréntesis deben aparecer**.

La **<declaración_de_variables_locales>** se refiere a **variables locales** que podrán ser utilizadas únicamente dentro del método.

Las **<instrucciones >** es una lista de sentencias, separadas por punto y coma.

<valor_retorno>, debe ser una expresión (constante, aritmética, lógica del tipo que se especificó como <tipo_retorno>, y establece la expresión que retornará la función una vez que haya terminado.

Ejemplos:

```
private int Suma(int x, int y)
{
    int z;
    z=x+y;
    return z;
}

void Saludo() //Sin retorno
{
    Console.WriteLine("Hola mundo");
}

public void Saludo_personalizado(string nombre)
{
    Console.WriteLine("Hola {0}", nombre);
}
```



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

Llamadas a métodos

Las llamadas a métodos son la forma en que vamos a invocar(ejecutar) un método desde una parte del código.

Para llamar un método basta con poner el nombre del método y enviar sus parámetros (si son necesarios).

Por ejemplo:

```
Suma(6,3);
```

```
Saludo();
```

```
Saludo_personalizado("Gaby");
```

En la programación orientada a objetos si los métodos se encuentran dentro de la misma clase y no son estáticos se pueden llamar directamente:

```
Class clase1
```

```
{
    int metodo1()
    {
        metodo2();
        Return 7;
    }
    void metodo2()
    {
    }
}
```

Si los métodos son estáticos se pueden llamar anteponiendo la clase primero:

```
Class clase1
```

```
{
    int metodo1()
    {
        clase1.metodo2();
        Return 7;
    }
    static void metodo2()
    {
    }
}
```



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

Si la llamada es desde un método estático a un método no estático se deben crear objetos de la clase para poder invocarlos:

```
class Program
{
    static void Main(string[] args)
    {
        Program objeto = new Program();
        objeto.metodo1();
    }
    int metodo1()
    {
        metodo2();
        return 7;
    }
    void metodo2()
    {
        Console.WriteLine("Hola");
    }
}
```

Si la llamada es entre métodos estáticos se hace de manera directa:

```
class Program
{
    static void Main(string[] args)
    {
        Program objeto = new Program();
        objeto.metodo1();
        Console.WriteLine(metodo3());
    }
    int metodo1()
    {
        metodo2();
        return 7;
    }
    void metodo2()
    {
        Program.metodo3();
        Console.WriteLine("Hola");
    }
    static string metodo3()
    {
        return "Hola";
    }
}
```



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

Parámetros de salida out y referencia ref

Los **parámetros de salida** permiten al método almacenar información en una variable determinada como de salida identificada por la palabra reservada **out**. Cuando se invoca al método debe utilizarse igualmente out.

Ejemplo

```
//Llamada al método
int num;
miMetodo(out num);

int miMetodo(out int Val)
{
    Val=100;
    return 1;
}
```

Los **parámetros por referencia** proporcionan al método variables de tipo referencia. El método puede leer el valor contenido en la referencia y también puede modificarlo. Para determinar este tipo de parámetro se utiliza la palabra reservada **ref**. Cuando se invoca al método debe utilizarse igualmente ref.

Los parámetros por referencia deben estar siempre inicializados al invocar al método.

Ejemplo

```
//Llamada al método
int num =80;
miMetodo(ref num);

void miMetodo(ref int Val)
{
    Val=Val+100;
}
```



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

Ejercicios de la Práctica:

Mediante estos ejercicios el alumno aprenderá a utilizar métodos para agrupar bloques de instrucciones así como hacer llamadas a los métodos.

Abre el Visual C# y crea un nuevo proyecto cuyo nombre de solución sea **Practica7Metodos** y tu proyecto **ejemplo_llamadas_metodos**.

1. A continuación escribe el siguiente código (NOTA. No lo copies, escribe línea por línea, el objetivo de este ejercicio es ir observando cuando programas como llamar a los métodos. Se recomienda escribir primero los métodos y después escribir el código del método Main)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ejemplo_llamadas_metodos
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("*****Llamadas desde Main*****");
            metodo_siempre_disponible("enero", "febrero", "marzo", "abril", "mayo");
            Program objeto_interno1 = new Program();
            objeto_interno1.llamadas_internas_metodos();

            otra_clase objeto_otra_clase = new otra_clase("Gabriela");
            objeto_otra_clase.metodo_publico();
            objeto_otra_clase.operando_arreglos(1,2,3,4,5,6,7,8,9);
        }

        static void metodo_siempre_disponible(params string[] args)
        {
            Console.WriteLine("Metodo siempre disponible");
            Console.WriteLine("Los argumentos que recibí son");
            foreach (string s in args)
                Console.WriteLine(s);
        }

        void metodo_interno()
        {
            Console.WriteLine("*****Llamadas desde metodo_interno*****");
            Console.WriteLine("Metodo interno que no recibe ni devuelve valores");
            //Otra forma de utilizar llamadas internas
            Console.WriteLine("La suma de 1+1 es {0}", this.metodo_externo_suma(1, 1));
        }

        public int metodo_externo_suma(int a, int b)
        {
            Console.WriteLine("Metodo publico que recibe y devuelve valores");
            int res;
            res = a + b;
            return res;
        }

        void llamadas_internas_metodos()
        {

```



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

```
Console.WriteLine(" *****Llamadas desde Metodo
llamadas_internas_metodos*****");
metodo_interno();
Console.WriteLine("La suma de 3+5 es {0}",metodo_externo_suma(3, 5));
metodo_siempre_disponible("lunes", "martes", "miercoles");
}
}
class otra_clase
{
    public otra_clase(string nombre)
    {
        Console.WriteLine("Hola " + nombre);
    }
    private void metodo_privado_otra_clase()
    {
        Console.WriteLine("Solo es disponible de manera interna");
    }
    public int[] operando_arreglos(params int[] arr)
    {
        int[] arreglo_temp = new int[arr.Length];
        int i=0;
        foreach(int n in arr)
        {
            arreglo_temp[i] = n * 5;
            i++;
        }
        return arreglo_temp;
    }
    public void metodo_publico()
    {
        Console.WriteLine(" *****Llamadas clase otra_clase.metodo_public*****");
        metodo_privado_otra_clase();
    }
}
}
```

2. Crea un proyecto que contenga los siguientes métodos:
 - a) Menu. Deberá llamar a cada uno de los siguientes métodos:
 - b) Escribir un método que reciba como parámetros un vector v de números enteros comprendidos entre 1 y 10, y un número n. Eliminar del vector v todas las ocurrencias del número n, desplazando hacia la izquierda todos los elementos del vector sin que queden huecos. Los elementos de la derecha del vector se rellenan con 0. Retornar el vector resultado.

Ejemplo:
Vector = 2 5 8 2 9 7 4 3 2 8
N= 2
Resultado: 5 8 9 7 4 3 8 0 0 0

- c) Escribe un método que reciba como parámetros dos cadenas, verifique y retorne verdadero si la 2ª. cadena está incluida dentro de la 1er. Cadena o falso en caso contrario.

Ejemplo:
Cadena1: El dia esta soleado
Cadena2: sol
Resultado: verdadero

Practica7. Métodos



TEMA2. METODOLOGÍA DE LA PROGRAMACIÓN ESTRUCTURADA

- d) Escribir un método que reciba como parámetro un vector de números enteros y posteriormente escriba por pantalla todos los números introducidos indicando además cual es el mayor y el menor.
- e) Escribe un método que reciba como parámetros dos cadenas y un número n. A continuación, tiene que insertar la segunda cadena dentro de la primera a partir de la posición n. Deberá retornar la cadena resultante.

Ejemplo:

Cadena1:Tecnicas de Programacion

Cadena2: UNAM Ingenieria

n = 5

Resultado: TecniUNAM Ingenieriacas de Programacion

- f) Escribe un método que reciba como parámetro un número n y que genere e imprima una matriz unitaria de orden n. Una matriz unitaria de orden n es una matriz de nxn en la que todos los elementos valen 0, excepto los de la diagonal principal que valen 1. La diagonal principal es la que va de la esquina superior izquierda a la inferior derecha.