

- Unidad 3

# ESTRUCTURAS DE CONTROL

(Capítulo 4 bibliografía)

Operadores Lógicos - Instrucción de selección y características.



» Ventre, Luis O.



## Planteo

### Problema, como visualizarlo?

En los lenguajes estructurados **la ejecución secuencial** de las instrucciones cumple un **orden** determinado – FLUJO DE CONTROL.

En la resolución algorítmica de problemas, frente a **diferentes “condiciones”** es necesario que nuestro sistema responda con **trazas diferentes**.

### Solución

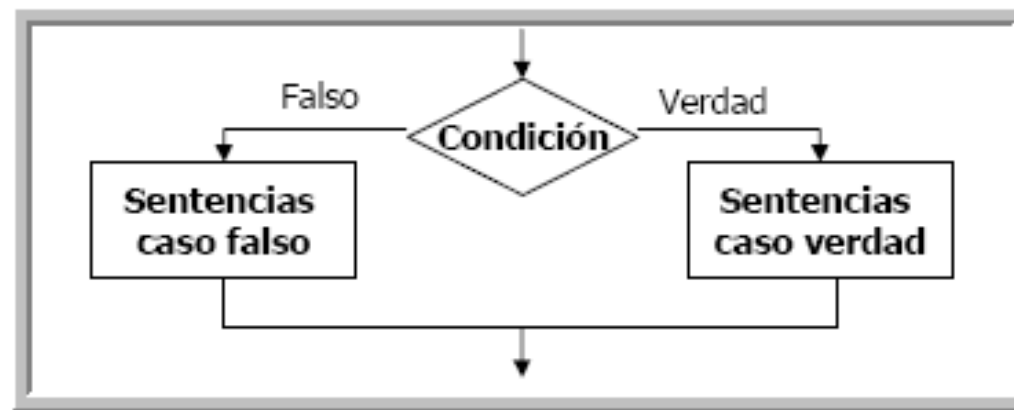
Disponer de una **estructura** que nos permita **modificar el flujo de control** de nuestro programa.

La **estructura de decisión**, o selección, es la estructura que nos permitirá de acuerdo a diferentes situaciones **modificar la secuencia de instrucciones a ejecutar**.

## Como implementa el lenguaje la solución?

El lenguaje, brinda una instrucción, la cual esta basada en una **condición o expresión**, que se evalúa y dependiendo de su resultado se bifurcan las posibles secuencias de ejecución de código.

Su diagrama de flujo equivalente es:





## Elementos de la estructura

### Como esta compuesta la estructura de selección?

Los componentes de la estructura en la versión simple de su pseudocódigo son:

```
if (condición)
    instrucc. ejecutada si condición es verdadera;
else
    instrucc. ejecutada si condición es falsa;
```

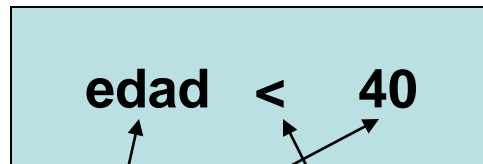
Cuando un programa en ejecución encuentra la palabra reservada IF, se **evalúa la “condición”**, si su resultado es un **numero positivo o negativo diferente** de 0, es interpretada como **verdadera y se ejecuta la instrucción a continuación**, si evalúa a 0, es considerada falsa y se ejecuta la instrucción correspondiente al bloque else.



## Elementos: La condición

**CONDICION:** Cualquier expresión valida para el lenguaje (incluso una asignación).

Expresiones mas usadas: **EXPRESIONES RELACIONALES**



**operandos**      **operador relacional**

Además cada **operando** puede ser una **variable** o una **constante**;

Ver tabla de operadores

### Operadores relacionales:

C:

<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que

==	Igual que
!=	Distinto que



## Elementos: La condición

Las **expresiones relacionales** también son llamadas **condiciones**. Estas expresiones son evaluadas y producen un resultado numérico; si la expresión es verdadera evalúan a “1” caso contrario evalúan a “0”.

La verificación de esto mediante código sería:

```
cout<<“El valor de 3 < 4 es:”<< (3 < 4) <<endl;  
cout<<“El valor de 2.0 > 3.0 es:” << (2.0 > 3.0)<<endl;
```

La salida producida será.

```
El valor de 3 < 4 es: 1  
El valor de 2.0 > 3.0 es: 0
```

....y si coloco la instrucción `cout<<true?`



## Elementos: La condición

### Operadores Lógicos:

Además de usar expresiones relacionales simples, pueden crearse condiciones mas complejas usando los operadores lógicos AND “y”, OR “o” y NOT “no”.

Cuando el operador AND &&, se usa con dos expresiones simples, la condición es verdadera solo si **ambas expresiones individuales son verdaderas por si mismas**

Operadores lógicos:		
Pseudocódigo:	C:	
y	&&	Conjunción
o		Disyunción
no	!	Negación

Ej: (voltaje > 48) && (miliamperes < 10)

Precedencia de operadores relacionales es mayor. Se puede sacar ().



## Criterios de Selección

El operador lógico **OR**, también se aplica entre dos expresiones; la condición se satisface si **cualquiera de las dos expresiones O AMBAS son verdaderas**:

Ej:      double voltaje=47, miliamperes=10, valor=0.0;  
          cout<<(voltaje > 48) || (miliamperes < 10) || valor;



Precedencia de operadores relacionales es mayor. Se puede sacar ().

El operador lógico **NOT es unitario**, se usa para cambiar UNA expresión a su estado opuesto. Si la expresión tiene valor diferente de 0, o sea verdadera; !expresión produce valor de 0, o sea falso.

Ej: .... *int edad = 26;*  
          *(edad > 40) evalúa a 0 o sea falso*  
          *y !(edad > 40) evalúa ??*





## Criterios de Selección

### Precedencia y asociatividad:

Los operadores relacionales y lógicos también tienen una precedencia y asociatividad como los operadores aritméticos. La tabla siguiente enumera la precedencia

Operador	Asociatividad
! unitario - ++ --	derecha a izquierda
* / %	izquierda a derecha
+ -	izquierda a derecha
< <= > >=	izquierda a derecha
== !=	izquierda a derecha
&&	izquierda a derecha
	izquierda a derecha
= += -= *= /=	derecha a izquierda

### Algunos Ejemplos:

Expresión	Expresión equivalente	Valor	Interpretación
$i + 2 == k - 1$	$(i + 2) == (k - 1)$	0	falso
$3 * i - j < 22$	$(3 * i) - j < 22$	1	verdadero
$i + 2 * j > k$	$(i + (2 * j)) > k$	1	verdadero
$k + 3 <= -j + 3 * i$	$(k + 3) <= ((-j) + (3 * i))$	0	falso
$'a' + 1 == 'b'$	$( 'a' + 1 ) == 'b'$	1	verdadero
$key - 1 > 'p'$	$(key - 1) > 'p'$	0	falso
$key + 1 == 'n'$	$(key + 1) == 'n'$	1	verdadero
$25 >= x + 1.0$	$25 >= (x + 1.0)$	1	verdadero



## Criterios de Selección

### Precedencia y asociatividad:

$(6 * 3 == 36 / 2) \parallel (13 < 3 * 3 + 4) \&\& !(6 - 2 < 5)$

$(6 * 3 == 36 / 2) \parallel (13 < 3 * 3 + 4) \&\& !(6 - 2 < 5)$

$(18 == 18) \parallel (13 < 9 + 4) \&\& !(4 < 5)$

$1 \parallel (13 < 13) \&\& !1$

$1 \parallel 0 \&\& 0$

$1 \parallel 0$

**1**

Operador	Asociatividad
! unitario - ++ --	derecha a izquierda
* / %	izquierda a derecha
+ -	izquierda a derecha
< <= > >=	izquierda a derecha
== !=	izquierda a derecha
&&	izquierda a derecha
	izquierda a derecha
= += -= *= /=	derecha a izquierda



## La instrucción IF ELSE

*La instrucción IF, **dirige a la computadora** a ejecutar una serie de una o mas instrucciones basadas en el resultado de una **comparación**.*

*Primero se evalúa la expresión, y luego se ejecuta la instrucción correspondiente.*

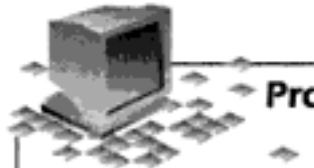
*Observemos que la **expresión evaluada va entre paréntesis**.*

***Por claridad** se escribe la instrucción de la siguiente manera:*

```
if (expresion) ← no va punto y coma aqui
    instruccion1;
else ← no va punto y coma aqui
    instruccion2;
```

***Es de importancia notar que solo llevan punto y coma las líneas de las instrucciones!!***

## La instrucción IF ELSE – Un ejemplo



Programa 4.1

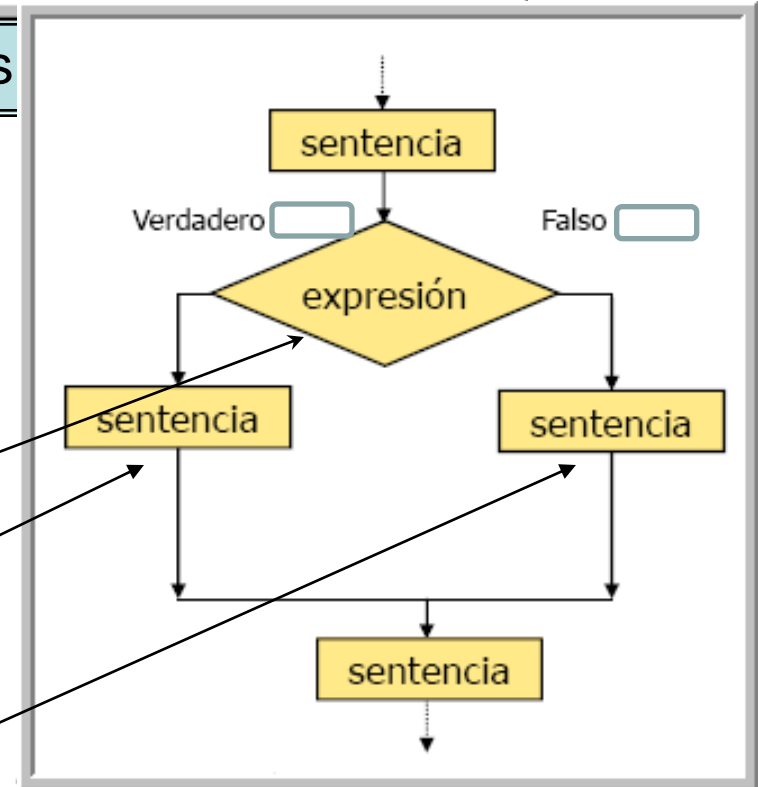
```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double radio;

    cout << "Por favor introduzca el radio: ";
    cin >> radio;

    if (radio < 0.0 )
        cout << "Un radio negativo es invalido" << endl;
    else
        cout << "El area de este circulo es " << 3.1416 * pow(radio,2) << endl;
}
```

El resultado de dos



```
Por favor introduzca el radio: 2.5
El area de este circulo es 19.635
Presione una tecla para continuar . . .
```



## La instrucción IF ELSE

- **Instrucciones Compuestas:**

En las secciones de instrucción de IF o ELSE, solo se permite una instrucción. En caso de ser necesario realizar mas de una instrucción, es posible utilizar una ***instrucción compuesta***. Esta es una ***secuencia de instrucciones individuales contenidas entre llaves***.

```
if (expresion)
{
    instruccion1; // pueden colocarse dentro de las llaves
    instruccion2; // tantas instrucciones como sean
                  // necesarias
    instruccion3; // cada instruccion debe terminar con un;
}
else
{
    instruccion4;
    instruccion5;
    .
    .
    instruccion_n;
}
```

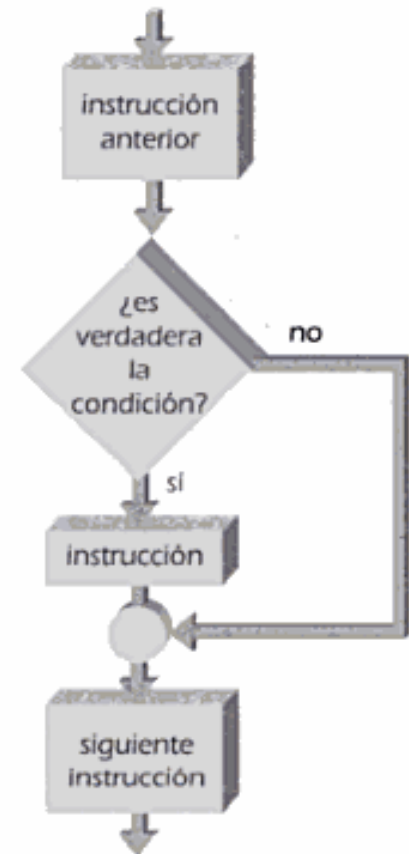


## La instrucción IF ELSE

- **IF unidireccional:**
- Una modificación útil, implica **omitir por completo la parte ELSE** de la instrucción if else; quedando una forma abreviada:

```
if (expresion)  
    instrucción;
```

La instrucción que sigue a if, solo se ejecuta si expresión tiene un valor diferente a 0.





## INSTRUCCIONES IF ANIDADAS

- **Instrucciones If anidadas:**

Como se ha visto una instrucción if else, puede contener instrucciones simples o compuestas; puede incluso usarse otra instrucción if else dentro del cuerpo de “if” o de “else” de una instrucción: La instrucción de una o mas instrucciones if se llama **instrucción if anidada**.

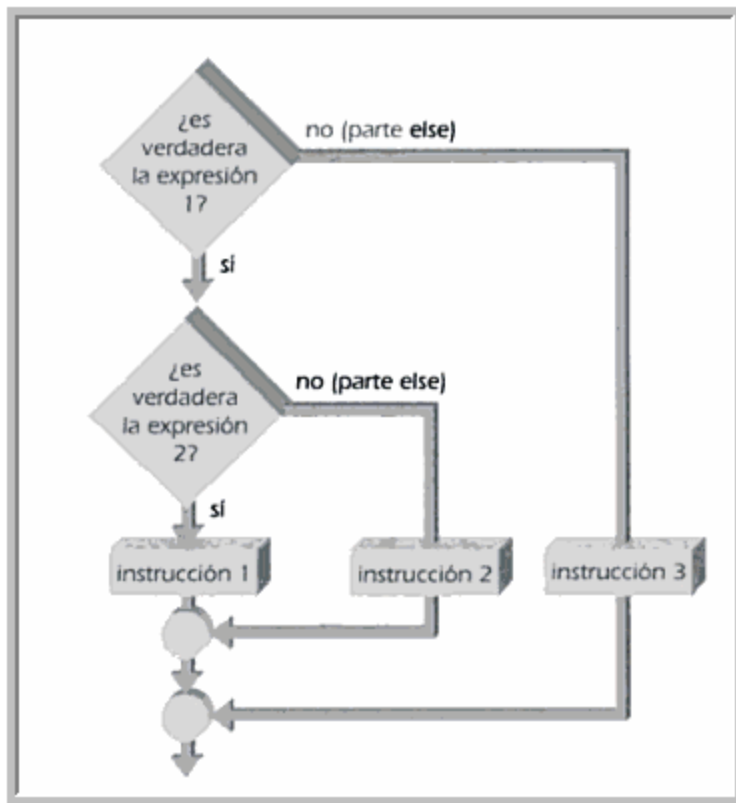
**Máximo cuidado, con las llaves que rodean al if unidireccional, ya que el lenguaje asocia el “else” al if mas “cercano”.**

```
if (horas < 9)
{
    if (distancia > 500)
        cout << "oprime";
}
else
    cout << "suelte";
```

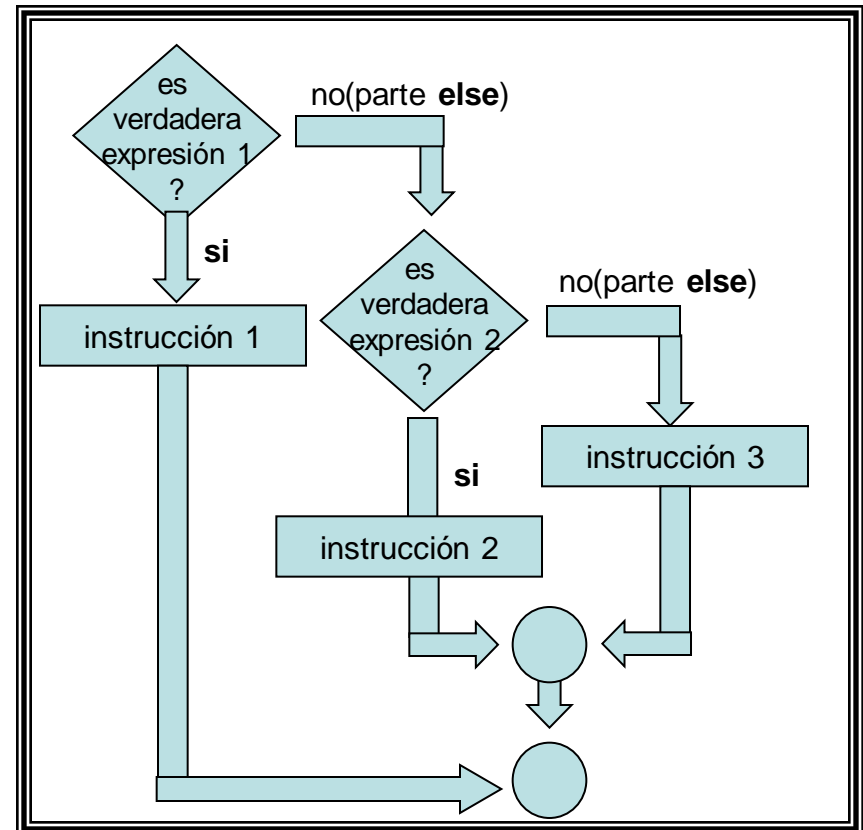
```
if (horas < 9)
    if (distancia > 500)
        cout << "oprime";
    else
        cout << "suelte";
```

## INSTRUCCIONES IF ANIDADAS

### IF anidada en sección IF



### IF anidada en sección ELSE.







## INSTRUCCIONES IF ANIDADAS

- **Cadena IF ELSE:**

En general la anidación ilustrada en la **figura izquierda** anterior (sección IF) **es confusa**. Sin embargo el anidamiento en **sección ELSE** es una construcción muy útil.

```
if (expresion_1)
    instruccion1;
else
    if (expresion_2)
        instruccion2;
    else
        instruccion3;
```

```
if (expresion_1)
    instruccion1;
else if (expresion_2)
    instruccion2;
else if (expresion_3)
    instruccion3;
    .
    .
    .
else if (expresion_n)
    instruccion_n;
else
    ultima_instruccion;
```

Como los espacios en blanco son ignorados por el compilador puede reemplazarse por la denominada formalmente **CADENA IF ELSE**



## INSTRUCCIONES IF ANIDADAS

En esta construcción, cada condición se evalúa en orden, y si **CUALQUIER CONDICION ES VERDADERA** se ejecuta la instrucción correspondiente y el **RESTO DE LA CADENA SE TERMINA**.

La instrucción asociada con el **ELSE final**, solo se ejecuta si ninguna de las condiciones fue verdadera. Seria como un caso por omisión; *muy útil para detectar una condición errónea*.

```
if (expresion_1)
    instruccion1;
else if (expresion_2)
    instruccion2;
else if (expresion_3)
    instruccion3;
    .
    .
    .
else if (expresion_n)
    instruccion_n;
else
    ultima_instruccion;
```



## Detalles en la instrucción IF ELSE

Generalmente se utilizan expresiones relacionales para verificar, pero también puede colocarse una instrucción como:

```
if (num)  
    cout<<"LOTERIA!!";  
else  
    cout<<"PERDISTE!!";
```

Esta instrucción es válida, ya que num, es una expresión válida, el mensaje LOTERIA!! es desplegado si num tiene cualquier valor diferente de "0" y el mensaje PERDISTE!! es desplegado si num tiene un valor de "0".



## Detalles en la instrucción IF ELSE

- **Problemas asociados con la instrucción IF ELSE:**

**USAR EL OPERADOR DE ASIGNACION “=” EN LUGAR DEL  
OPERADOR RELACIONAL “==”.**

Ej: Supongamos que se pretende usar la expresión a continuación con motivo de imprimir el mensaje cuando edad sea igual a 40.

```
if (edad==40)  
cout<<“!Feliz 4ta DECADA!”;
```

Pero por error se colocó:

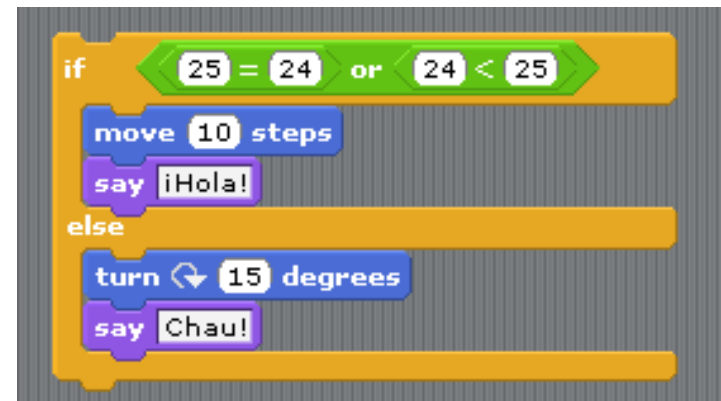
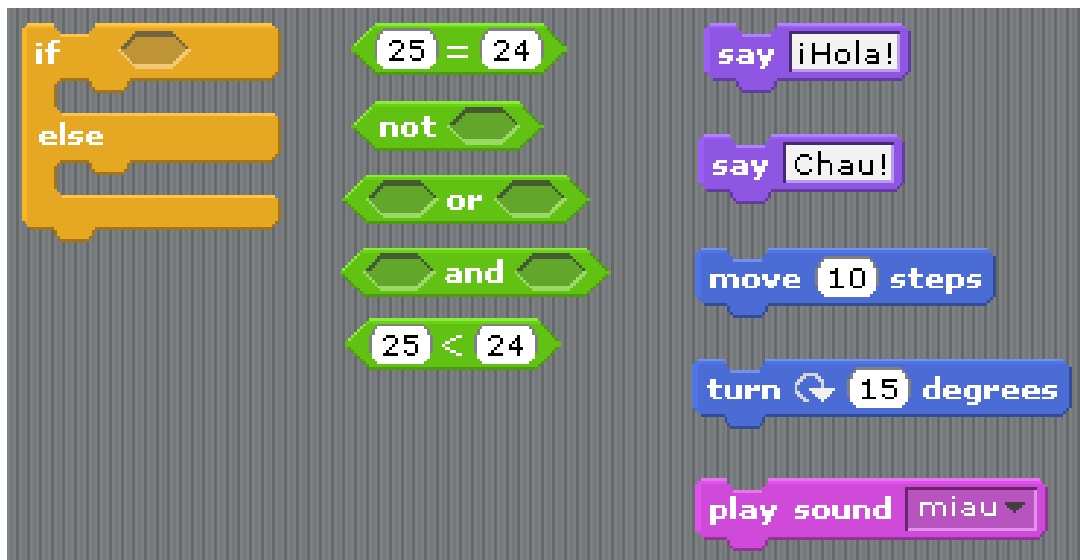
```
if (edad=40)  
cout<<“!Feliz 4ta DECADA!”;
```

Cuando se imprimirá el mensaje y porque?

## Representación grafica

### Como podría visualizarse la instrucción?

Scratch<sup>1</sup>. Lenguaje de programación visual, desarrollado en MIT, para la enseñanza de la informática a partir de edades entre 6 y 8 años hasta 16 años. Se pueden desarrollar fácilmente historias y proyectos y compartirlos a nivel mundial






## La instrucción IF ELSE

- Un ejemplo:

El siguiente programa verifica si el valor de tipo\_temp es f, con esto determina si ejecutar la parte de IF o la parte de ELSE.

 Programa 4.2

```
#include <iostream>
#include <iomanip>
using namespace std;

// un programa para conversion de tempera
int main()
{
    char tipo_temp;
    double temp, fahrenheit, celsius;

    cout << "Introduzca la temperatura que
    cin >> temp;
    cout << "Introduzca una f si la temp
    cout << "\n o una c si la temperatu
    cin >> tipo_temp;

    // establecer los formatos de sal
    cout << setiosflags(ios::fixed)
    << setiosflags(ios::showpoint)
    << setprecision(2);
```

```
    if (tipo_temp == 'f')
    {
        celsius = (5.0 / 9.0) * (temp - 32.0);
        cout << "\nLa temperatura Celsius equivalente es "
        << celsius << endl;
    }
    else
    {
        fahrenheit = (9.0 / 5.0) * temp + 32.0;
        cout << "\nLa temperatura Fahrenheit equivalente es "
        << fahrenheit << endl;
    }

    return 0;
}
```

```
Introduzca la temperatura que se va a convertir: 212
Introduzca una f si la temperatura es en grados Fahrenheit
o una c si la temperatura es en grados Celsius: f

La temperatura Celsius equivalente es 100.00
Presione una tecla para continuar . . .
```



## Ejemplo cadena IF ELSE

**Suponga que debe realizar un programa que genera la salida de una unidad convertidora digital usando la siguiente relación entrada salida.**

Peso de entrada	Lectura de salida
mayor que o igual a 90 lbs	1111
menor que 90 lbs pero mayor que 80 lbs	1110
menor que 80 lbs pero mayor que o igual a 70 lbs	1101
menor que 70 lbs pero mayor que o igual a 60 lbs	1100
menor que 60 lbs	1011

*Este ejemplo aprovecha el hecho de que la cadena se detiene una vez que encuentra una expresión verdadera.*



## Ejemplo cadena IF ELSE

```
#include <iostream>
using namespace std;

int main()
{
    int digout;
    double inlbs;

    cout << "Introduzca el peso: ";
    cin >> inlbs;

    if (inlbs >= 90)
        digout = 1111;
    else if (inlbs >= 80)
        digout = 1110;
    else if (inlbs >= 70)
        digout = 1101;
    else if (inlbs >= 60)
        digout = 1100;
    else
        digout = 1011;

    cout << "La salida digital es " << digout << endl;

    return 0;
}
```

```
Introduzca el peso: 72.5
La salida digital es 1101
Presione una tecla para continuar . . .
```

← Cadena if-else





## Instrucción SWITCH

Esta instrucción proporciona una alternativa a la cadena if-else, para casos donde se **compara** el valor de una **expresión de numero entero** con un **valor específico**.

*Su sintaxis:*

```
switch (expresión)
{ //inicio de instrucción compuesta
  case valor_1: //termina en dos puntos
    instrucción1;
    instrucción2;
    ...
    break;

  case valor_2:
    instrucción;
    ...
    break;

  default:
    instrucciónaa;
    ...
} // fin de switch y de la instrucción compuesta
```



## Componentes Instrucción SWITCH

**Esta instrucción utiliza cuatro palabras claves nuevas: switch, case, break y default.**

**“switch”** identifica el **inicio de la instrucción**. La expresión entre paréntesis a continuación es **evaluada** y el resultado **comparado** con los sucesivos valores de la instrucción.

**“case”** identifica o etiqueta los valores individuales que se comparan con el valor de la expresión entre paréntesis. Cuando los valores corresponden se ejecutan las instrucciones.

Puede haber cualquier cantidad de etiquetas case.

Si el valor comparado no se corresponde con ninguno, no se ejecuta nada excepto que este la etiqueta **default**.

Una vez encontrado una correspondencia no se evalúan mas case, y se ejecuta la instrucción compuesta interna hasta el break.



## Instrucción SWITCH

**“break”** identifica el fin de un case particular y causa una salida inmediata de la instrucción switch. Si no es colocado, se ejecutan todos los cases hasta encontrar una instrucción de finalización.

De acuerdo al valor de la expresión la palabra “case” determina **el punto de inicio de ejecución** dentro de la instrucción, y la palabra “break” **determina el punto de finalización**.

Ejemplo:

Apilado  
de casos {

```
switch (numero)
{
    case 1:
        cout << "Que tenga una buena mañana\n";
        break;
    case 2:
        cout << "Que tenga un buen dia\n";
        break;
    case 3:
    case 4:
    case 5:
        cout << "Que tenga una buena tarde\n";
}
```



## Instrucción SWITCH

En el ejemplo anterior si la variable tiene almacenado, 3, 4 o 5 se muestra el mismo mensaje. Y si la variable tiene cualquier numero que no este en la lista, no se imprime nada, porque no hay un **default**.

**default:** esta instrucción es opcional, y opera en caso que el valor de la «expresion» no sea igual a ningún valor **case**.

Una buena practica de programación es ordenar los **case** **ascendente** pero no es un requisito.

En función que los datos de tipo **carácter** son almacenado previo convertirlos a números enteros; es posible usar una instrucción switch para cambiar con base en el valor de una expresión de carácter.

```
switch(eleccion)
{
    case 'a':
    case 'e':
        cout<<"vocal";

    default:
        cout<<"no es a ni e";
}
```



```
int main()
{
    int opselect; //declaro variables
    double fnum, snum;

    cout<<"Por favor introduzca dos numeros : ";
    cin>> fnum >> snum; //ingreso valores
    cout<<"\nIntroduzca un codigo seleccionado: ";
    cout<<"\n    1 para adiccion ";
    cout<<"\n    2 para multiplicacion ";
    cout<<"\n    3 para division: ";
    cin>>opselect; //ingreso operacion a realizar

    switch(opselect) //estructura de seleccion de operacion
    { case 1:
        cout<<"\nLa Suma de los numeros introducidos es "<< fnum + snum<<"\n\n\n"; //imprimi
        break;
        case 2:
        cout<<"\nEl producto de los numeros introducidos es "<< fnum * snum<<"\n\n\n";
        break;
        case 3:
        cout<<"\nEl primer numero dividido entre el segundo es: " << fnum / snum<<"\n\n\n";
        break;
    }

    system("pause"); //pauso el sistema para ver resultados
    return 0;
}
```

Por favor introduzca dos numeros : 25 2

Introduzca un codigo seleccionado:

1 para adiccion

2 para multiplicacion

3 para division: 3

El primer numero dividido entre el segundo es: 12.5

Presione una tecla para continuar . . .



## Apéndice

Además de compararse datos numéricos, puede compararse datos de carácter. Por ejemplo en el código Unicode la letra A se almacena usando un código que tiene un valor numérico inferior que la letra B, y así sucesivamente.

Expresión	Valor	Interpretación
'A' > 'C'	0	falso
'D' <= 'Z'	1	verdadero
'E' == 'F'	0	falso
'g' >= 'm'	0	falso
'b' != 'c'	1	verdadero
'a' == 'A'	0	falso
'B' < 'a'	1	verdadero
'b' > 'z'	1	verdadero

Expresión	Valor	Interpretación
"Hola"> "Adios"	1	verdadero
"SOLANO" > "JIMENES"	1	verdadero
"123" > "1227"	1	verdadero
"Bejuco" > "Beata"	1	verdadero
"Hombre" == "Mujer"	0	falso
"planta" <"planeta"	0	falso

Pueden compararse cadenas de caracteres, un espacio en blanco precede, o sea “es menor que” todas las letras y números. Las cadenas se comparan carácter a carácter y la que tenga el menor carácter primero es “menor”. VER TABLA ASCII!!!





## Apéndice

Tabla ASCII

Expresión	Valor
'A' > 'C'	0
'D' <= 'Z'	1
'E' == 'F'	0
'g' >= 'm'	0
'b' != 'c'	1
'a' == 'A'	0
'B' < 'a'	1
'b' > 'z'	1

Expresión	Valor	Interpretación
"Hola"> "Adios"	1	verdadero
"SOLANO" > "JIMENES"	1	verdadero
"123" > "1227"	1	verdadero
"Bejuco" > "Beata"	1	verdadero
"Hombre" == "Mujer"	0	falso
"planta" < "planeta"	0	falso

1	␣	26	→	51	3	76	L	101	e	126		151		176		201		226		251
2	⦿	27	←	52	4	77	M	102	f	127		152		177		202		227		252
3	♥	28	L	53	5	78	N	103	g	128		153		178		203		228		253
4	⬢	29	↔	54	6	79	O	104	h	129		154		179		204		229		254
5	♠	30	▲	55	7	80	P	105	i	130		155		180		205		230		255
6	♣	31	▼	56	8	81	Q	106	j	131		156		181		206		231		256
7	•	32	⦿	57	9	82	R	107	k	132		157		182		207		232		257
8	▣	33	!	58	:	83	S	108	l	133		158		183		208		233		258
9	○	34	"	59	;	84	T	109	m	134		159		184		209		234		259
10	■	35	#	60	<	85	U	110	n	135		160		185		210		235		260
11	⚡	36	\$	61	=	86	V	111	o	136		161		186		211		236		261
12	☉	37	%	62	>	87	W	112	p	137		162		187		212		237		262
13	♪	38	&	63	?	88	X	113	q	138		163		188		213		238		263
14	♫	39		64	@	89	Y	114	r	139		164		189		214		239		264
15	☀	40	(	65	A	90	Z	115	s	140		165		190		215		240		265
16	▶	41	)	66	B	91	[	116	t	141		166		191		216		241		266
17	◀	42	*	67	C	92	\	117	u	142		167		192		217		242		267
18	↕	43	+	68	D	93	]	118	v	143		168		193		218		243		268
19	!!	44	,	69	E	94	^	119	w	144		169		194		219		244		269
20	¶	45	-	70	F	95	_	120	x	145		170		195		220		245		270
21	§	46	.	71	G	96	`	121	y	146		171		196		221		246		271
22	—	47	/	72	H	97	a	122	z	147		172		197		222		247		272
23	ı	48	0	73	I	98	b	123	{	148		173		198		223		248		273
24	↑	49	1	74	J	99	c	124		149		174		199		224		249		274
25	↓	50	2	75	K	100	d	125	}	150		175		200		225		250		275



## Apéndice

- **Alcance de un bloque:**
- Toda instrucción contenida dentro de una instrucción compuesta, constituye un solo bloque de código. Cualquier variable declarada dentro de dicho bloque solo tiene significado entre su declaración y las llaves de cierre que definen el bloque:

```
{ // comienzo bloque exterior  
int a = 25, b= 17;  
cout<<"El valor de a es" << a<<"y b es"<<b
```

```
    { //comienzo bloque interior  
      float a = 46.25;  
      int c = 10;
```

```
      cout<<"a es ahora"<<a<<"b es ahora"<<b  
      <<"y c es ahora"<<c<<endl;  
    } //fin de bloque interior
```

```
cout<<"a es ahora" << a<<"y b es"<<b<<endl;  
} //fin de bloque exterior
```

**El valor de a es 25 y b es 17  
a es ahora 46.25 b es ahora 17 y c es 10  
a es ahora 25 y b es 17**

*También puede observarse en DEV++*