2023

Programación II

Procedimientos y Funciones

PROCEDIMIENTOS Y FUNCIONES

Un **módulo** es un bloque de código creado para llevar a cabo una determinada tarea. La utilización de módulos dentro del programa permitirá, como su nombre lo indica, llegar al concepto de modularidad. En todo algoritmo se busca alcanzar la modularidad, base para lograr la tan deseada modificabilidad. Un módulo correctamente desarrollado no necesita tener conocimiento sobre la lógica implementada dentro de los módulos que invoca (concepto de caja negra). Por lo que la solución total (algoritmo principal) quedará compuesta por partes (módulos) independientes, permitiendo prácticamente que cualquier modificación requerida, se logre sin retocar nada más que el módulo involucrado.

Procedimientos: Son aquellos módulos que no devuelven un valor, en su declaración se utiliza la palabra clave **void**. Un procedimiento es utilizado para llevar a cabo una tarea determinada para la cual se considera que el módulo invocante no requerirá información de la operación a realizar. Un ejemplo práctico sería un procedimiento dedicado al blanqueo de pantalla.

Funciones: Son aquellos módulos que devuelven un valor, es decir en la declaración de la función se especifica el tipo de dato (int, char, float, etc) que la misma retornará al módulo invocante. El valor retornado por la función será especificado dentro de la misma a través de la palabra clave **return** seguido por la expresión a retornar, y podrá ser utilizado por el módulo invocante para alcanzar su objetivo. El valor retornado por la función será un valor único, inclusive podría llegar a ser una estructura, pero nunca un arreglo, en todo caso un puntero al mismo (concepto que se detallará más adelante). Un ejemplo práctico es una función dedicada a la obtención del valor absoluto de un número dado.

Procedimiento	

Declaración	void NombreProcedimiento (TipoParam1 NombreParam1, TipoParam2 NombreParam2,….)
	{
	declaración variables;
	código
	}

Procedimiento que presenta un mensaje por pantalla cuyo destinatario debe ser especificado.	

Declaración	static void Saludar(string Usuario)
	{
	Console.WriteLine("Hola {0}.", Usuario);
	}

Utilización	Saludar("Pedro"); Luego de la ejecución de esta línea, se presentará el siguiente mensaje: Hola Pedro.

Función	

Declaración	TipoDatoRetornar NombreFuncion (TipoParam1 NombreParam1, TipoParam2 NombreParam2,….)
	{
	declaración variables;
	código
	return (expresión);
	}

Función que retorna el Valor Absoluto de un número entero.	

Declaración	static int ValorAbsoluto(int Numero)
	{
	if (Numero < 0)
	{
	return (-Numero);
	}
	else
	{
	return (Numero);
	}
	}

Utilización	ValorAbs = ValorAbsoluto(-3); Luego de la ejecución de esta línea, la variable ValorAbs contendrá el valor 3.

PARÁMETROS

Un **parámetro o argumento**, es el elemento por medio del cual se transmiten datos de un módulo a otro. Los parámetros pueden ser clasificados como:

Por Valor: también conocido como parámetro de solo entrada. Es un parámetro que, en el momento de la invocación del módulo (función/procedimiento), el valor por él recibido es copiado al mismo y por lo tanto no se tiene referencia de la variable utilizada en ese instante. En consecuencia, cualquier modificación en el contenido del parámetro no se verá reflejada en la variable utilizada al momento de la invocación.

Por Referencia: es un parámetro que, en el momento de la invocación del módulo (función/procedimiento), establece una referencia con la variable utilizada en ese instante. En consecuencia, cualquier modificación en el contenido del parámetro se verá reflejada en la variable utilizada al momento de la invocación. Se debe ser cuidadoso al momento de utilizar este tipo de parámetros ya que cualquier modificación en el contenido de éstos afectara al módulo invocante.
Los parámetros por referencia pueden ser a su vez clasificados como parámetros de:

a. **Entrada/Salida:** Para identificar un parámetro como parámetro de Entrada/Salida, éste deberá estar precedido por la sigla "**ref**" en la declaración e invocación del módulo. Los parámetros tipo arreglo son tratados como parámetros de Entrada/Salida.
b. **Salida:** Para identificar un parámetro como parámetro de Salida, éste deberá estar precedido por la sigla "**out**" en la declaración e invocación del módulo.

Procedimiento que intercambia el contenido de dos parámetros pasados por referencia.	

Declaración	static void Intercambiar (ref int Param1, ref int Param2)
	{
	int aux;
	aux = Param1;
	Param1 = Param2;
	Param2 = aux;
	}

Utilización	int Valor1, Valor2; Valor1 = 10; Valor2 = 20; Intercambiar(ref Valor1, ref Valor2); Luego de la ejecución de esta última línea, la variable Valor1 contendrá el valor 20, y Valor2 contendrá el valor 10.

FUNCIONES RECURSIVAS

Un tipo especial de funciones son las llamadas ***funciones recursivas***. Estas se caracterizan por, como su nombre lo indica, invocarse así mismas. Son funciones poco frecuentes y utilizadas en situaciones, algoritmos muy particulares. Se debe tener en cuenta que este tipo de funciones, por su propia naturaleza, deben contar con una condición de finalización ya que de lo contrario el módulo se invocaría por tiempo indefinido, originando un loop infinito. El hilo de ejecución en cada invocación de la función no va a continuar dentro de la misma hasta que finalice la ejecución de la función invocada. Este proceso continuará hasta que se alcance la condición de finalización. Una vez alcanzado este punto de corte, el hilo de ejecución ira retornando de la función invocada a la invocante hasta llegar a la primera invocación. Obteniendo así el resultado buscado.

Función que calcula el factorial de un numero dado, en forma recursiva.	

Declaración	static uint factorial(uint num)
	{
	if (num>1)
	{
	return num * factorial(num-1);
	}
	else
	{
	return 1;
	}
	}
	
	//tener en cuenta que los números negativos no tienen factorial

Utilización	resultado = factorial(4); Luego de la ejecución de esta línea, la variable resultado contendrá el valor 24.

Funcionamiento	*Llamada a la función factorial (4) desde main:* return 4 * factorial (3); **esta expresión será resuelta solo cuando se conozca factorial(3)** *Llamada a la función factorial (3) desde factorial(4):* return 3 * factorial (2); **esta expresión será resuelta solo cuando se conozca factorial(2)** *Llamada a la función factorial (2) desde factorial(3):* return 2 * factorial (1); **esta expresión será resuelta solo cuando se conozca factorial(1)** *Llamada a la función factorial (1) desde factorial(2):* return 1;

MAIN CON PARÁMETROS

El resultado de toda compilación exitosa es un archivo con el mismo nombre que el programa fuente, pero con extensión .exe (NombrePrograma.exe). De esta forma es posible ejecutar el programa desarrollado directamente desde la línea de comandos, con el solo hecho de copiar el nuevo archivo a un pc el usuario de la misma está en condiciones de utilizarlo, sin necesidad de contar con un compilador.
Main, como su nombre lo indica, es el módulo principal del programa y como todo procedimiento/función puede aceptar parámetros que en su caso provienen de la línea de comandos. De esta manera el usuario podrá especificarle al archivo .exe ciertos elementos que deberá tener en cuenta a lo largo de su ejecución.
Para permitir que estas especificaciones brindadas por el usuario puedan ser interpretadas por el programa se deberá incorporar un arreglo denominado **args** de una dimensión de tipo string como parámetro en la definición del Main. Este parámetro se cargará automáticamente al iniciarse la ejecución del programa con los valores correspondientes.

Ejemplo

```
//De cada argumento se listara su posición y el texto tipeado por el usuario  static
void Main(string[] args)
{    int
i;
   for (i = 0; i < args.Length; i++)
   {
      Console.WriteLine("El argumento {0} es: {1}", i, args[i]);
   }
   Console.ReadKey();
}
```

Cabe destacar que C# cuenta con la posibilidad de enviar argumentos al main que serán tenidos en cuenta durante la depuración. Para ello se seguirán los siguientes pasos: Proyecto > Propiedades de … > Depurar, en la ficha presentada se visualizará un cuadro de texto descripto como “Argumentos de la línea de comandos:” en el cual se tipeará cada uno de los argumentos que se desean enviar separados por un espacio.