PROGRAMACIÓN EN C# .NET

Módulo 3

Instrucciones fundamentales

Ing. Bruno López Takeyas

Instituto Tecnológico de Nuevo Laredo

MÉTODOS

Método	Descripción	Ejemplo
Abs(x)	Valor absoluto	Abs(-23.7) es 23.7
Ceiling(x)	Redondeo	Ceiling(9.2) es 10.0
Cos(x)	Coseno (x en radianes)	Cos(0.0) es 1.0
Exp(x)	Método exponencial e^x	Exp(1.0) es 2.7182818284509451
Floor(x)	Redondea al valor inmediato menor	Floor(9.2) es 9.0
Log(x)	Logaritmo natural (base e)	Log(2.7182818284509451) es 1.0
Max(x, y)	Valor mas grande de x e y	Max(2.3, 12.7) es 12.7
Min(x, y)	Valor mas pequeño de x e y	Min(2.3, 12.7) es 2.3
Pow(x, y)	x^y	Pow(2.0, 3.0) es 8.0
Sin(x)	Seno (x en radianes)	Sin(0.0) es 0.0
Sqrt(x)	Raíz cuadrada	Sqrt(9.0) es 3.0
Tan(x)	Tangente (x en radianes)	Tan(0.0) es 0.0

USO DE MÉTODOS

- Para calcular la raíz cuadrada de x…

```
double resultado = Math.Sqrt(x);
```

- Si se olvida invocar el método de la clase Math sin colocar el nombre de la clase antes del punto, puede provocar un error

```
double area = Math.PI*Math.Pow(radio,2.0);
```


GENERAR NÚMEROS ALEATORIOS

- Utilizar la clase **`Random`** (ubicada en el namespace **`System`**)

```
Random Numero = new Random();
int x = Numero.Next();
```

- El método **`Next( )`** genera un número aleatorio entre 0 y la constante **`Int32.MaxValue`** (2, 147, 483, 647)

GENERAR NÚMEROS ALEATORIOS

- Para generar números entre 0 y 5

```
Random Numero = new Random();
int valor = Numero.Next(6);
```

- Para generar números entre 1 y 6

```
Random Numero = new Random();
int valor = Numero.Next(1,7);
```


OPERADORES DE BITS

Operador	Descripción	Ejemplo
>>	Desplazamiento a la derecha	x = x>>1;
<<	Desplazamiento a la izquierda	y = y<<2;
&	AND (bits)	w = w & 3;
|	OR (bits)	q = q | 4;
^	XOR (OR Exclusivo) (bits)	r = r ^ 3;

OPERADORES DE ASIGNACIÓN

Operador	Descripción	Equivalencia	Ejemplo
+=	Suma	x=x+2;	x+=2;
-=	Resta	y=y-3;	y-=3;
*=	Multiplicación	w=w*4;	w*=4;
/=	División	a=a/7.3;	a/=7.3;
%=	Residuo	b=b%7;	b%=7;
<<=	Desplazamiento izquierda (bits)	r=r<<1;	r<<=1;
>>=	Desplazamiento derecha (bits)	t=t>>2;	t>>=2;
&=	And (bits)	q=q&3;	q&=3;
^=	Or Exlusivo XOR (bits)	e=e^4;	e^=4;
|=	Or (bits)	f=f|3;	f|=3;

OPERADORES RELACIONALES

Operador	Descripción	Ejemplo
<	Menor que	X < 3
>	Mayor que	Y > 8.2
<=	Menor o igual que	W <= 5.7
>=	Mayor o igual que	Q >= ‘A’
==	Igual (idéntico)	S == 9
!=	Diferente	T != ‘@’

OPERADORES LÓGICOS

Operador	Descripción	Ejemplo
&&	AND	X<3 && Y>8
||	OR	CALIF<0 || CALIF>100
!	NOT	!(G>7)

INSTRUCCIONES CONDICIONALES

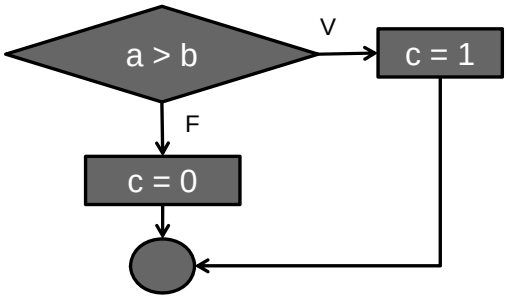
- Simple

```
if(x<3)
   Console.Write("Mayor");
else
   Console.write("No es mayor");
```

- Múltiple

```
switch(opcion)
{
   case 1 : Console.Write("Uno");             break;
   case 2 : Console.Write("Dos");             break;
   default : Console.Write("Fuera de rango"); break;
}
```

OPERADOR CONDICIONAL ?:




```
int a=3, b=2, c;
c = (a>b) ? 1 : 0;
```


CICLOS

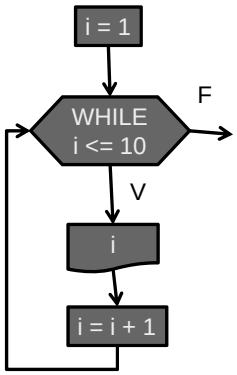
- while
- do – while
- for
- foreach

- Todos los ciclos son de tipo “mientras”; es decir, iteran mientras la condición es verdadera

COMPONENTES DE LOS CICLOS

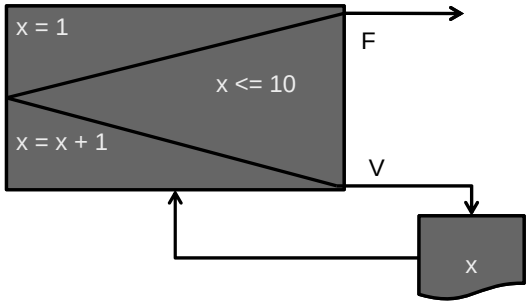
1. Inicialización
2. Condición
3. Incremento / Decremento
4. Cuerpo del ciclo

CICLO WHILE



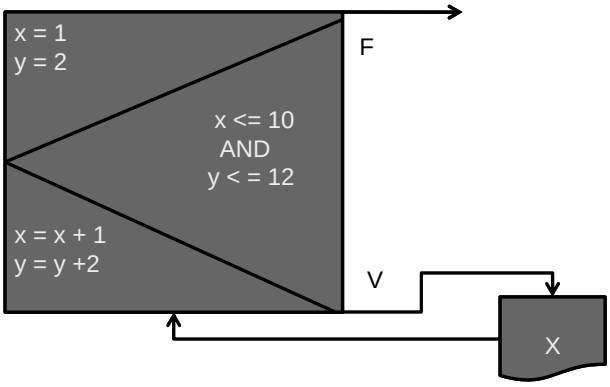

```
i=1;
while(i<=10)
{
   Console.Write("\n i = {0}", i);
   i++;
}
```

CICLO FOR



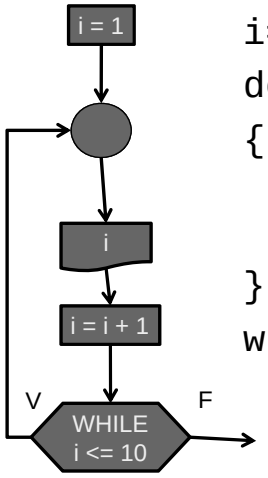

```
for(x = 1; x<=10; x++)
{
   Console.Write("\n x = {0}", x);
}
```


CICLO FOR (EJEMPLO)




```
for(x = 1, y = 2; x<=10 && y<=12; x++, y+=2)
{
   Console.Write("\n x = {0}", x);
}
```

CICLO DO - WHILE




```
i=1;
do
{
   Console.Write("\n i = {0}", i);
   i++;
}
while(i<=10);
```


CICLO FOREACH

- Este ciclo es especialmente útil cuando se desea recorrer todos los elementos de una lista, matriz o colección

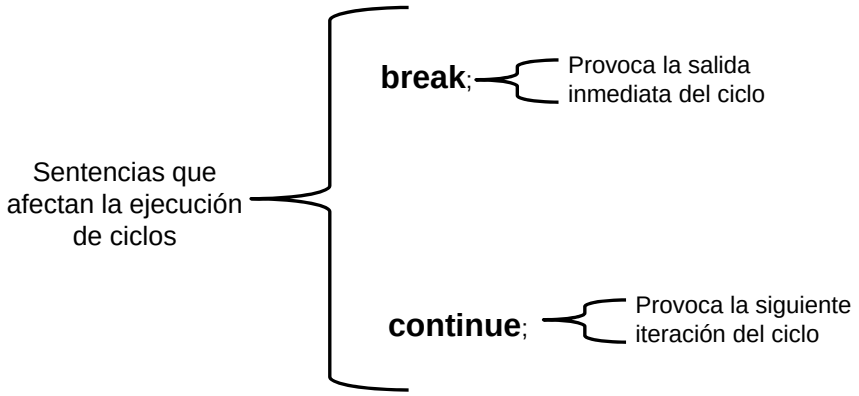
```
foreach(string nombre in Personas)
{
   Console.Write("\n nombre = {0}", nombre);
}
```

CICLO FOREACH

```
string[] Alumno= new string[]{"Pepe",
“Rodolfo", "Maria", “Fabiola",
"Miguel"};

foreach(string nombre in Alumno)
{
  Console.WriteLine("\n{0}",nombre);
}
```


BREAK Y CONTINUE



DIFERENCIAS DE LOS CICLOS

- *do – while*. Se recomienda cuando se desea ejecutar el cuerpo al menos 1 vez. Es útil para las validaciones en la captura de datos.
- *while*. Se recomienda cuando se desea ejecutar el cuerpo del ciclo 0 (cero) o más veces.
- *for*. Es recomendable cuando se conoce la cantidad de iteraciones que ejecutará el ciclo.

ÁMBITO DE LAS VARIABLES

- El ámbito de una variable define dónde puede usarse una variable
- Una variable local declarada en un bloque de programa, sólamente puede ser usada en ese bloque
- El ámbito de una variable también aplica a los métodos y a los ciclos for

ÁMBITO DE LAS VARIABLES

```
for(int x = 1; x<=10; x++)
{
   Console.Write(x);
}

Console.Write(x);
```