PROGRAMACIÓN ORIENTADA A OBJETOS
2o. PARCIAL

Practica #3
Funciones, Procedimientos y Recursividad en C#

Daniel Vázquez de la Rosa
d_vazquez@outlook.com

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Septiembre 2018

Datos Personales

Crear un Documento en Word que incluya una caratula que contenga tus datos. Dicho documento será guardado como: Grupo_Apellido Paterno_NombreAlumno_Practica3_POO y deberá contener toda la información de la practica creada.

Objetivos de la Práctica

- 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.

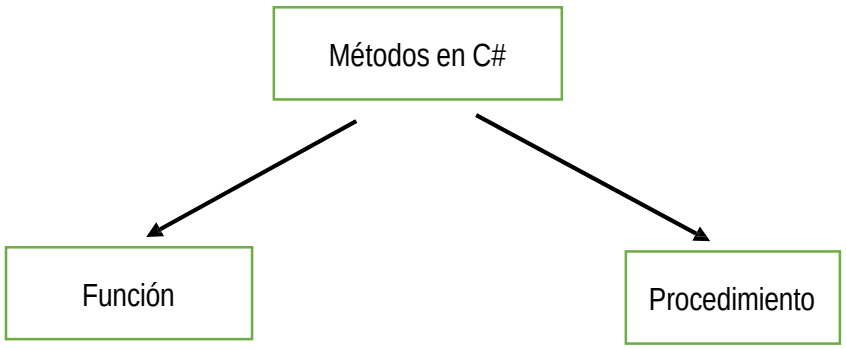
Introducción

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!).

$$n! = n*(n-1)*(n-2)*\ldots*3*2*1$$

Desarrollo

Ejemplo 1

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.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace G6_Ejemplo_1
{
    class Program
    {
        static void Suma()
        {
            Double r;
            Console.WriteLine("Primera Cantidad: ");
            Double n1 = Double.Parse(Console.ReadLine());
            Console.WriteLine("Segunda Cantidad: ");
            Double n2 = Double.Parse(Console.ReadLine());
            r = n1 + n2;
            Console.WriteLine("Resultado: "+r);
        }
        static void Main(string[] args)
        {
            Console.Title = "TRABANADO CON FUNCIONES";
            //Llamada a la función
            Suma();
            Console.ReadKey();
        }
    }
}
```

Ejemplo 2

Se desea calcular el cuadrado de los números del 1 al 10 utilizando funciones

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace G6_Ejemplo_2
{
    class Program
    {
        //Prototipo de la fución
        static Double Potencia(Double x)
        {
            return x * x;
        }
        static void Main(string[] args)
        {
            Console.Title = "FUNCIONES";
            Double x;
            //Llamada a la función
            for(x=1; x<=10; x++)
            {
                Console.WriteLine("\nEl cuadrado de "+x+" es: "+Potencia(x));
            }
            Console.ReadKey();
        }
    }
}
```

Ejemplo 3

Ahora observe que se desarrollara 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 mismo.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace G6_Ejemplo_3
{
    class Program
    {
        static void Potencia()
        {
            Console.WriteLine("Calculando cuadrados:");
            for(int i=1; i<=10; i++)
            {
                Console.WriteLine("\nEl cuadrado de " + i + " es: " + Math.Pow(i,2));
            }
        }
        static void Main(string[] args)
        {
            //Ahora desde aquí únicamente hacemos los siguiente:
            //Llamamos o invocamos el procedimiento, así:
            Potencia();
            Console.ReadKey();
        }
    }
}
```

Ejemplo 4

Desarrollar un programa que aplique la recursividad con un factorial de un número. Cabe 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.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Factorial_Recursividad
{
    class Program
    {
        static int Factorial(int x)
        {
            if(x == 0 || x == 1 )
            {
                return 1;
            }
            else
            {
                return x * Factorial(x - 1);
            }
        }
```

```
22          static void Main(string[] args)
23          {
24              int x;
25              String Respuesta;
26              Console.Title = "Factorial con recursividad";
27              do {
28                  Console.Clear();
29                  Console.WriteLine("\tDame un número: ");
30                  x = int.Parse(Console.ReadLine());
31                  Console.WriteLine("\tRespuesta = " + x + "! = " + Factorial(x));
32                  Console.WriteLine("\t¿Quieres calcular otro factorial? (S / N)");
33                  Respuesta = Console.ReadLine();
34              } while (Respuesta == "S" || Respuesta == "s");
35              Console.ReadKey();
36          }
37      }
38  }
```

Responda

Una vez que has hecho estos ejemplos, deberás de contestar los siguientes ejercicios

1. 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].
2. Desarrollar un programa que implemente una función para convertir coordenadas polares a rectangulares. Debe tener en cuenta lo siguiente:
   $x = r\cos(\theta)$; $y = r\text{sen}(\theta)$
3. Desarrolle un programa que permita aplicar 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.

Esta Práctica se deberá de entregar a más tardar el día jueves 27/09/2018 antes de las 10:00 PM, toda práctica que no sea enviada dentro de este horario será evaluada con cero.

Al Enviar la Practica, deberás poner en el asunto Practica3_Grupo_ApellidoPaterno_ApellidoMaterno_Nombre

Bibliografía

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