



Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

1.- Introducción

Docente: Michell Jáuregui Valdés

Lógica Computacional

La lógica computacional es la misma lógica matemática aplicada al contexto de las ciencias de la computación. Su uso es fundamental en varios niveles: en los circuitos computacionales, en la programación lógica y en el análisis y optimización (de recursos temporales y espaciales) de algoritmos.

Hoy en día, la lógica es extensamente aplicada en los campos de inteligencia artificial y de ciencias de computación, y estos campos proporcionan una rica fuente de problemas en la lógica formal e informal. La teoría de la argumentación es un buen ejemplo de cómo la lógica está siendo aplicada a la inteligencia artificial.

https://es.wikipedia.org/wiki/L%C3%B3gica_computacional

Lógica Computacional

Está muy arraigado estudiar Pseudolenguaje antes de aprender un Lenguaje formal.

Esto tiene varios Pros y Contras, al comprender la mecánica involucrada en la lógica del pseudolenguaje, nos permite trabajar con ella y resolver variados problemas y avanzar mucho en el conocimiento de la programación.

Pero finalmente no estamos programando realmente, solo ejercitamos la forma en que se resuelven problemas, el siguiente paso es traducir esto a un lenguaje formal como Java, C#, C, C++ etc.

Lógica Computacional

La traducción para algunos estudiantes es un paso traumático, el hecho de sentir que se perdió tiempo en este proceso es un sentimiento generalizado.

Con esto no quiero decir que el PseudoLenguaje es una pérdida de tiempo simplemente opino que cuando los módulos son muy acotados es necesario optimizar el tiempo de aprendizaje, invirtiendo el mayor tiempo necesario en trabajar en un lenguaje formal, y allí realizar todos los ejercicios de lógica, los ejemplos básicos para que el estudiante sienta que está realmente trabajando en un lenguaje real y que podrá aplicarlo en la vida real y en funciones orientadas a su carrera.

Lógica Computacional

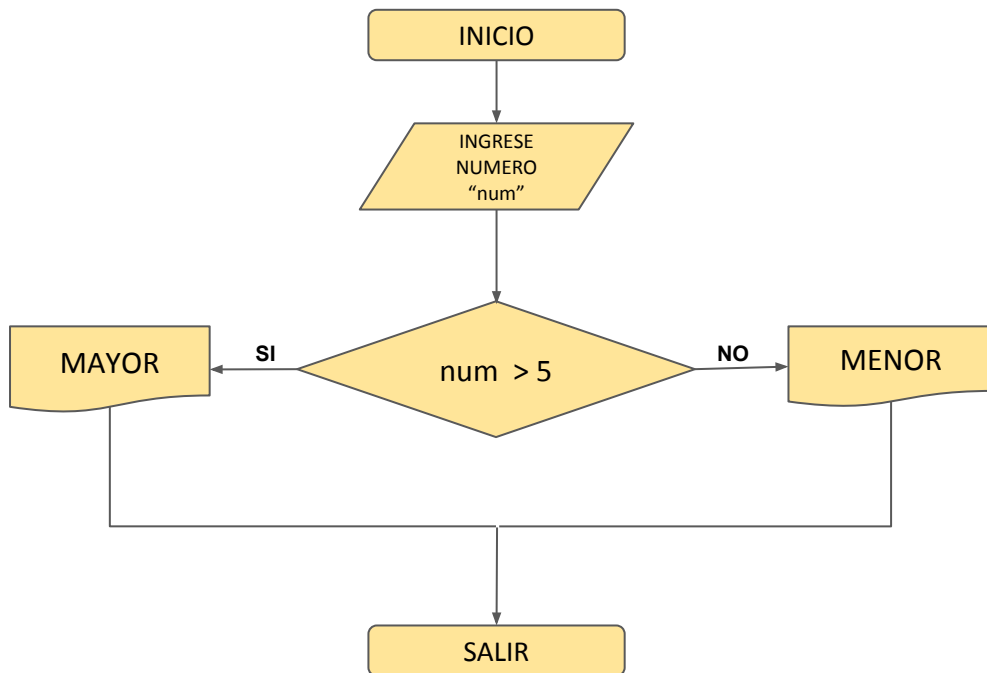
Pseudolenguaje

```
Mientras num > 0 Hacer
    Si num > 5 Entonces
        Escribir "Mayor";
    Sino
        Escribir "Menor";
    Fin Si
Fin Mientras
```

C#

```
While ( num > 0){
    if( num > 5){
        Console.WriteLine("Mayor");
    }else{
        Console.WriteLine("Menor");
    }
}
```

Diagrama de flujo





Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

2.- Diagramas de Flujo

Docente: Michell Jáuregui Valdés

Diagramas de flujo







“Un diagrama de flujo es un diagrama que describe un proceso, sistema o algoritmo informático.”

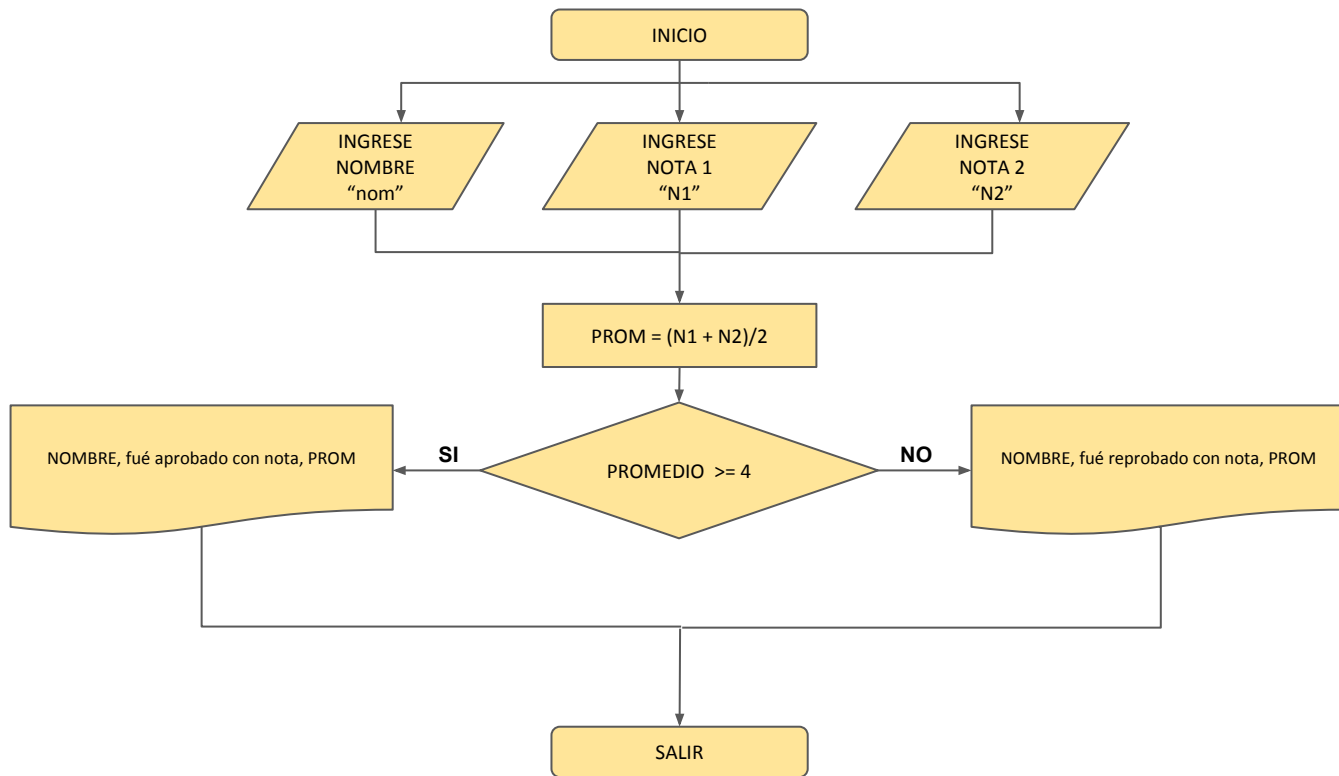
Una forma fácil y rápida para graficar un algoritmo informático, es utilizando un diagrama de flujo, su utilidad radica en el hecho de que en una simple hoja de papel podemos describir claramente ¿qué hace nuestro programa?, ¿cómo realiza esta acción? y ¿qué resultado obtenemos del todo ese proceso?.

La utilidad de este tipo de diagramas es fundamental cuando queremos explicar cómo debe funcionar la lógica de nuestro programa.

Es una herramienta muy poderosa y muy útil.

Diagramas de flujo

	Inicio / Fin
	Control de Flujo
	Proceso
	Entrada / Input
	Decisión
	Mostrar / Output



```
Console.Write("Ingrese Nombre: ");

string nom = Console.ReadLine();

Console.Write("Ingrese un numero 1: ");

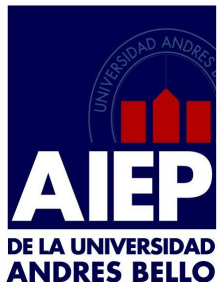
double n1 = Convert.ToDouble(Console.ReadLine());

Console.Write("Ingrese un numero 2: ");

double n2 = Convert.ToDouble(Console.ReadLine());

double promedio = (n1 + n2) / 2;

if (promedio >= 4)
{
    Console.WriteLine(nom+" fue aprobado con nota: "+promedio);
}
else
{
    Console.WriteLine(nom + " fue reprobado con nota: " + promedio);
}
```



Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

3.- Hola Mundo con C#

Docente: Michell Jáuregui Valdés

Variables



Para estudiar de una forma más práctica las estructuras y la mecánica de la programación nos basaremos en el lenguaje de programación C#.

- 1) Es un lenguaje muy amigable con los programadores.
- 2) La plataforma .NET permite crear cualquier tipo de aplicación.

Aplicaciones de escritorio.

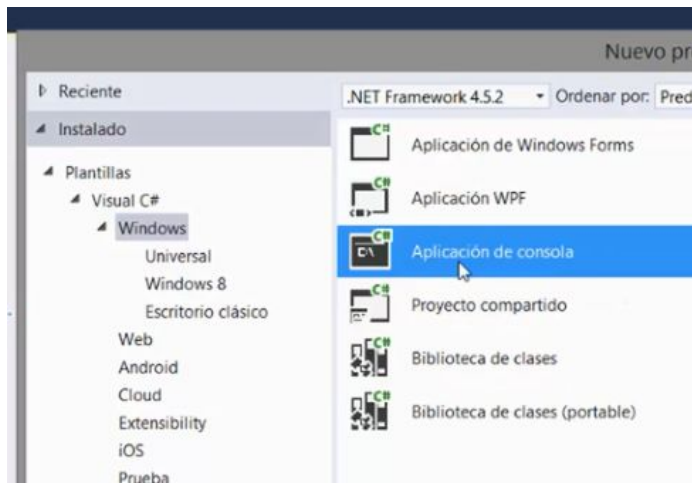
Aplicaciones Móviles Híbridas (con Xamarin).

Aplicaciones Web con MVC.
- 3) Ahora es Open Source.
- 4) Existe una gran demanda de programadores C#.
- 5) Su sintaxis es similar a Java, C y C++ por lo tanto luego de aprender C# puedes aprender estos otros lenguajes con menos dificultad.

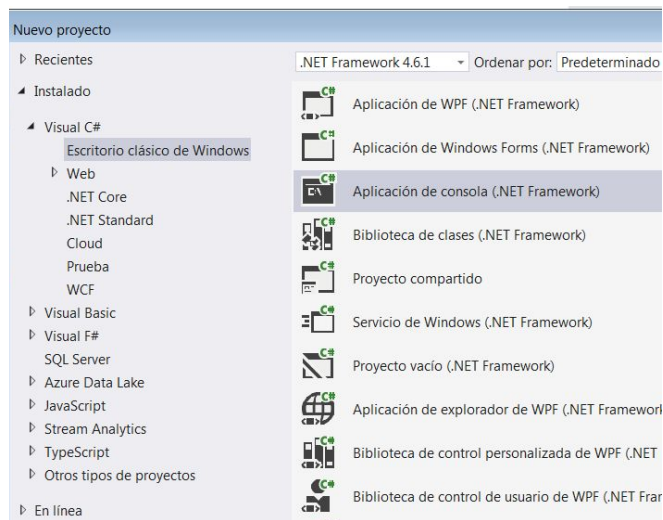
Hola mundo con C#

1) Abrir Visual Studio ir a: Archivo/Nuevo/Proyecto

Visual Studio Community 2015



Visual Studio Enterprise 2017



Hola mundo con C#

2) Escribir el nombre de la aplicación.

Nombre: HolaMundo

Ubicación: C:\Users\...

Solución: Crear nueva solución

Nombre de la solución: HolaMundo

☒ Crear directorio para la solución

☐ Agregar a control de código fuente

Examinar...

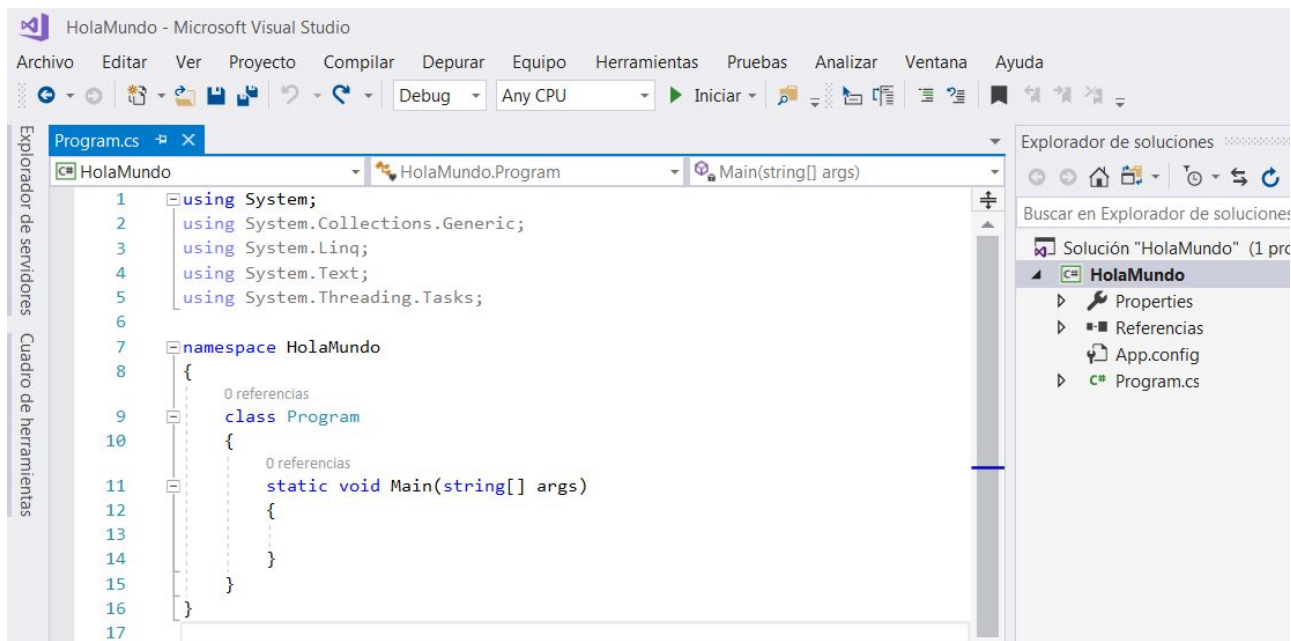
Aceptar Cancelar

3) Seleccionar “Examinar” para crear la ruta de nuestro proyecto.

4) Seleccionar “Aceptar”.

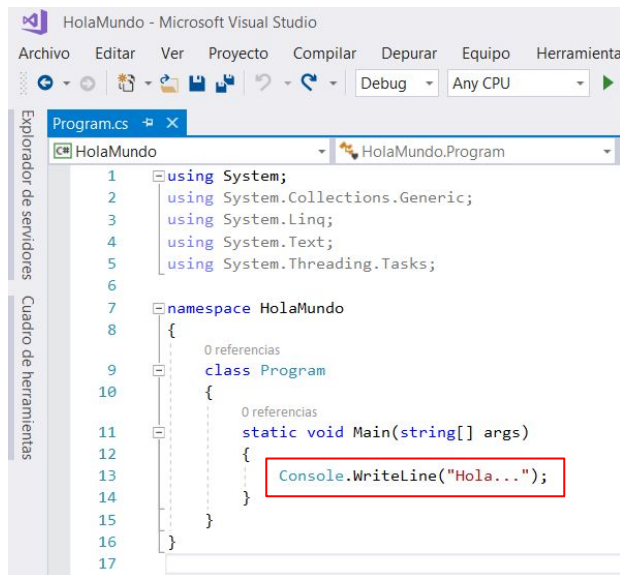
Hola mundo con C#

5) Este será el resultado:

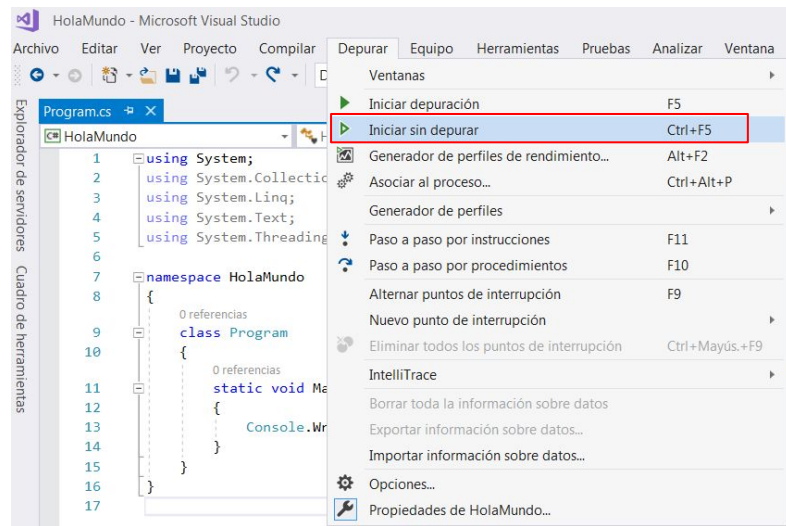


Hola mundo con C#

- 6) Escribir el siguiente Código:
`Console.WriteLine("Hola...");`

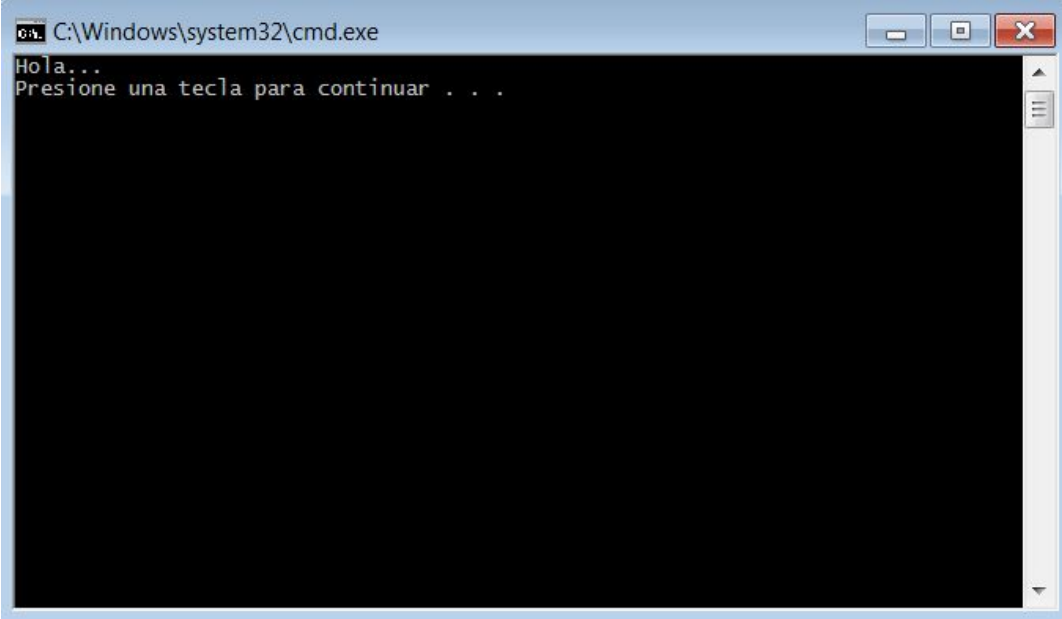


- 7) Seleccionar “Iniciar sin Depurar”



Hola mundo con C#

7) Si todo sale bien éste será el resultado:

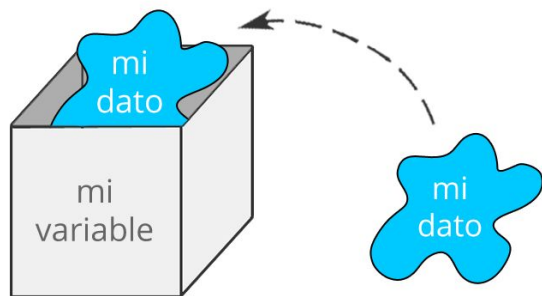


A screenshot of a Windows Command Prompt window. The title bar shows the path `C:\Windows\system32\cmd.exe`. The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt displays the text `Hola...` on the first line and `Presione una tecla para continuar . . .` on the second line. The rest of the window is black, indicating it is waiting for a key press to continue.

Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

4.- Variables



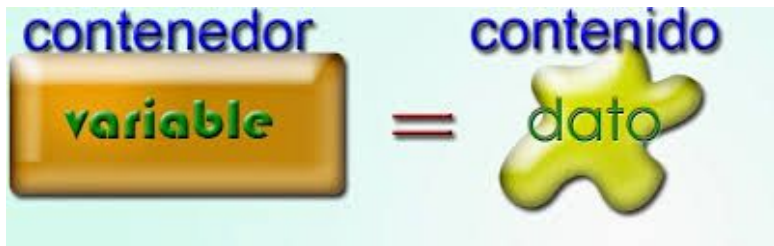
Docente: Michell Jáuregui Valdés

Variables

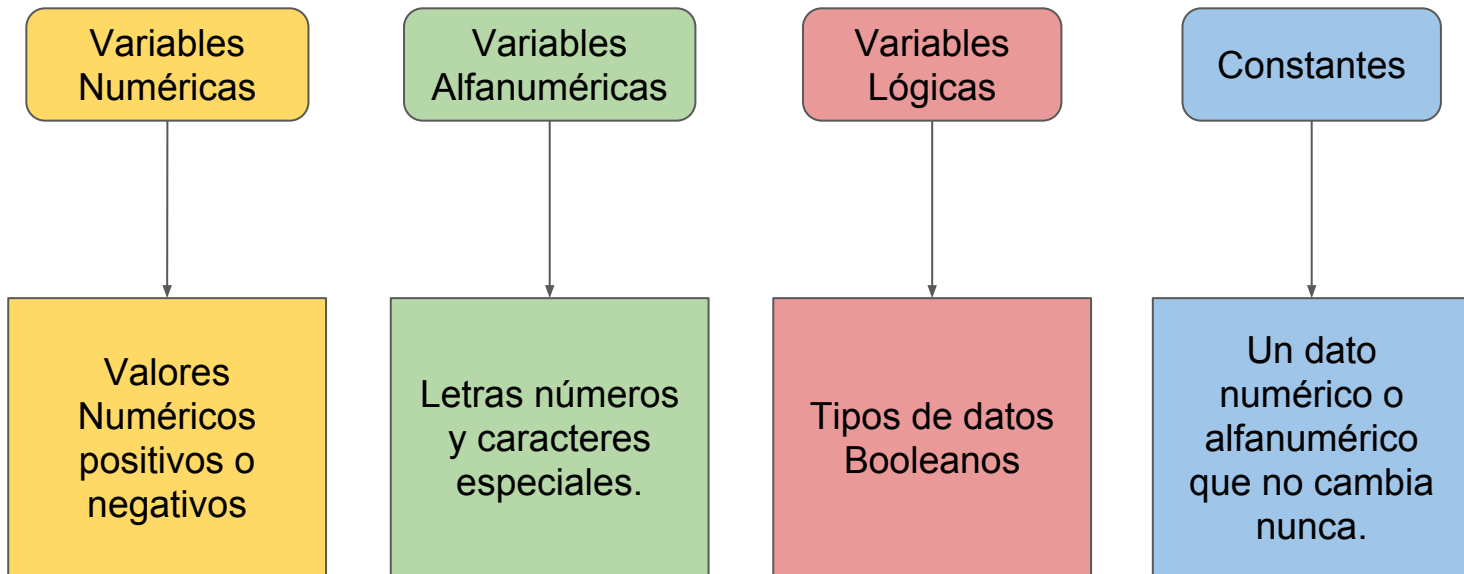
Una variable está formada por un espacio en el sistema de almacenaje (Contenedor) y un nombre simbólico (Nombre de la variable) que está asociado a dicho espacio. Este espacio contiene información la cual es el (Contenido o el valor de la variable) representado por datos.

x = "juan";

¿Cuanto vale la variable **x**?



Tipos de datos



Tipos de Variables

	Rango de la Variable
int	2.147.483.647 y -2.147.483.648
long	9.223.372.036.854.775.807 y -9.223.372.036.854.775.808
float	3,402823e38 y -3,402823e38
double	1,79769313486232e308 y -1,79769313486232e308
char	Un carácter Unicode. *
string	Una cadena de caracteres Unicode.
bool	True o False.

Ejemplo de variables con valores

Enteros:

```
int a = 4;
```

```
int b = 6;
```

```
int c = 12;
```

Una variable de tipo **int** permite ingresar solo valores enteros.

Ejemplo de variables con valores

Decimales:

double n1 = 6,5;

double n2 = 5,5;

double n3 = 4,5;

Una variable de tipo **double** permite ingresar valores enteros y valores decimales.

Ejemplo de variables con

Caracteres Alfanuméricos:

```
char letra= "a";
```

```
char letra= "7";
```

```
char letra= "i";
```

Una variable de tipo **char** permite ingresar solo 1 carácter, pero incluye todos los valores de la Tabla Ascii.

Una secuencia caracteres de tipo **char** forman una cadena de tipo **string**.

Ejemplo de variables con

Cadena de Caracteres Alfanuméricos:

```
string nom1 = "H";
```

```
string nom2 = "1 + 1";
```

```
string nom3 = "Cuevas N° 66";
```

Una variable de tipo **string** permite ingresar cadenas de caracteres de tipo Alfanuméricos, esto incluye Números, símbolos etc (Ver Tabla Ascii). Pero los números y los símbolos Ej. (!= []{})^{*} / + - %) se comportarán como letras perdiendo todo su valor matemático.

En el ejemplo "1 + 1", no generará ningún tipo de resultado y se imprimirá en pantalla como un texto simplemente.

Ejemplo de variables

Booleanas:

```
bool res = true;
```

```
bool res = false;
```

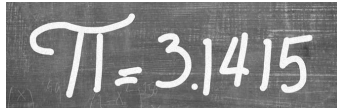
Una variable de tipo **bool** tiene 2 estados. Verdadero o Falso.

Ejemplo de

Constantes:

```
const pi = 3.1415;
```

Una tipo de dato **const** permite ingresar un valor pero este nunca podrá cambiar dentro de la ejecución del programa.

A photograph of a chalkboard with the mathematical constant pi (π) written in white chalk, followed by an equals sign and the decimal value 3.1415.
$$\pi = 3.1415$$

Tipos de Variables - Tabla de caracteres

* **Unicode** es el estándar de codificación de caracteres universal utilizado para la representación de texto para procesamiento del equipo. Unicode proporciona una manera consistente de codificación de texto multilingüe y facilita el intercambio de archivos de texto internacionales.

* **Ascii.** Sistema de codificación de caracteres alfanuméricos que asigna un número del 0 al 127 a cada letra, número o carácter especial recogidos; el ASCII extendido permite hasta 256 caracteres distintos.

Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sinc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles		
32	espacio	
33	!	
34	"	
35	#	
36	\$	
37	%	
38	&	
39	'	
40	(
41)	
42	*	
43	+	
44	,	
45	-	
46	.	
47	/	
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	
59	;	
60	<	
61	=	
62	>	
63	?	
64	@	
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	
87	W	
88	X	
89	Y	
90	Z	
91	[
92	\	
93]	
94	^	
95	_	
96	`	
97	a	
98	b	
99	c	
100	d	
101	e	
102	f	
103	g	
104	h	
105	i	
106	j	
107	k	
108	l	
109	m	
110	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
120	x	
121	y	
122	z	
123	{	
124		
125	}	
126	~	

ASCII extendido			
128	Ç	160	á
129	ü	161	í
130	é	162	ó
131	â	163	ú
132	ä	164	ñ
133	à	165	Ñ
134	á	166	ª
135	ç	167	º
136	ê	168	¿
137	ë	169	©
138	è	170	¬
139	ï	171	½
140	ì	172	¼
141	í	173	¿
142	À	174	«
143	Á	175	»
144	É	176	»
145	æ	177	»
146	Æ	178	»
147	ô	179	»
148	ö	180	»
149	õ	181	À
150	ù	182	Â
151	û	183	Ä
152	ÿ	184	©
153	Ö	185	»
154	Ü	186	»
155	ø	187	»
156	£	188	»
157	Ø	189	»
158	×	190	¥
159	f	191	¬
192	Ł	224	Ó
193	ł	225	ß
194	ł	226	Ô
195	ł	227	Õ
196	ł	228	ö
197	ł	229	Ö
198	ł	230	μ
199	ł	231	ρ
200	ł	232	ρ
201	ł	233	Ů
202	ł	234	Ů
203	ł	235	Ů
204	ł	236	Ÿ
205	ł	237	Ÿ
206	ł	238	—
207	ł	239	·
208	ł	240	≡
209	ł	241	±
210	ł	242	—
211	ł	243	¼
212	ł	244	¶
213	ł	245	§
214	ł	246	÷
215	ł	247	°
216	ł	248	°
217	ł	249	°
218	ł	250	°
219	ł	251	°
220	ł	252	°
221	ł	253	°
222	ł	254	■
223	ł	255	nbsp

Declarando e Inicializando Variables tipo “int”

DECLARANDO VARIABLE TOTAL

Tipo de variable

Nombre de la variable

`int total;`

La variable “total” fue declarada de tipo “int” esto significa que es un entero y su rango está entre (-2.147.483.648 y 2.147.483.647).

La variable “total” solo puede recibir enteros.

INICIALIZANDO O ASIGNAR VALOR A LA VARIABLE TOTAL

Nombre de la variable

Valor asignado a la variable

`total = 50;`

Si usted asigna un alfanumérico a esta variable, se producirá un error.

Declarando e Inicializando Variables tipo “float”

DECLARANDO VARIABLE TOTAL

Tipo de variable

Nombre de la variable

`float total;`

INICIALIZANDO O ASIGNAR VALOR A LA VARIABLE TOTAL

Nombre de la variable

Valor asignado a la variable

`total = 6,5;`

La variable “total” fue declarada de tipo “float” esto significa que es un entero y su rango está entre (-3,402823e38 y 3,402823e38).

La variable “total” puede recibir números enteros y con decimales.

Por ejemplo, es ideal para sacar un promedio de notas.

Si usted asigna un alfanumérico a esta variable, se producirá un error.

Declarando e Inicializando Variables tipo “float”

```
float nota1;  
float nota2;  
float promedio;
```

```
nota1 = 5,6;  
nota2 = 4,0;
```

```
promedio = (nota1 + nota2)/2;
```

El resultado es:

4,8

Declarando e Inicializando Variables tipo “string”

DECLARANDO VARIABLE TOTAL

Tipo de variable

Nombre de la variable

`string nombre;`

INICIALIZANDO O ASIGNAR VALOR A LA VARIABLE TOTAL

Nombre de la variable

Valor asignado a la variable

`nombre = “juan”;`

La variable “**nombre**” fue declarada de tipo “**string**” esto significa que es una cadena de caracteres alfanuméricos Unicode.

La variable “**nombre**” puede recibir caracteres alfanuméricos (letras, numeros, pero a los números se comportan como cadenas).

```
string calle;  
string numero;  
string direccion;
```

```
calle = “Alameda”;  
numero = “433”;
```

```
direccion = calle+” “+numero;
```

El resultado es:

“Alameda 433”

Declarando e Inicializando Variables tipo “string”

```
string nota1;  
string nota2;  
string promedio;
```

```
nota1 = “5,6”;  
nota2 = “4,0”;
```

```
promedio = nota1 + nota2;
```

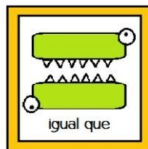
El resultado es:

“5,64,0”

Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

5.- Operadores Aritméticos y Lógicos



Docente: Michell Jáuregui Valdés

Tabla de Operadores Aritméticos

Operación C#	Operador aritmético	Expresión algebraica	Expresión en C#
Suma	+	$f + 7$	$f + 7$
Resta	-	$p - c$	$f - c$
Multiplicación	*	$b * m$	$b * m$
División	/	x/y	x / y
Residuo	%	$r \text{ mod } s$	$r \% r$
Incremento	++	$i + 1$	$i ++$
Decremento	--	$i - 1$	$--i$

Tabla de Operadores Lógicos de Igualdad y Relaciones

Operadores estándar algebraicos	Operador de C#	Operador de C#	Significado de la condición en C#
Operadores de Igualdad			
=	==	x == y	x es igual a y
≠	!=	x != y	x no es igual a y
Operadores relacionales			
>	>	x > y	x es mayor que y
<	<	x < y	x es menor que y
≥	>=	x >= y	x es mayor o igual que y
≤	<=	x <= y	x es menor o igual que y
Operadores logicos			
Y	&&	(x > 3) && (x < 3)	(x es mayor que 3) y (x es menor que 3)
O		(x > 3) (x < 3)	(x es mayor que 3) o (x es menor que 3)
NO	NOT	NOT (x = 3)	Negacion de (x es igual a 3)

Tabla de Precedencia de Operadores Aritméticos.

Operador	Operación	Orden de evaluación
Se evalúa Primero		
*	Multiplicación	Si hay varios operadores de este tipo, se evalúan de izquierda a derecha
/	División	
%	Residuo	
Se evalúa primero		
+	Suma	Si hay varios operadores de este tipo, se evalúan de izquierda a derecha
-	Resta	

Tabla de secuencia de Escape

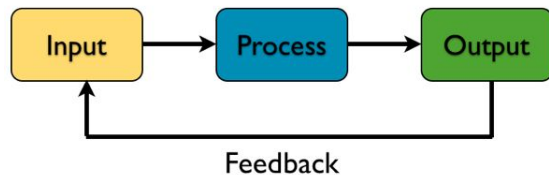
Secuencia de Escape	Descripción
\n	Nueva línea. Posiciona el cursor de la pantalla en el comienzo de la línea.
\t	Tabulación horizontal. Desplaza el cursor de la pantalla a la siguiente marca de tabulación
\r	Retorno de carro. Posiciona el cursor al comienzo de la línea actual.
\\	Barra diagonal inversa. Se utiliza para colocar un carácter de barra diagonal inversa en la cadena
\"	Se utiliza para colocar un carácter en doble comilla Ejemplo: <code>console.write("\"entre comillas\");</code>



Módulo:

EIT 218 - TALLER DE PROGRAMACIÓN I

6.- Datos: Ingreso y Salida



Docente: Michell Jáuregui Valdés

Ingreso de Datos

Cuando necesitamos ingresar datos a nuestro programa utilizaremos los siguientes comandos **Console.Write()** y **Console.ReadLine()** , esto nos permitirá ingresar textos, valores etc. a nuestro programa.

```
Console.Write("Ingrese su nombre:");
```

Ingrese su nombre:

```
string nom = Console.ReadLine();
```

Ingrese su nombre: **JUAN**

Ahora la variable **nom** contiene el valor **JUAN**

Salida de Datos

Para mostrar una cadena de texto, una variable o una mezcla de ambas se utiliza la instrucción **Console.WriteLine()** esto nos permite mostrar por consola los resultados obtenidos luego de un proceso o simplemente mandar un mensaje.

```
Console.WriteLine("Su nombre es: ", nom);
```

Su nombre es: **JUAN**

Salida de Datos

Otro ejemplo de la utilización del comando **Console.WriteLine()** en este caso utilizamos una variable de tipo double la cual contiene un promedio con 1 decimal, también se puede utilizar datos tipo, int, string, bool, const etc.

```
Console.WriteLine("Su promedio es: ");
```

Su promedio es:

```
double prom = 6.5;  
Console.WriteLine("Su promedio es: ", prom);
```

Su promedio es: 6,5

Ingreso de Datos y Conversión a otros formatos.

Cuando ingresamos datos a nuestro programa utilizando los comandos **Console.Write()** y **Console.ReadLine()**, obtenemos los datos en formato **string** por defecto.

¿Pero qué sucede cuando ingresamos un **número** en formato **int**?

Para eso se utiliza el comando **Convert.ToInt32()**

```
Console.Write("Ingrese un numero entero: ");  
  
int num = Convert.ToInt32(Console.ReadLine());  
  
Console.WriteLine(num);
```

Ingrese un número entero: 7

Ahora la variable **num** contiene el valor 7

Ingreso de Datos y Conversión a otros formatos.

Ahora quiero ingresar un número de tipo **double**.

Para eso se utiliza el comando **Convert.ToDouble()**

```
Console.Write("Ingrese un numero con decimales: ");  
  
int num = Convert.ToDouble (Console.ReadLine());  
  
Console.WriteLine(num);
```

Ingrese un número con decimales: 6,5

Ahora la variable **num** contiene el valor 6,5

Salida de Datos con más de 1 variable

Ahora quiero mostrar por pantalla varias variables de distinto tipo al mismo tiempo.

```
double num = 6.5;  
string nombre = "Juan";  
string estado = "Aprobado";  
  
Console.WriteLine("Promedio: {0} {1} {2}", num , nombre , estado);
```

Promedio: 6,5 Juan Aprobado

Módulo:

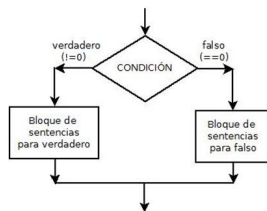
EIT 218 - TALLER DE PROGRAMACIÓN I

7.- Sentencias Condicionales

Instrucción *if-else* en C:

```
if (condición)
{
    sentencia_v1;
    ...
    sentencia_vn;
}
else
{
    sentencia_f1;
    ...
    sentencia_fn;
}
```

if-else (alternativa doble):



Docente: Michell Jáuregui Valdés

Sentencias Condicionales - if, else

La sentencia If ejecuta una acción indicada sólo cuando la condición es verdadera; de lo contrario la acción es pasada por alto.

```
If (nota >= 4)
{
    Console.WriteLine("Aprobado");
}
```

La instrucción de selección If else nos permite especificar una acción a realizar cuando la condición es verdadera y una acción cuando la condición es falsa.

```
if (nota >= 4)
{
    Console.WriteLine("Aprobado");
}
else
{
    Console.WriteLine("Reprobado");
}
```


Sentencias Condicionales - if, else

```
int num;
```

```
num = 10;
```

```
if (num > 5)
```

```
{
```

```
    Console.WriteLine("num es mayor que 5");
```

```
}
```



Resultado:

num es mayor que 5

```
int num;
```

```
num = 10;
```

```
if (num > 20)
```

```
{
```

```
    Console.WriteLine("num es mayor que 20");
```

```
}else{
```

```
    Console.WriteLine("num es menor que 20");
```

```
}
```



Resultado:

num es menor que 20

Sentencias Condicionales - if else anidadas

La instrucción If...else puede estar dentro de otras instrucciones.

Ejemplo:

Imprimir A para notas de los exámenes que sean mayores a 90, B para el rango de 80 a 89, C para el rango de 70 a 79, D para el rango de 60 a 69 y F para todas las demás:

```
if (nota >= 90)
{
    Console.WriteLine("A");
}
else
{
    if (nota >= 80)
    {
        Console.WriteLine("B");
    }
    else
    {
        if (nota >= 70)
        {
            Console.WriteLine("C");
        }
        else
        {
            if (nota >= 60)
            {
                Console.WriteLine("D");
            }
            else
            {
                Console.WriteLine("F");
            }
        }
    }
}
```

Sentencias Condicionales - if anidado utilizando else if

```
int num = 4;  
if (num > 5)  
{  
    Console.WriteLine("num es mayor que 5");  
}  
else if (num < 5)  
{  
    Console.WriteLine("num es Menor que 5");  
}
```



Resultado:

num es Menor que 5

```
int a = 8;  
int b = 8;  
if (a > b)  
{  
    Console.WriteLine("{0} > {1}", a, b);  
}  
else if (a < b)  
{  
    Console.WriteLine("{0} < {1}", a, b);  
}  
else  
{  
    Console.WriteLine("{0} = {1}", a, b);  
}
```



Resultado:

8 = 8

Sentencias Condicionales - Switch

La instrucción de selección múltiple switch permite que en un algoritmo contenga una serie de decisiones, en las cuales una variable o expresión se probará por separado contra cada uno de los valores constantes enteros que puede asumir, y se tomarán diferentes acciones.

Ejemplo:

- Obtener el día de semana en palabras al optar entre los números 1 al 7.

```
int op;
Console.WriteLine("1- Lunes");
Console.WriteLine("2- Martes");
Console.WriteLine("3- Miercoles");
Console.WriteLine("4- Jueves");
Console.WriteLine("5- Viernes");
Console.WriteLine("6- Sabado");
Console.WriteLine("7- Domingo");
Console.Write("Escriba el numero de la semana 1 a 7");
op=Convert.ToInt32(Console.ReadLine()); //lee la entrada del usuario
switch(op)
{
    case 1: Console.Write("Eligio Lunes");
            break;
    case 2: Console.Write("Eligio Martes");
            break;
    case 3: Console.Write("Eligio Miercoles");
            break;
    case 4: Console.Write("Eligio Jueves");
            break;
    case 5: Console.Write("Eligio Viernes");
            break;
    case 6: Console.Write("Eligio Sabado");
            break;
    case 7: Console.Write("Eligio Domingo");
            break;
    default: Console.Write("Solo opte por numeros 1 al 7");
            break;
} //fin switch
```

Sentencias Condicionales - Switch

```
string dia = "3";  
switch (dia)  
{  
    case "1":  
        Console.WriteLine("Día lunes");  
        break;  
    case "2":  
        Console.WriteLine("Día Martes");  
        break;  
    case "3":  
        Console.WriteLine("Día Miércoles");  
        break;  
    case "4":  
        Console.WriteLine("Día Jueves");  
        break;  
    case "5":  
        Console.WriteLine("Día Viernes");  
        break;  
    default:  
        Console.WriteLine("Producto no encontrado");  
        break;  
}
```



Resultado:

Día Miércoles

Módulo:

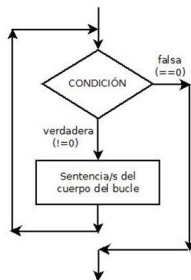
EIT 218 - TALLER DE PROGRAMACIÓN I

8.- Ciclos Iterativos

Bucle *while* en C:

```
while (condición)
{
    sentencia_1;
    ...
    sentencia_n;
}
```

Bucle *while* (mientras):

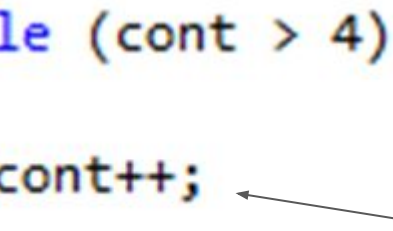


Docente: Michell Jáuregui Valdés

Ciclos de repetición - while

Esta sentencia permite ejecutar repetidamente, mientras se cumpla una determinada condición, una sentencia o bloque de sentencias.

variable contador



```
while (cont > 4)
{
    cont++;
}
```

**incrementando la
variable contador**

Ciclos de repetición - while

```
int n = 0; // declaración del contador
while (n < 5)
{
    Console.WriteLine("Valor del contador n: {0}", n);
    n++;
}
```

Resultado:

Valor del contador n: 0

Valor del contador n: 1

Valor del contador n: 2

Valor del contador n: 3

Valor del contador n: 4

Ciclos de repetición - do while

Esta instrucción do...while evalúa la condición a continuación de ciclo... después de ejecutar el cuerpo; por lo tanto, éste siempre se ejecuta cuando menos una vez.

```
do  
{  
    cont++;  
} while (i > cont);
```

**incrementando la
variable contador**



variable contador



Ciclos de repetición - do while

```
int c = 0;

do
{
    Console.WriteLine("Valor del contador c: {0}", c);
    c++;
} while (c < 5);
```

Resultado:

Valor del contador c: 0

Valor del contador c: 1

Valor del contador c: 2

Valor del contador c: 3

Valor del contador c: 4

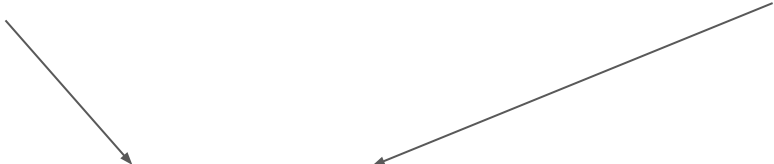
Ciclos de repetición - for

for es quizás el tipo de bucle más versátil y utilizado del lenguaje C#.

Permite ejecutar una sentencia repetidamente un número determinado de veces.

variable contador

**incrementando la
variable contador**



```
for ( i = 0; i < 5; i++)
```

```
{
```

```
    Console.WriteLine("Valor del contador i: {0}", i);
```

```
}
```

Ciclos de repetición - for

```
for (i = 0; i < 5; i++)  
{  
    Console.WriteLine("Valor del contador i: {0}", i);  
}
```

Resultado:

Valor del contador i: 0

Valor del contador i: 1

Valor del contador i: 2

Valor del contador i: 3

Valor del contador i: 4