Intro. a la Programación con C# .NET

Contenido del módulo

- Repaso instrucción while
- Instrucción do..while
- Instrucción for
- Introducción a arreglos
- Ejercitación

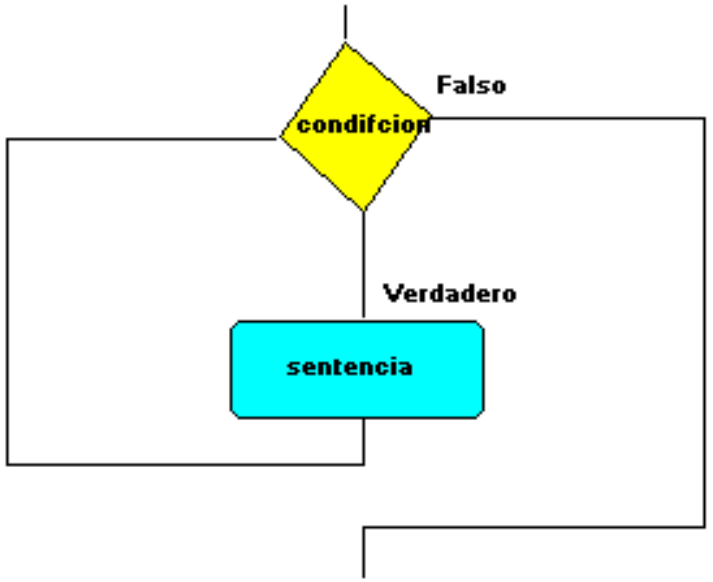
Intro. a la Programación con C# .NET

REPASO

Instrucción While

La instrucción while permite ejecutar un bloque de instrucciones mientras se de una cierta condiciòn. Su sintaxis de uso es:

```
while (<condición>)  {
        <instrucciones>
    }
```



Intro. a la Programación con C# .NET

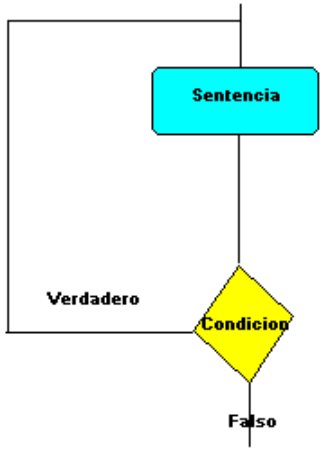
Instrucción do...while

La instrucción do...while es una variante del while que se usa así:

Do {

<instrucciones>

} while(<condición>);



La diferencia entre el **do...while** y el **while** es que **do...while** primero ejecuta las **<instrucciones>** y luego evalúa la **<condición>** para ver si se repite su ejecución, es decir que con **do…while** siempre se ejecutan las instrucciones al menos una vez aún cuando la condición sea falsa desde el principio.

Intro. a la Programación con C# .NET

Ejemplo con do...while

```
// Estructura DO While
// esta estructura asegura por lo menos una pasada
int a = 1;
do
{

    Console.WriteLine(a);
    a++;
} while (a <= 10);
Console.WriteLine("-- fin estructura do while --");
Console.WriteLine(a);
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Ejemplo con while anidado

```
// Estructura While Anidada
int externo = 1;
int interno = 1;
while (externo <= 4)
{
    while (interno <= 4)
    {
      Console.WriteLine("Externo:" + externo + " Interno:" + interno);
      interno++;
    }
    interno = 1;
    Console.WriteLine("");
    externo++;
}
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Instrucción for

La instrucción for realiza un ciclo una cantidad de veces predefinida.

Consta de tres partes:

- la sentencia inicialización que se ejecuta al comienzo del ciclo
- la condición que se evalúa al entrar al ciclo y luego al comienzo de cada iteración
- la modificación se ejecuta al final de cada iteración, y previene un bucle infinito (se ejecuta al final de cada iteración).

```
for (<inicialización>; <condición>; <modificación>)  {
        <instrucciones>
 }
```

Intro. a la Programación con C# .NET

Ejemplo con for

```
// Estructura FOR
for (int a = 1; a <= 10; a++)
{
    Console.WriteLine(a);
}
Console.WriteLine("---fin de la estructura FOR---");
Console.ReadKey();
```

Luego de ejecutar agregar antes del Console.ReadKey() la siguiente instrucción:

Console.WriteLine(a);

Conclusión: la variable a es local al for.

Intro. a la Programación con C# .NET

Ejemplo con for

NO Recomendado!

```
for (int a = 1; a <= 10; a++)
    Console.WriteLine(a);
Console.WriteLine("---fin de la estructura FOR---");
Console.ReadKey();

//si solo hay una linea en el ciclo puedo omitir las llaves
//NO RECOMENDADO
```

Intro. a la Programación con C# .NET

Ejemplo con for anidado

```
// FOR anidado
for (int externo = 1; externo <= 4; externo++)
{
  for (int interno = 1; interno <= 5; interno++)
  {
    Console.WriteLine("Externo:" + externo + " Interno:" + interno);
  }
  Console.WriteLine("");
}
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Ejemplo con for y varias instrucciones separadas por comas

```
// For con varias instrucciones en encabezado
for (int a = 1, b=10; a <= b; a++,b--,Console.WriteLine("---"))
{
    Console.WriteLine("a tiene:" + a + " b tiene:" + b);
}
Console.ReadKey();
```

Hacer el seguimiento de las variables!

Intro. a la Programación con C# .NET

Ejercitación con for

Lab 3 Ejercicio 2: Imprimir los números del 1 al 10 salteando de a dos uno abajo del otro.

Lab 3 Ejercicio 3: Imprimir los números del 10 al 1 uno abajo del otro.

Lab 3 Ejercicio 4: Imprimir la suma de los números impares del 1 al 10.

Intro. a la Programación con C# .NET

Ejercitación con for y caracteres

Lab 3 Ejercicio 8: Imprimir la siguiente figura (concatenar):

@

@@

@@@

@@@@

@@@@@

Intro. a la Programación con C# .NET

Ejercitación con while

Lab 3 Bonus-While Ejercicio 2: Implementar un código que imprima el mayor y el menor de una serie de números que vamos introduciendo por teclado. El ingreso termina cuando el usuario ingresa el número 99.

Antes de hacer el ejercicio, pensar cuántas variables necesito y cómo debo inicializarlas!

Intro. a la Programación con C# .NET

Ejercitación con while

Solución

```
int numero;
int mayor=0;
int menor=100;

Console.Write("Ingrese un numero (99=FIN):");
numero=int.Parse(Console.ReadLine());
while (numero!=99)
{
  if (numero>mayor) mayor=numero;
  if (numero<menor) menor=numero;

  Console.Write("Ingrese un numero (99=FIN):");
  numero=int.Parse(Console.ReadLine());
}
Console.WriteLine("maximo valor: "+ mayor);
Console.WriteLine("minimo valor: "+ menor);
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Arreglos

Un arreglo es una variable que permite almacenar varios valores del mismo tipo.

Arreglos de una dimensión

Un **array** es una variable que puede almacenar varios valores simultáneamente. Cada uno de estos valores se identifica por su posición al cual se llama índice. Así, para acceder al primer elemento del array habría que usar el índice cero, para el segundo el índice uno, para el tercero el índice dos, y así sucesivamente. Se lo declara de la siguiente forma :

```
tipo[] variable;
```

Intro. a la Programación con C# .NET

En C# los arrays son objetos derivados de la clase **System.Array**.

Cuando declaramos un arreglo en C# este aún no se habrá creado, en consecuencia, antes de usarlos habrá que instanciarlo, como si fuera cualquier otro objeto.

```
string[] nombres; // Declaración del array
nombres = new string[3]; // Instanciación del array
```

El arreglo **nombres** será utilizable únicamente a partir de su instanciación, y podrá tener un máximo de tres elementos (0, 1 y 2).

El índice del array comienza de CERO (0) y el índice del último elemento es el total de elementos menos uno (3-1=2)

Intro. a la Programación con C# .NET

Carga de Arreglos

En el ejemplo se declaran e instancian dos arreglos, uno con strings y otro con números (luego completar con la siguiente diapositiva):

```
//Arreglos
String[] nombres = new String[4];
int[] numeros = new int[4];
nombres[0] = "Carlos";
numeros[0] = 1;
nombres[1] = "Juan";
numeros[1] = 2;
nombres[2] = "Jose";
numeros[2] = 3;
nombres[3] = "Javier";
numeros[3] = 4;
```

Intro. a la Programación con C# .NET

Uso de Arreglos

Ingresar el el nro del índice del arreglo y mostrar su contenido…(Continúa de la diapositiva anterior):

```
//Uso de arreglos, continùa del ejemplo anterior
int i;
Console.WriteLine("Ingrese el ìndice desde 0 a 3");
i = Convert.ToInt32(Console.ReadLine());
if (i >= 0 && i <= 3)
{
    Console.WriteLine (nombres[i]);
    Console.WriteLine (numeros[i]);
}
else
{
    Console.WriteLine("Indice incorrecto");
}
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Muestra de un elemento y recorrido

En el ejemplo vemos como acceder a un determinado elemento y aparte cómo recorrerlo:

```
// Salida de una posición en especial
Console.WriteLine("Accedo a " + numeros[2] + "   " + nombres[2]);
Console.WriteLine();

// Recorrido del vector
for (int a = 0; a < 4; a++)
{
    Console.WriteLine("Recorro " + numeros[a] + "   " + nombres[a]);
}
Console.ReadKey();
```

Intro. a la Programación con C# .NET

Recorrido usando la propiedad Lenght

```
// El metodo length devuelve la longitud del arreglo
Console.WriteLine("Longitud de nombres es : " + nombres.Length);

// Recorrido mejorado del arreglo
for (int a = 0; a < nombres.Length; a++)
{
    Console.WriteLine(numeros[a] + "  " + nombres[a]);
}
```

Intro. a la Programación con C# .NET

Ejercitación

Crear un arreglo y cargarlo con números enteros.

Mostrar la suma de los números del arreglo.

Mostrar el promedio de los números del arreglo.

Intro. a la Programación con C# .NET

Solución

```
//Arreglos Ejercicio con suma y promedio
int[] numeros = new int[4];
numeros[0] = 1;
numeros[1] = 2;
numeros[2] = 3;
numeros[3] = 4;
int suma = 0;
for (int a = 0; a < numeros.Length; a++)
{
    suma = suma + numeros[a];
}
Console.WriteLine("Suma es {0} y promedio {1} es ",suma, suma/numeros.Length);
Console.ReadKey();
```