Programación en C#

Elementos del lenguaje

Elemento	*Ejemplos*
palabraclave	`public class int`
literal	`7 18 14`
operador	`+ - * / % =`
identificador	`suma Console nota`
puntuación	`; { } ,`

Palabras clave

```
abstract  as        base      bool      break
byte      case      catch     char      checked
class     const     continue  decimal   default
delegate  do        double    else      enum
event     explicit  extern    false     finally
fixed     float     for       foreach   goto
if        implicit  in        int       interface
internal  is        lock      long      namespace
new       null      object    operator  out
override  params    private   protected public
readonly  ref       return    sbyte     sealed
short     sizeof    stackalloc static   string
struct    switch    this      throw     true
try       typeof    uint      ulong     unchecked
unsafe    ushort    using     virtual   void
while
```

Literales

tipo	*literales*
logicas	true false
reales	3.14159
enteras	7 4 14 42
caracter	'X'
cadena	"Hello"
nulo	null

Operadores

categoría	*operadores*
aritméticos	`+ - * / %`
relacionales	`== != < <= > >=`
lógicos	`&& || ! ?:`
asignación	`=`

Identificadores

- **Reglas para construirlos**
  - **Pueden contener letras y números**
  - **Deben empezar con una letra**
  - **Diferencian minúsculas y mayúsculas**
- **Recomendaciones**
  - **Utilizar nombres con sentido**
  - **Evitar en lo posible las abreviaturas**
  - **Seguir los convenios de mayus./minus.**
    - **camelCase**
    - **PascalCase**

Identificadores (II)

Ejemplos

- **nota1 es correcto**
- **1nota no es correcto**
- **nota no es igual que Nota**
- **camelcase: notaFinal**
- **pascalcase: NotaFinal**
- **Preferible NotaFinal a NFin**

Namespaces

- **En aplicaciones grandes pueden existir conflictos en la nomenclatura de los tipos**
- **Los tipos se agrupan en categorias o Namespaces**
- **Los Namespaces deben de ser referenciados para utilizar los tipos incluidos en ellos**
- **Podemos incluir un namespace al principio del programa con la sentencia 'using'**
- **Los Namespaces más utilizados son:**
  - **System**
  - **System.IO**
  - **System.Windows.Forms**
  - **System.Drawing**

Namespaces (II)

```
class HolaMundo
{
   static void Main( )
   {
      System.Console.WriteLine("Hola Mundo");
   }
}
```

```
using System;

class HolaMundo
{
   static void Main( )
   {
      Console.WriteLine("Hola Mundo");
   }
}
```

```
Hola Mundo
_
```

Signos de puntuación

- En el código los espacios en blanco y los saltos de línea se ignoran
- El código está formado por sentencias
- Con punto y coma (;) indicamos el final de una sentencia
- Varias sentencias pueden estar agrupadas en un bloque delimitado por llaves ({...})

Signos de puntuación (II)

```
using System;
class Notas
{
   static void Main( )
   {
      int teoria = 4;
      int practica = 2;
      int final;
      final = teoria + practica;
      Console.WriteLine(final);
   }
}
```

```
6
_
```

Declaraciones

- Las declaraciones introducen una variable en un bloque
- Una variable tiene un identificador y un tipo
- Al declarar una variable podemos asignarle un valor inicial
- En una sentencia podemos declarar varias variables de un mismo tipo
- Ejemplos
  - int nota;
  - int nota = 5;
  - int practica = 4, teoria = 2, nota;
  - string nombre;
  - string apellido1 = “Cuartas”, apellido2 = “Hernandez”;

Expresiones

- Las expresiones generan un valor
- No podemos utilizar una variable en una expresión a menos que antes le asignemos un valor
- Ejemplo:

```
int a = 4, b = 2, c;
c = b * (a - 1);
```

- Ejemplo:

```
int a, b = 2, c;
c = b * ((a = 4) – 1);
```

Clase Console

- Se encuentra en el namespace System.
- Nos permite la entrada y salida de datos.
- Console.ReadLine( );
  Lee un dato del teclado al pulsar intro.
  Devuelve un string.
- Console.Write(string);
  Escribe un string en la consola.
- Console.WriteLine(string);
  Escribe un string en la consola y salta a la siguiente linea.
- Tanto Write como WriteLine aceptan también otros tipos además de string.

Clase Console (II)

Ejemplo de entrada y salida de datos

```
using System;
class Saludo
{
   static void Main( )
   {
      string nombre;
      Console.WriteLine("Como te llamas ?");
      nombre = Console.ReadLine( );
      Console.Write("Hola, ");
      Console.WriteLine(nombre);
      Console.ReadLine( );
   }
}
```

```
Como te llamas ?
Miguel
Hola, Miguel
_
```

Tipos enteros

tipo	*System.*	*Rango*	*signo*	*bits*
sbyte	SByte	-128 a 127	si	8*
ushort	Uint16	0 a 65.535	no	16
uint	UInt32	0 a $4x10^9$	no	32
ulong	UInt64	0 a $18x10^{18}$	no	64
byte	Byte	0 a 255	no	8*
short	Int16	-32.768 a 32.767	si	16
int	Int32	± $2x10^9$	si	32
long	Int64	± $9x10^{18}$	si	64

System.Int32

- int es un alias para System.Int32
  - int suma = 14;
  - System.Int32 suma = 14;
- MaxValue y MinValue nos dan el rango
  - int.MaxValue vale 2.147.483.647
  - int.MinValue vale –2.147.483.648
- Convert.ToInt32(…) para convertir cualquier tipo a System.Int32
  - Convert.ToInt32("14") vale 14 entero
  - Convert.ToInt32(14.73) vale 15 entero

Tipos reales

tipo	*System.*	*Rango*	*dígitos sig.*	*bits*
float	Single	$\pm 3.4 \times 10^{38}$	7	32
double	Double	$\pm 1.7 \times 10^{308}$	15	64
decimal	Decimal	$\pm 7.9 \times 10^{28}$	28	128

System.Double

- double es un alias para System.Double
- Constantes
  - double.MaxValue El mayor double $1.7x10^{308}$
  - double.MinValue El menor double $-1.7x10^{308}$
  - double.Epsilon El double más pequeño mayor que 0.
  - double.NegativeInfinity –Infinito.
  - double.PositiveInfinity +Infinito.
  - double.NaN (Not A Number) Cuando el resultado no es un número.
- Métodos que devuelven true o false (bool)
  - double.IsInfinity(d) Comprueba si d contiene el valor infinito
  - double.IsNaN(d) Comprueba si d contiene el valor NaN

Tipos texto

tipo	*System.*	*Contiene*	*Ejemplo*	*bits*
char	Char	Un caracter	'c'	16
string	String	Varios caracteres	"oceano"	--

Ejemplos

```
char inicial = 'm';
char letra;
char saltoLinea;

letra = 'c';
saltoLinea = '\n';
```

```
string nombre = "Alba";
string apellido1, apellido2;

apellido1 = "Cuartas";
apellido2 = System.Console.ReadLine( );
```

Secuencias de escape

- Representan caracteres especiales
- Comienzan por una barra inclinada (\)
  - \n Salto de línea
  - \t Tabulación
  - \\ Barra inclinada

```
Console.Write("C:\temp\nC:\\temp");
```

```
C:    emp
C:\temp
```

System.Char

- char es un alias para System.Char
- El tipo char almacena un carácter o una secuencia de escape
- El tipo char se puede tratar también como entero (operaciones aritméticas)
- Métodos que devuelven true o false (bool)
  - char.IsDigit(c) Comprueba si el contenido de c es un número
  - char.IsLetter(c) Comprueba si el contenido de c es una letra
  - char.IsWhiteSpace(c) Comprueba si es un espacio en blanco

System.Char (II)

Tabla de códigos ASCII

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

System.Char (III)

```
using System;
class ASCII
{
   static void Main()
   {
      char firstPerson = Convert.ToChar(73);
      char operatorChar = (char)3;
      char a = 'T', b = 'N', c = 'X';
      a = Convert.ToChar(a + 1);
      b = Convert.ToChar(b + 1);
      c = Convert.ToChar(c + 1);
      Console.Write(firstPerson);
      Console.Write(operatorChar);
      Console.Write(c);
      Console.Write(b);
      Console.Write(a);
      Console.ReadLine();
   }
}
```

```
I♥YOU_
```

System.String

- **string es un alias para System.String**
- **El tipo string puede almacenar una cadena de caracteres y secuencias de escape**

- **Para obtener la longitud de un string:**
  VariableNombre.Length
- **Para concatenar strings:**
  El operador +
- **Para acceder a un carácter de un string a través de un índice:**
  VariableNombre[n]

Empieza en 0 !!!!!
Devuelve un char

System.String (II)

```
using System;
class Cadenas
{
   static void Main( )
   {
      string cadena, resultado;
      Console.WriteLine("Introduzca cadena de longitud > 2:");
      cadena = Console.ReadLine( );
      resultado = "Longitud: " + cadena.Length +
         "\nSegundo carácter: " + cadena[1];
      Console.WriteLine(resultado);
      Console.ReadLine( );
   }
}
```

```
Introduzca cadena:
casino
Longitud: 6
Segundo carácter: a
_
```

Tipo booleano

tipo	System.	Contiene	Ejemplo	bits
bool	Boolean	true o false	true	16

Ejemplos

```
int nota;
bool suspenso, aprobado, matricula;

nota = 4;
suspenso = (nota < 5);
aprobado = (nota >= 5);
matricula = (nota == 10);
```

```
bool blanco, botella = true, leche;

blanco = true;

leche = (blanco && botella);
```

Clase Convert

- Se encuentra en el namespace System.
- Nos permite la conversión de tipos.

- Convert.ToInt32(a);
  Devuelve el valor de a convertido a int
- Convert.ToDouble(b);
  Devuelve el valor de b convertido a double
- Convert.ToString(c);
  Devuelve el valor de c convertido a string

- Conversiones explicitas vs implícitas

Clase Convert (II)

Lectura de números por consola

```
using System;
class Convertir
{
   static void Main( )
   {
      Console.Write("Introduzca un valor entero: ");
      int a = Convert.ToInt32(Console.ReadLine( ));
      Console.Write("Introduzca un valor real: ");
      double d = Convert.ToDouble(Console.ReadLine( ));
      Console.ReadLine( );
   }
}
```

Clase Math

- Se encuentra en el namespace System
- Nos permite realizar las operaciones matemáticas más frecuentes.
- Constantes
  - Math.PI = 3.1415926535…
  - Math.E = 2.7182818285…
- Para utilizar sus métodos:

  Math.*NombreMetodo***(***argumento1***, *argumento2***, ...);**
- Ejemplo

  double a = Math.Sqrt(49) + Math.PI;

Clase Math. Métodos usuales.

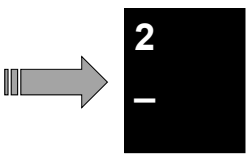
Sqrt (x)	Raiz cuadrada de x.	**Sqrt (9) = 3**
Pow (x, y)	x elevado a y.	**Pow (2, 7) = 128**
Log (x)	Logaritmo de x en base e.	**Log (7.389) = 2**
Exp (x)	e elevado a x.	**Exp (2) = 7.389**
Sin (x)	Seno de x.	**Sin (0) = 0**
Cos (x)	Coseno de x.	**Cos (0) = 1**
Tan (x)	Tangente de x.	**Tan (0) = 0**
Abs (x)	Valor absoluto de x.	**Abs (-2.1) = 2.1**
Ceiling (x)	Redondeo al entero superior.	**Ceiling (2.1) = 3**
Floor (x)	Redondeo al entero inferior.	**Floor(2.1) = 2**
Max (x, y)	Máximo de x e y.	**Max (2.1, 1.5) = 2.1**
Min (x, y)	Mínimo de x e y.	**Min(2.1, 1.5) = 1.5**

Caso de la división entera

Problema:

Si en una expresión participan enteros, el resultado es entero.

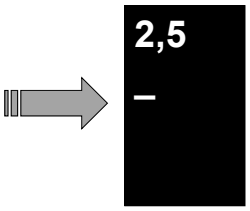
```
int total = 5;
double mitad;
mitad = total / 2;
Console.WriteLine(mitad);
```



Solución:

Convertir alguno de los operandos de la expresión en double.

```
int total = 5;
double mitad;
mitad = Convert.ToDouble(total) / 2;
// ó también: mitad = (double)total / 2;
// ó también: mitad = total / 2.0;
Console.WriteLine(mitad);
```



Otros operadores de asignación

Nos sirven para reducir código

Suma = Suma + 14 es igual que Suma += 14

Suponiendo int a = 10;

Operador	*Ejemplo*	*Explicación*	*Resultado*
+=	a += 5	a = a + 5	a vale 15
-=	a -= 6	a = a - 6	a vale 4
*=	a *= 4	a = a * 4	a vale 40
/=	a /= 2	a = a / 2	a vale 5
%=	a %= 3	a = a % 3	a vale 1

Operadores de incremento y decremento

- **Operador incremento (x++)**
  - **Añade 1 a una variable**
  - **Es lo mismo que x = x + 1**
- **Operador decremento (x--)**
  - **Resta 1 a una variable**
  - **Es lo mismo que x = x – 1**
- **PreIncremento y PostIncremento**
  - **x++ ó x-- Primero ejecutan la acción y luego añaden o restan uno a la variable.**
  - **++x ó -- x Primero añaden o restan uno a la variable y luego ejecutan la acción.**

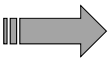
Operadores de incremento y decremento (II)

Operador	*Nombre*	*Ejemplo*
++	PreIncremento	++a
Incrementa a en 1, luego utiliza el nuevo valor de a en la expresión		
++	PostIncremento	a++
Usa el valor de a en la expresión y después lo incrementa en 1		
--	PreDecremento	--a
Decrementa a en 1, luego utiliza el nuevo valor en la expresión		
--	PostDecremento	a--
Usa el valor de a en la expresión y después lo decrementa en 1		

Operadores de incremento y decremento (III)

PostIncremento

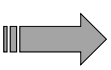
```
int c = 5;
Console.WriteLine(c);
Console.WriteLine(c++);
Console.WriteLine(c);
```



```
5
5
6
_
```

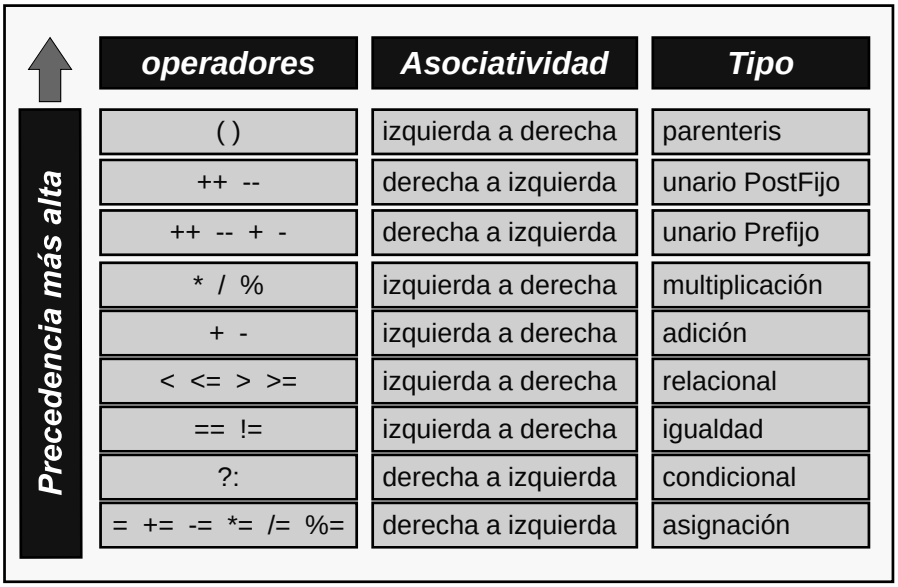
PreIncremento

```
int c = 5;
Console.WriteLine(c);
Console.WriteLine(++c);
Console.WriteLine(c);
```



```
5
6
6
_
```

Precedencia y Asociatividad



operadores	Asociatividad	Tipo
( )	izquierda a derecha	parenteris
++ --	derecha a izquierda	unario PostFijo
++ -- + -	derecha a izquierda	unario Prefijo
* / %	izquierda a derecha	multiplicación
+ -	izquierda a derecha	adición
< <= > >=	izquierda a derecha	relacional
== !=	izquierda a derecha	igualdad
?:	derecha a izquierda	condicional
= += -= *= /= %=	derecha a izquierda	asignación

Comentarios

- **Se utilizan para hacer aclaraciones sobre el código**
- **Los comentarios son ignorados por el compilador**
- **Comentarios en una linea //...**
- **Comentarios en múltiples lineas /*...*/**

```
class HolaMundo
{
   /*  Este programa muestra por pantalla
       la frase Hola mundo */
   static void Main ( )
   {
      System.Console.WriteLine("Hola Mundo");
   }
 } // Aquí finaliza el programa
```

Escribiendo código

- **Buscar un estilo propio para escribir un código claro y legible.**
- **Ejemplos:**
  - **No más de una sentencia en cada linea de código.**
  - **Un espacio a cada lado de un operador binario.**
  - **Un espacio despues de una coma pero no antes.**
  - **Un espacio despues de cada palabra clave.**
  - **Ningún espacio antes de punto y coma.**
  - **Ningún espacio entre un operador unario y su operando.**
  - **Distinguir los bloques lógicos de códigos con tabulaciones.**
  - **Una tabulación en la segunda y siguientes líneas cuando una sentencia ocupa más de una línea.**

Pseudocódigo

- **Es una representación informal de lo que hace el programa**
- **Nos ayuda a planificar algoritmos complicados**
- **Se escribe en lenguaje natural (español)**
- **Una vez escrito solo nos queda ir traduciendo sus partes a código C#**
- **Podemos empezar por escribir las tareas principales y luego ir refinando cada una de ellas con un mayor detalle**

Pseudocódigo (II)

```
string Numero1, Numero2;
int a, b, c;

Console.WriteLine(
   “Introduzca dos números: “);

Numero1 = Console.ReadLine();
Numero2 = Console.ReadLine();

a = Convert.ToInt32(Numero1);
b = Convert.ToInt32(Numero2);

c = a + b;

Console.WriteLine(c);
```

Calcular la suma de dos números

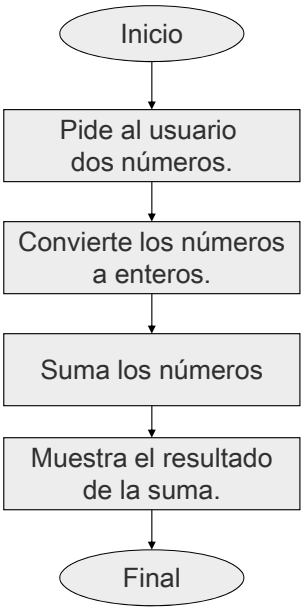
Pedir al usuario dos números.
Leer los números del teclado.
Convertirlos a enteros.
Sumar los dos números.
Mostrar el resultado de la suma

Diagramas de flujo

- **Son una representación de la sucesión de acciones en un programa**
- **Las acciones se representan con los siguientes símbolos**
  - **Un ovalo para indicar el principio o fin**
  - **Un rectangulo para indicar una acción**
  - **Un rombo para indicar una decisión**
- **Las acciones se unen entre si mediante flechas que indican el posible flujo del programa.**

Diagramas de flujo (II)

```
string Numero1, Numero2;
int a, b, c;

Console.WriteLine(
   "Introduzca dos números: ");

Numero1 = Console.ReadLine();
Numero2 = Console.ReadLine();

a = Convert.ToInt32(Numero1);
b = Convert.ToInt32(Numero2);

c = a + b;

Console.WriteLine(c);
```



Estructuras de control

- **Por norma general en un programa se ejecuta una sentencia y después la que viene a continuación.**
- **Las estructuras de control nos permiten alterar este orden secuencial de ejecución.**
  - **Estructuras de selección.**
    - **if / else**
    - **switch**
  - **Estructuras de repetición.**
    - **while**
    - **do / while**
    - **for**

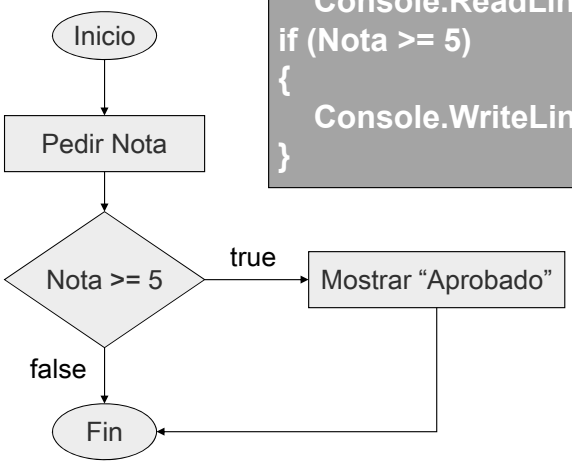
La estructura if

```
if (condicion)
{
    ...
}
```

- ***condición*** **puede ser cualquier expresión que devuelva un tipo bool (true ó false)**
- **Si *condición* es true se ejecuta lo que pongamos entre las llaves.**
- **Si *condición* es false no se ejecuta.**

La estructura if (II)

```
Console.WriteLine("Introduzca nota: ");
int Nota = Convert.ToInt32(
    Console.ReadLine( ));
if (Nota >= 5)
{
    Console.WriteLine("Aprobado");
}
```




```
Introduzca nota:
7
Aprobado
_
```

```
Introduzca nota:
4
_
```

La estructura if / else

```
if (condicion)
{
   ...
}
else
{
   ...
}
```

- opcionalmente podemos añadir un *else*.
- si la *condición* es *false* se ejecuta lo que va entre llaves a continuación del *else*.

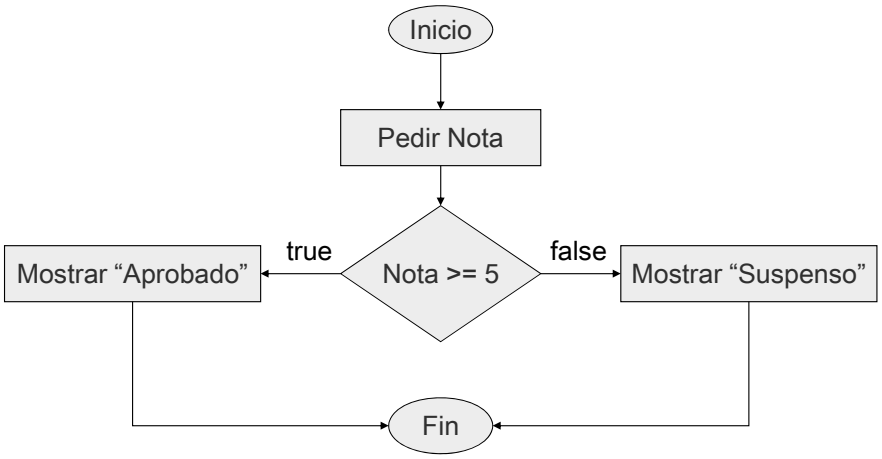
La estructura if / else (II)

```
Console.WriteLine("Introduzca nota: ");
int Nota = Convert.ToInt32(
   Console.ReadLine( ));
if (Nota >= 5)
{
   Console.WriteLine("Aprobado");
}
else
{
   Console.WriteLine("Suspenso");
}
```

```
Introduzca nota:
7
Aprobado
_
```

```
Introduzca nota:
4
Suspenso
_
```

La estructura if / else (III)



‘ifs’ anidados

```
Console.WriteLine(“Introduzca nota: “);
int Nota = Convert.ToInt32(Console.ReadLine());
if (Nota < 5)
{
      Console.WriteLine(“Suspenso”);
}
else
{
      if (Nota < 7)
      {
            Console.WriteLine(“Aprobado”);
      }
      else
      {
            if (Nota < 9)
            {
                  Console.WriteLine(“Notable”);
            }
            else
            {
                  Console.WriteLine(“Sobresaliente”);
            }
      }
}
```

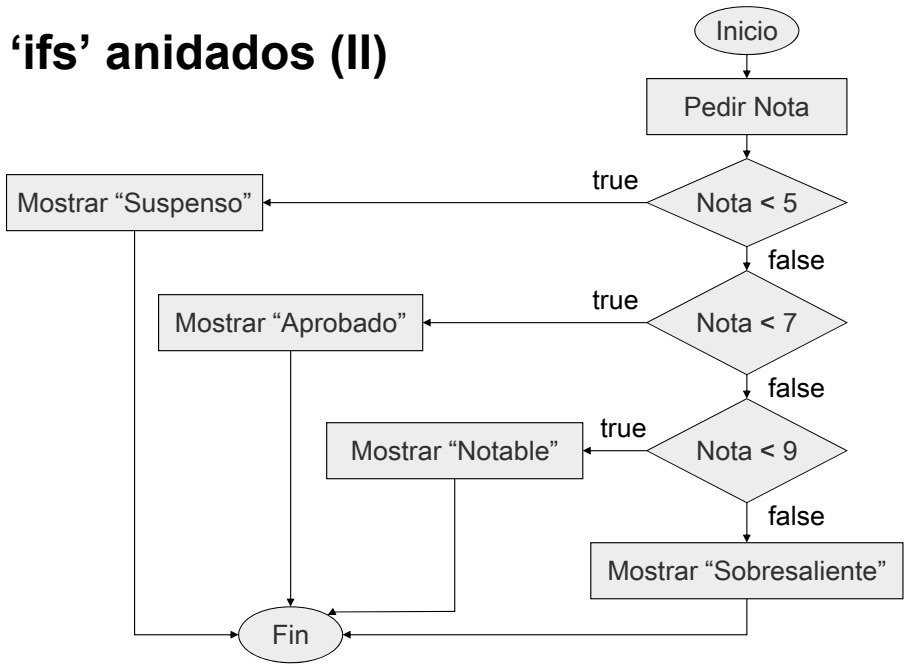
```
Introduzca nota:
4
Suspenso
_
```

```
Introduzca nota:
5
Aprobado
_
```

```
Introduzca nota:
8
Notable
_
```

```
Introduzca nota:
9
Sobresaliente
_
```

‘ifs’ anidados (II)



Operador condicional (?:)

(*condicion* ? *ValorSiCierta* : *ValorSiFalsa*)

- **Si condicion es true devuelve un valor y si es false devuelve el otro.**
- **También se pueden anidar una dentro de otra.**

Ejemplo:

```
Console.WriteLine(“Introduzca nota: “);
int Nota = Convert.ToInt32(
   Console.ReadLine( ));
Console.WriteLine(Nota >= 5 ?
   “Aprobado” : “Suspenso”);
```

```
Introduzca nota:
7
Aprobado
_
```

```
Introduzca nota:
4
Suspenso
_
```

La estructura while

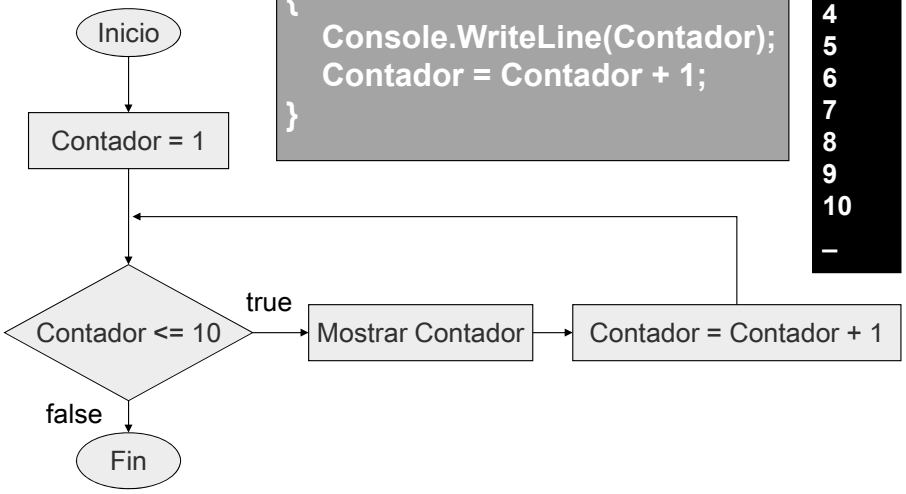
```
while (condicion)
{
    ...
}
```

- ***condición*** **puede ser cualquier expresión que devuelva un tipo bool (*true* ó *false*)**
- **Lo que va entre llaves se ejecutará de forma repetitiva mientras que *condición* sea *true*.**
- **Dentro de la llaves tiene que haber algo que modifique el resultado de la *condición* o sino se repetiría indefinidamente.**
- **Si la primera vez *condición* es *false*, lo que va entre llaves no se ejecuta ninguna vez.**

La estructura while (II)

```
int Contador = 1;
while ( Contador <= 10 )
{
    Console.WriteLine(Contador);
    Contador = Contador + 1;
}
```

```
1
2
3
4
5
6
7
8
9
10
_
```



Algoritmos

Repetición controlada por un contador

- **Problema**

  Pedir al usuario 10 notas y calcular la media

- **Pseudocodigo**

Hacer el total igual a 0
Hacer el contador igual a 0

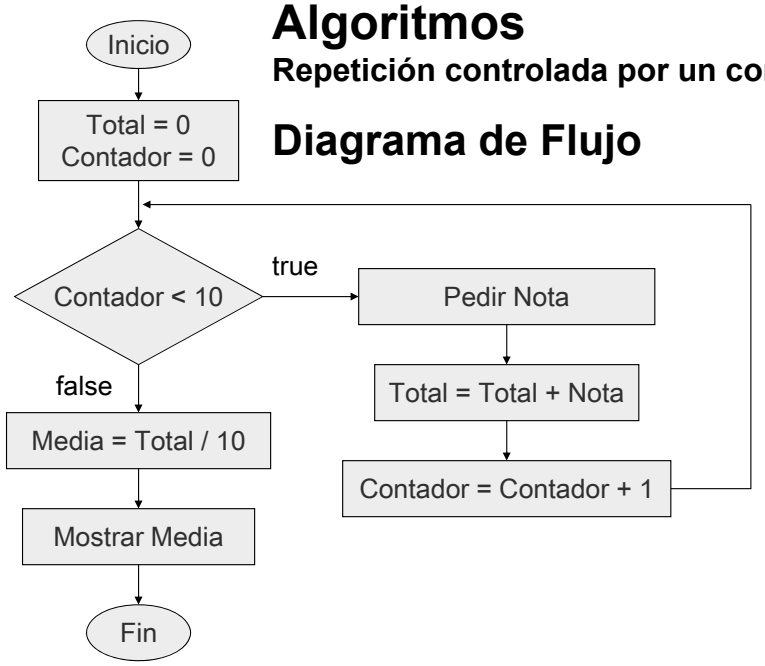
Mientras que el contador sea menor que 10
    Pedir la siguiente nota
    Sumar la nota al total
    Sumar uno al contador

Hacer la media igual al total dividido entre 10
Mostrar la media

Algoritmos

Repetición controlada por un contador (II)

Diagrama de Flujo



Algoritmos

Repetición controlada por un contador (III)

Código

```
int Contador = 0;
double Nota, Total = 0, Media;

while (Contador < 10)
{
   Console.WriteLine("Nota " + (Contador + 1) + ": ");
   Nota = Convert.ToDouble(Console.ReadLine());
   Total += Nota;
   Contador++;
}
Media = Total / 10;
Console.WriteLine("La media es " + Media);
```

Algoritmos

Repetición controlada por un valor centinela

- **Problema**

  Pedir varias notas y calcular la media

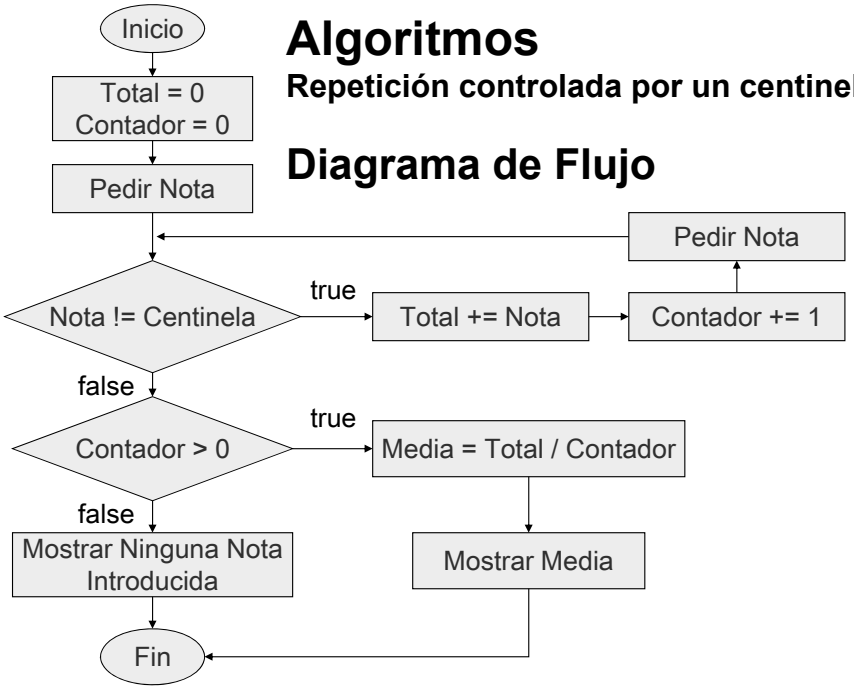
- **Pseudocódigo**

Hacer el total igual a 0
Hacer el contador igual a 0
Pedir la primera nota (puede ser el valor centinela)
Mientras que el usuario no introduzca el valor centinela
    Sumar la nota al total
    Sumar uno al contador
    Pedir la siguiente nota
Si el contador es mayor que cero
    Hacer la media igual al total dividido entre el contador
    Mostrar la media
sino
    Mostrar que no se han introducido notas

Algoritmos

Repetición controlada por un centinela (II)

Diagrama de Flujo



Algoritmos

Repetición controlada por un valor centinela (III)

```
int Contador = 0;
double Nota, Total = 0, Media;

Console.Write("Introduzca nota (-1 Fin): ");
Nota = Convert.ToDouble(Console.ReadLine());

while (Nota != -1)
{
   Total += Nota;
   Contador++;

   Console.Write("Introduzca nota (-1 Fin): ");
   Nota = Convert.ToDouble(Console.ReadLine());
}
```

Algoritmos

Repetición controlada por un valor centinela (IV)

```
// Continuación

if (Contador > 0)
{
   Media = Total / Contador;
   Console.WriteLine("La media es " + Media);
}
else
{
   Console.WriteLine(
      "No se ha introducido ninguna nota.");
}
```

Algoritmos

Repetición con condicionales dentro

- **Problema**

  Pedir varias notas y mostrar el número de suspensos y aprobados

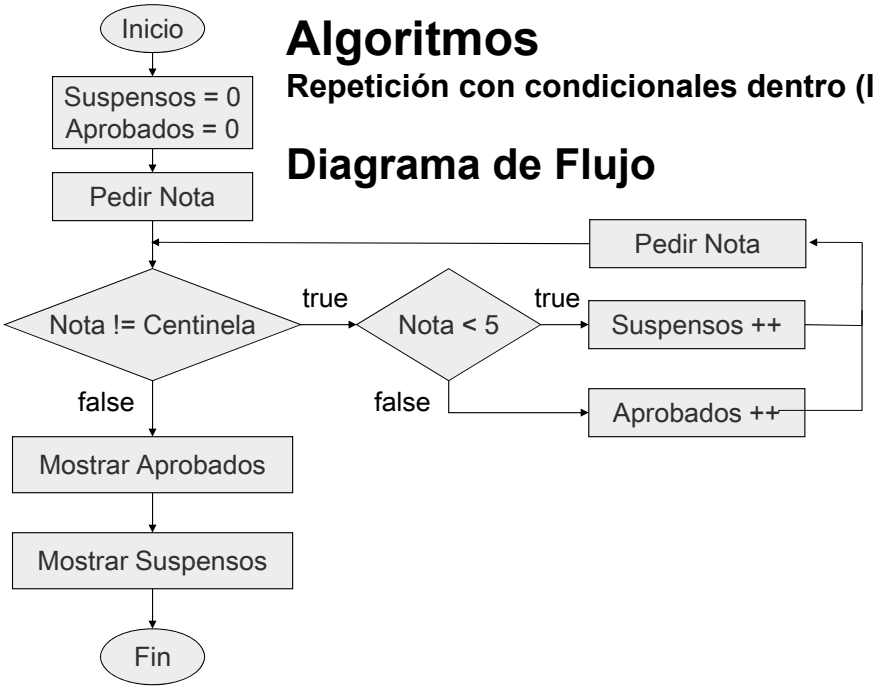
- **Pseudocódigo**

  Hacer Suspensos y Aprobados igual a 0
  Pedir la primera nota (puede ser el valor centinela)
  Mientras que el usuario no introduzca el valor centinela
  Si la Nota es menor de 5
  Sumar uno a Suspensos
  sino
  Sumar uno a Aprobados
  Pedir la siguiente nota
  Mostrar Aprobados y Suspensos

Algoritmos

Repetición con condicionales dentro (II)

Diagrama de Flujo



Algoritmos

Repetición con condicionales dentro (III)

```
int Suspensos = 0, Aprobados = 0;
Console.Write("Introduzca nota (-1 Fin): ");
double Nota = Convert.ToDouble(Console.ReadLine( ));
while (Nota != -1)
{
   if (Nota < 5)
   {
      Suspensos++;
   }
   else
   {
      Aprobados++;
   }
   Console.Write("Introduzca nota (-1 Fin): ");
   Nota = Convert.ToDouble(Console.ReadLine());
}
Console.WriteLine("Aprobados: " + Aprobados);
Console.WriteLine("Suspensos: " + Suspensos);
```

Operadores lógicos

operador	*Nombre*	*Utilización*
&&	Y lógico	expresion1 && expresion2
||	O lógico	expresion1 || expresion2

- ***expresion1*** **y** ***expresion2*** **deben ser de tipo bool. Estos operadores a su vez devuelven un valor booleano.**
- **Operador && - Si** ***expresion1*** **es** ***true*** **y** ***expresion2*** **es** ***true*** **devuelve** ***true***, **sino devuelve** ***false***.
- **Operador || - Si** ***expresion1*** **es** ***true*** **o** ***expresion2*** **es** ***true*** **devuelve** ***true***, **sino devuelve** ***false***. **(Si ambas son** ***true*** **evidentemente devuelve también** ***true***).

Operadores lógicos (II)

Ejemplo:

```
Console.WriteLine("Introduzca nota: ");
int Nota = Convert.ToInt32(
   Console.ReadLine());

if (Nota < 0 || Nota > 10)
{
   Console.WriteLine("Nota incorrecta");
}

if (Nota >= 7 && Nota < 9)
{
   Console.WriteLine("Notable");
}
```

```
Introduzca nota:
8
Notable
_
```

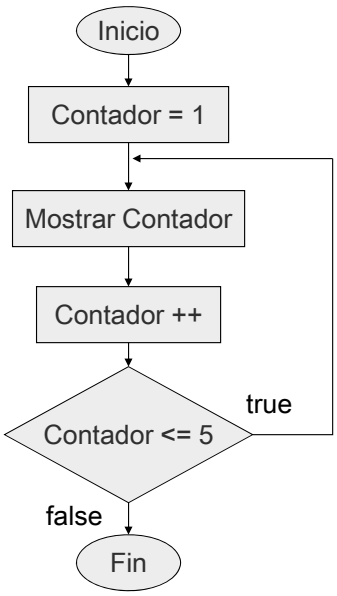
```
Introduzca nota:
12
Nota incorrecta
_
```

La estructura do / while

```
do
{
   ...
} while (condicion);
```

- ***condición*** **puede ser cualquier expresión que devuelva un tipo bool (*true* ó *false*)**
- **Lo que va entre llaves se ejecuta SIEMPRE una primera vez.**
- **Lo que va entre llaves se volverá a ejecutar mientras que la *condición* sea *true*.**
- **Dentro de la llaves tiene que haber algo que modifique el resultado de la *condición* o sino se repetiría indefinidamente.**
- **TERMINA EN PUNTO Y COMA.**

La estructura do / while (II)




```
int Contador = 1;

do
{
   Console.WriteLine(Contador);
   Contador ++;
} while ( Contador <= 5 );
```

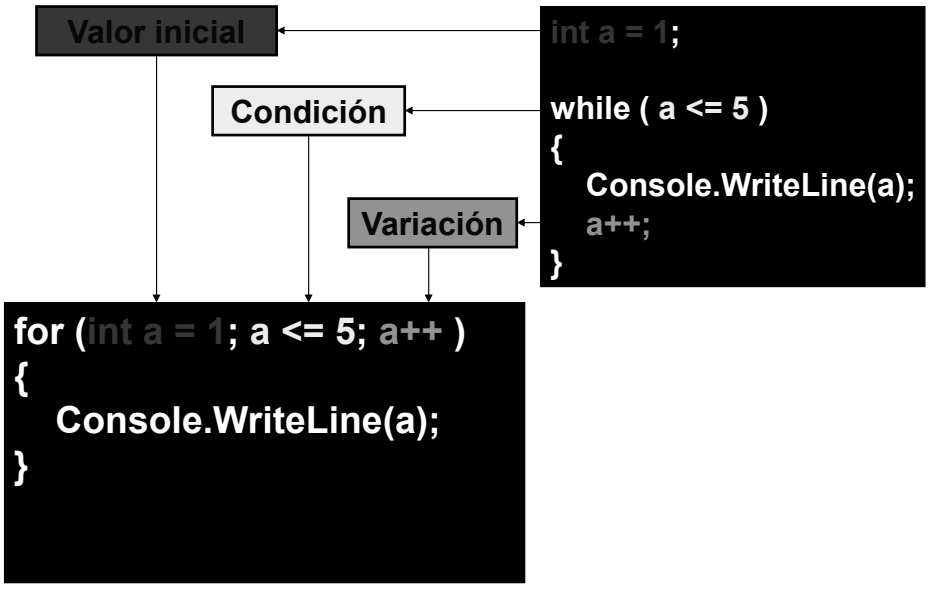
```
1
2
3
4
5
_
```

La estructura for

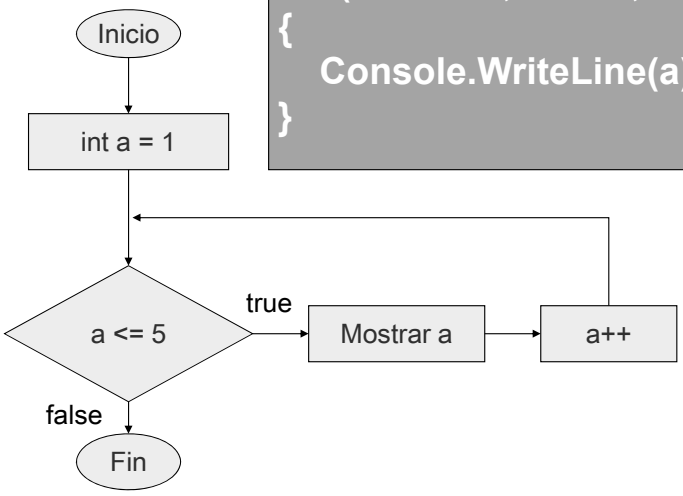
```
for ( Exp1; Exp2; Exp3 )
{
    ...
}
```

- La estructura for se utiliza para repetir lo que va entre llaves según *tres expresiones*.
- *Exp1* – Inicializa la variable de control al principio.
- *Exp2* – Condición en función de la variable de control para continuar repitiendo.
- *Exp3* – Modificación de la variable de control después de cada repetición .
- El separador de las tres expresiones es el punto y coma.

La estructura for (II)



La estructura for (III)




```
for ( int a = 1; a <= 5; a++ )
{
   Console.WriteLine(a);
}
```

```
1
2
3
4
5
_
```

La estructura for (IV). Bucles Anidados.

```
for ( int a = 1; a <= 5; a++ )
{
      for ( int b = 1; b <= 5; b++)
      {
            if ( b <= a )
            {
                  Console.Write("O");
            }
      }
      Console.Write("\n");
}
Console.ReadLine();
```

```
O
OO
OOO
OOOO
OOOOO
_
```

La estructura switch / case

- El tipo de la *Expresion* debe de ser entero, string o char.
- En función del valor que tenga *Expresion* ejecuta el código que va a continuación del *case* que tiene ese valor.
- Después de incluir código en un case deberemos poner la sentencia break.
- Podemos incluir opcionalmente un *default* que se ejecuta si no se ejecutaron ninguno de los anteriores.

```
switch ( Expresion )
{
   case Valor1:
      ...
      break;
   case Valor2:
      ...
      break;
   ...

   default:
      ...
      break;
}
```

La estructura switch / case (II)

```
Console.WriteLine("Introduzca día: ");
string Dia = Console.ReadLine();
Dia = Dia.ToUpper( );

switch ( Dia )
{
   case "SABADO":
      Console.WriteLine("Salir de copas.");
      break;
   case "DOMINGO":
      Console.WriteLine("Ir al cine.");
      break;
   default:
      Console.WriteLine("Ir a clase");
      break;
}
```

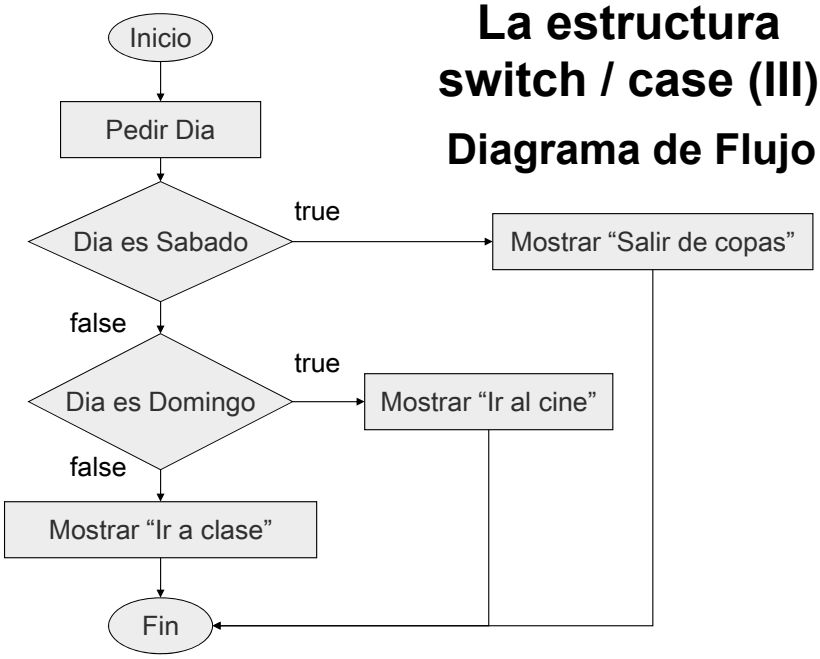
```
Introduzca día ?
Lunes
Ir a clase
_
```

```
Introduzca día ?
Sabado
Salir de copas
_
```

```
Introduzca día ?
Domingo
Ir al cine
_
```

La estructura switch / case (III)

Diagrama de Flujo



Bucles. break y continue

- ***Bucle*: sentencias que se repiten.**
- ***break* y *continue* alteran el orden normal de ejecución de un bucle.**
- ***break* – Termina la ejecución del bucle en ese momento de forma definitiva.**
- ***continue* – Finaliza la ejecución del ciclo en ese momento y vuelve al principio del bucle para repetirlo.**

Bucles. break y continue (II)

break

```
for ( int a = 1; a <= 5; a++ )
{
    if (a == 3)
        break;
    Console.WriteLine(a);
}
```

```
1
2
_
```

continue

```
for ( int a = 1; a <= 5; a++ )
{
    if (a == 3)
        continue;
    Console.WriteLine(a);
}
```

```
1
2
4
5
_
```

Bucles. break y continue (III)

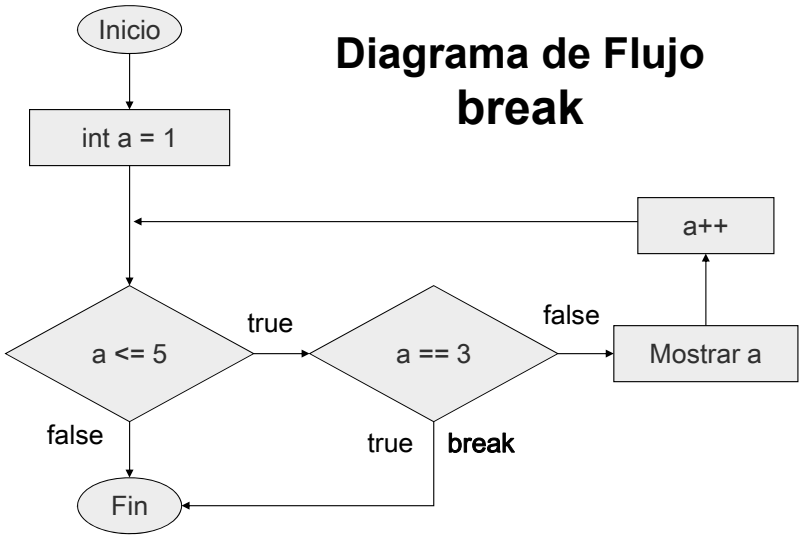


Diagrama de Flujo
break

Bucles. break y continue (IV)

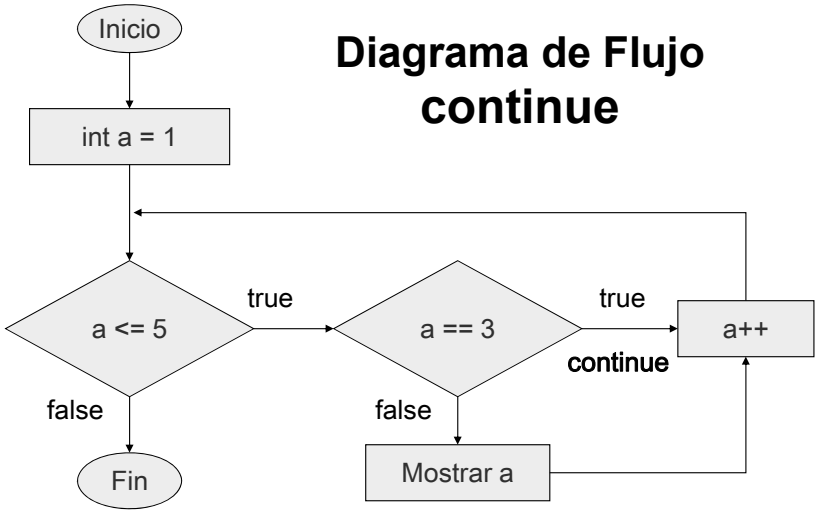


Diagrama de Flujo continue

Clase Random

- **Se encuentra en el namespace System**
- **Nos permite generar números aleatorios.**
- ***objetoRandom*.Next()**

  Devuelve un número aleatorio entre 0 e int.MaxValue (int.MaxValue = 2,147,483,647)
- **objetoRandom.Next(x)**

  Devuelve un número aleatorio entre 0 y x (sin incluir x).
- **objetoRandom.Next(x, y)**

  Devuelve un número aleatorio entre x e y (sin incluir y).

Clase Random (II)

EJEMPLO: Mostrar 20 números aleatorios del 1 al 6 colocando 5 en cada fila.

```
string resultado = “”;
int valor;
Random creadorRandom = new Random();
for( int i = 1; i <= 20; i++ )
{
   valor = creadorRandom.Next(1, 7);
   resultado += valor + “ “;
   if ( i % 5 == 0 )
      resultado += “\n”;
}
Console.WriteLine(resultado);
```

```
1 2 5 2 1
2 3 6 2 1
6 3 2 5 1
1 1 5 5 6

_
```