Facultad: Ingeniería
Escuela: Ingeniería en Computación
Asignatura: Programación I

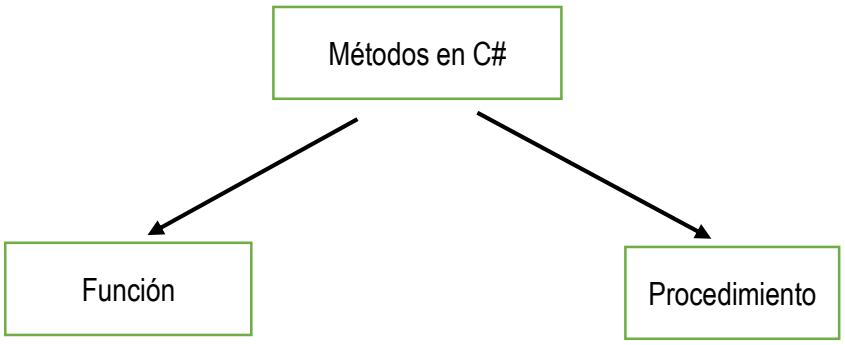
Tema: "Funciones, Procedimientos y Recursividad en C#".

Objetivos

- Utilizar la sintaxis de las funciones definidas por el usuario (programador) para resolver problemas.
- Identificar la diferencia entre una función y un procedimiento.
- Aplicar los conceptos de funciones y procedimientos aplicándolo a la recursividad.
- Hacer uso de recursividad, tomando como base el uso de funciones y procedimientos.

Introducción

Un problema complejo se puede dividir en pequeños subproblemas más sencillos. Estos subproblemas se conocen como ***módulos*** y su implementación en un lenguaje se llama ***subprograma (procedimientos y funciones)***.
Un subprograma realiza las mismas acciones que un programa, sin embargo, vamos a utilizar el subprograma o módulo para una acción u operación específica.
Un subprograma recibe datos de un programa y le devuelve resultados (el programa "llama" o "invoca" al subprograma, éste ejecuta una tarea específica y devuelve el "control" al programa que lo llamó).
En C# a las funciones o procedimientos se le conocen con el nombre de métodos.



Funciones (Retornan un valor)

En el ámbito de la programación, una función es un tipo de subalgoritmo, es el término para describir una secuencia de órdenes que hacen una tarea específica de una aplicación más grande.
Las declaraciones de las funciones, generalmente son especificadas por:

- **Un nombre único en el ámbito.** Nombre de la función con el que se identifica y se distingue de otras. No podrá ser otra función o procedimiento con ese nombre (salvo sobrecarga o polimorfismo en programación orientada a objetos).
- **Un tipo de dato de retorno.** Tipo de dato del valor que la función devolverá al terminar su ejecución.

- **Una lista de parámetros.** Especificación del conjunto de argumentos (pueden ser cero uno o más) que la función debe recibir para realizar su tarea.

Procedimientos (No retornan valor)

Fragmento de código (subprograma) que realiza una tarea específica y es relativamente independiente del resto del código. Los procedimientos suelen utilizarse para reducir la duplicación de códigos en un programa.
Los procedimientos pueden recibir parámetros, pero no necesitan devolver un valor como es el caso de las funciones.

Sintaxis Función.

Modificador_de_acceso Tipo_Devuelto Nombre_Función(tipo(s)_argunmento(s) **nombres**)

```
{
      //declaración de datos y cuerpo de la función.
      return (valor)
}
```

Sintaxis Procedimiento

Modificador_de_acceso void Nombre_procedimiento(tipo(s)_argunmento(s) **nombres**)

```
{
      //declaración de datos y cuerpo de la función.
}
```

Recursividad

Una función **recursiva** es una función que se llama a sí misma directa o indirectamente. La **recursividad o recursividad directa** es el proceso por el que una función se llama a sí misma desde el propio cuerpo de la función.
La recursividad indirecta implica más de una función.
La recursividad indirecta implica, por ejemplo, la existencia de dos funciones: uno() y dos(). Suponga que la función principal llama a uno() –una segunda llamada a uno()-. Esta acción es recursión indirecta, pero es recursiva, ya que uno() ha sido llamado dos veces, sin retornar ninguna su llamada.

Un proceso recursivo debe tener una condición de terminación, ya que de lo contrario va a continuar indefinidamente.
Un algoritmo típico que conduce a una implementación recursiva es el cálculo del factorial de un número.
El factorial de n (n!).
n! = n*(n-1)*(n-2)*…*3*2*1

Material y Equipo

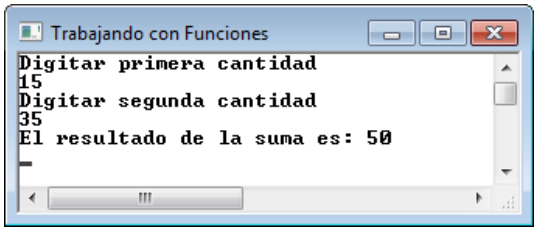
- Guía de laboratorio No. 6.
- Computadora con Visual Studio 2013 o superior.
- Dispositivo de almacenamiento (USB).

Procedimiento

G6_Ejemplo1: En nuestro primer ejemplo, vamos a implementar un **procedimiento** que sea capaz de realizar una simple suma. Esta sencilla implementación es con el objetivo de conocer la sintaxis y la lógica de los procedimientos.

```
1   namespace G6_Ejemplo1
2   {
3       class Program
4       {
5         static void Main(string[] args)
6         {
7           Console.Title = "Trabajando con Funciones";
8           // llamando a la funcion suma
9           suma();
10          Console.ReadKey();
11        }
12        static void suma()
13        {
14          Double r, n1, n2;
15          Console.WriteLine("Digitar primera cantidad");
16          n1 = Double.Parse(Console.ReadLine());
17          Console.WriteLine("Digitar segunda cantidad");
18          n2 = Double.Parse(Console.ReadLine());
19          r = n1 + n2;
20          Console.WriteLine("El resultado de la suma es: " + r);
21        }
22    }
23  }
```

Corrida del programa

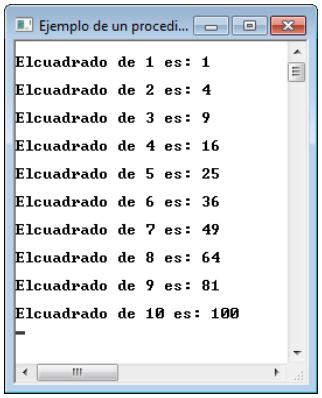


G6_Ejemplo2: Vamos a calcular el cuadrado de los números del 1 al 10 utilizando funciones:

```
1    namespace G6_Ejemplo2
2    {
3        class Program
4        {
5         static void Main(string[] args)
6         {
7          Console.Title = "Ejemplo de una funcion";
8          Double x;
9          // Llamada a la funcion
10         for (x = 1; x <= 10; x = x + 1)
11         {
12         Console.WriteLine("\nElcuadrado de " + x + " es: " + Potencia(x));
13         }
14         Console.ReadKey();
```

```
15        }
16          // Prototipo de la funcion
17          static Double Potencia(double x)
18          {
19           return x * x;
20          }
21      }
22    }
```

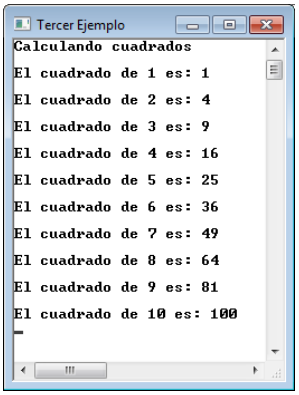
Corrida del programa



G6_Ejemplo3: Ahora, el mismo ejemplo anterior con la diferencia de que vamos a utilizar un **procedimiento** para hacer el cálculo de los cuadrados y además utilizaremos una función predeterminada para el mimo.

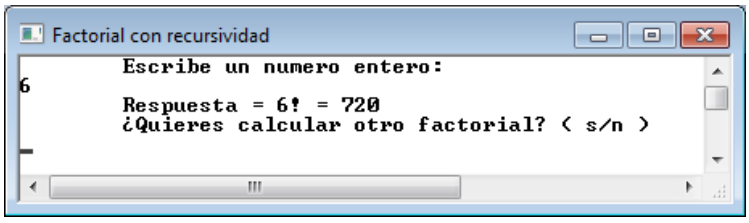
```
1    static void Main(string[] args)
2      {
3       Console.Title = "Tercer Ejemplo";
4       //Ahora desde aquí Llamamos o invocamos el procedimiento así:
5       Potencia();
6       Console.ReadKey();
7      }
8       static void Potencia()
9       {
10       int i;
11       Console.WriteLine("Calculando cuadrados");
12       for (i = 1; i <= 10; i = i + 1)
13       {
14        Console.WriteLine("\nEl cuadrado de " + i + " es: " + Math.Pow(i, 2));
15       }
16      }
```

Corrida del programa



G6_Ejemplo4: Bueno compañeros, en este ejemplo vamos a aplicar la **recursividad** con una de las aplicaciones más conocidas: el factorial de un número. Caben mencionar, que por definición conocemos lo siguiente: 0! = 1 y 1! = 0; por tal motivo tenemos estas dos soluciones triviales que significa que si es ingresado un 0 ó 1 el resultado será 1. Veamos el ejemplo:

```
1   class Program
2       {
3        static void Main(string[] args)
4        {
5         int x;
6         String Respuesta;
7         Console.Title = "Factorial con recursividad";
8         do
9         {
10        Console.Clear();
11        Console.WriteLine("\tEscribe un numero entero: ");
12        x = int.Parse(Console.ReadLine());
13        Console.WriteLine("\tRespuesta = " + x + "! = " + Factorial(x));
14        Console.WriteLine("\t¿Quieres calcular otro factorial? ( s/n )");
15        Respuesta = Console.ReadLine();
16        } while (Respuesta == "S" || Respuesta == "s");
17        Console.ReadKey();
18      }
19       static int Factorial(int x)
20        {
21          if (x == 0 || x == 1)
22          {
23          return 1;
24          }
25          else
26          {
27          return x * Factorial(x - 1);
28          }
29        }
30    }
```



Análisis de Resultados

1. Crear un programa que solicite el año de nacimiento de una persona y retorne la edad haciendo uso de funciones o procedimientos.
2. Cree un programa que contenga el siguiente menú:
   a) Dividir.
   b) Obtener cubo.
   c) Cálculo de IMC (Índice de Masa Corporal).
   d) Salir.

Consideraciones:

- El menú debe permanecer disponible hasta que el usuario elija la opción d.
- Utilizar una función o procedimiento para cada opción.
- Para la opción d, utilice la fórmula: IMC = Peso[Kg] / Altura^2[Metros].

3. Desarrollar un programa que implemente una función para convertir coordenadas polares a rectangulares. Debe tener en cuenta lo siguiente:
   $x = r\cos(\theta)$; $y = r\text{sen}(\theta)$

Investigación Complementaria

1. Escribir un programa que lea dos números desde teclado (x y n) e implemente una función para calcular la siguiente progresión:

   $$1 + X + X^2 + X^3 + \ldots + X^n$$

2. Aplicando recursividad: investigar en qué consiste la serie **Fibonacci** y desarrollar una aplicación para determinar el resultado de la misma. El usuario debe ingresar desde teclado el límite de la serie.
3. Desarrolle un programa en C# haciendo uso de un procedimiento para determinar el resultado de la siguiente serie:

   $$\frac{1}{2} - \frac{2}{2^2} + \frac{3}{2^3} - \cdots + \frac{n}{2^n}$$

Bibliografía

- Deitel, Harvey M. y Paul J. Deitel, Cómo Programar en C#, Segunda Edición, México, 2007.