Objetivos de la práctica:
- Conocer los operadores relacionales y lógicos.
- Trabajar las estructuras de control selectivas en la realización de programas.

Operadores relacionales

Los operadores relaciónales o de comparación sirven para comparar dos cantidades y devuelven un valor de "VERDADERO" si la relación es cierta y un valor de "FALSO" si no es cierta. No hay ninguna otra posibilidad.
Las cantidades que se comparan pueden ser, constantes, variables, valores que retornan funciones o expresiones donde se combinen las tres cosas.
El conjunto de estas cantidades combinadas con los operadores relacionales forman una expresión relacional.
Los operadores relacionales definidos dentro del C son:

> *Mayor que*	*!= Distinto de*	**>=***Mayor o igual que*	**<=***Menor o igual que*
< *Menor que*	**==** *Igual que* (no confundir con el operador de asignación **=**)		

Sobre los valores de VERDADERO y FALSO en el C/C++

Para el C, es verdadero cualquier valor que sea distinto de cero, siendo falso sólo cuando es igual a cero. Es por ello que tiene sentido una expresión del tipo **`if( x )`**, y esta será cierta siempre que **`x`** sea distinto de cero, independientemente del tipo **`x`**.
Por otro lado, habíamos dicho que una expresión relacional devolvía un valor de VERDADERO o de FALSO. En realidad devuelve el valor `1.0` cuando la expresión es cierta y `0.0` cuando es falsa.

Operadores lógicos

`&&`	*AND lógico*	`||`	*OR lógico*	`!`	*NOT lógico*

Prioridad y orden de evaluación de los operadores aritméticos, lógicos y relacionales

1) `( )`	2) `- ! ++ --`	3) `* / %`	4) `+ -`	5) `< <= > >=`
6) `== !=`	7) `&&`	8) `||`	9) `= *= /= %= += -=`	

Sentencias de Control Selectivas

- SENTENCIA **`if`** (Alternativa simple)
  Toma una decisión referente a la sentencia (o grupo de sentencias) que se debe ejecutar a continuación, basándose en el resultado (verdadero o falso) de una expresión.

➢ Sintaxis:

```
if ( Expresión Condicional )
        sentencia1;
```

```
if (Expresión Condicional)
{
        S1;
        S2;
        .
        Sn
}
```

  La expresión condicional está contenida entre paréntesis y su resultado decide si la **`sentencia1`** (o el bloque de sentencias **`{S1; S2;...; Sn}`**), que viene a continuación, se debe ejecutar o no (se ejecutará cuando el resultado sea VERDADERO).
  El flujo del programa continúa siempre hacia adelante, e independientemente de que se haya ejecutado o no la **`sentencia1`** o el bloque de sentencias, se ejecutará el resto del programa.

- SENTENCIA **`if ... else`** (Alternativa doble)
  Ofrece la posibilidad de ejecutar una ó más sentencias, como alternativa al caso que fuera FALSA la expresión condicional.

- Sintaxis:

```
if ( Expresión Condicional )
        sentencia1;
else
        sentencia2;
```

```
if (Expresión Condicional)
{
        Bloque de sentencias 1;
}
else
{
        Bloque de sentencias 2;
}
```

Si el resultado de la expresión es verdadero, se ejecutará el bloque de sentencias1, y si es falso, el bloque de sentencias2.
El flujo programa transcurre siempre hacia adelante. El programa ejecutará uno de los dos bloques de sentencias, y a continuación ejecutará el resto del programa.

Anidamiento de sentencias *if ... else*

Como parte del bloque de sentencias1 o del bloque de sentencias2, puede aparecer otras sentencias **if...else**.

Bloque de sentencias *if ... else*

El bloque de sentencias **if..else** es un caso particular de anidamiento de sentencias **if.. else** que merece una mención especial, debido a que es una configuración bastante usual dentro de los programas.

- Sintaxis:

```
if (Expresión Condicional1)
{
   Bloque de sentencias1;
}
else if (Expresión Condicional2)
     {
        Bloque de sentencias2;
     }
...
     else if ( Expresión CondicionalN-1 )
          {
             Bloque de SentenciasN-1;
          }
          else
          {
             Bloque de sentenciasN;
          }
```

Las expresiones condicionales se evaluarán de arriba a abajo y una vez encontrada una cuyo valor sea VERDADERO, el programa ejecutará su correspondiente bloque de instrucciones. Si no se cumple ninguna de las condiciones, se ejecutará el bloque de instrucciones asociado al último *else* (en el caso de que exista).

- SENTENCIA **switch** (Alternativa múltiple)
  Permite ejecutar una de varias acciones en función del valor de una expresión entera (**short**, **int** o **long**) o caracter (**char**).

- Sintaxis:

```
switch(expresión)
{
   case Constante1:
       Bloque de sentencias1;
       break;
   case Constante2:
       Bloque de sentencias2;
       break;
   case Constante3:
       Bloque de sentencias3;
       break;
   case .....
       ........
   default:
       Bloque de sentencia por defecto;
}
```

break es una orden imperativa de salida del bloque de la sentencia **switch**.

La sentencia **switch** compara sucesivamente el valor de la expresión contenida entre paréntesis, con cada una de la constantes situadas detrás de la palabra **case**. Cuando encuentra una correspondencia, ejecuta el conjunto de sentencias que viene a continuación hasta encontrar una sentencia **break** o hasta que termina la sentencia **switch**.

Igual que antes, el flujo del programa transcurre siempre hacia adelante, y una vez ejecutado el conjunto de sentencias asociado a la constante que coincide con la expresión, continúa la ejecución del resto del programa.

Resumen de CONCEPTOS BÁSICOS: Sentencias IF y SWITCH

Sentencia if

```
//Una alternativa
   if (a != 0)
     resultado = a/b
//Dos alternativas
   if (a >= 0)
       cout << "positivo";
   else
       cout << "negativo";

//Múltiples alternativas (if anidados)
   if (x < 0)
   {
       cout << "negativo";
       abs_x = -x;
   }
   else
       if (x == 0)
       {
             cout << "cero";
             abs_x = 0;
       }
       else
       {
            cout << "positivo";
            abs_x = x;
       }
```

Sentencia switch

```
switch (sig_car)
{
   case 'a':
     cout << "sobresaliente";
     break;

   case 'b' :
     cout << "notable ";
     break ;

   case 'c' :
     cout << "aprobado";
     break;

   case 'd' :
     cout << "suspenso";
     break;

   default://otros casos
     cout << "no valido";
} // fin de switch
```

ERRORES FRECUENTES DE PROGRAMACIÓN

1. Uno de los errores más comunes en una sentencia **`if`** es utilizar operador de asignación '=' en lugar de un operador de igualdad "==".

2. En una sentencia **`if`** anidado cada cláusula **`else`** se corresponde con la **`if`** precedente más cercana. Por ejemplo, en el segmento de programación siguiente:

```
if (a > 0)
if (b > 0)
c= a+b;
else
c = a* b * c;
d = a  * b;
```

¿Cuál es la sentencia **`if`** asociada a **`else`**?
El sistema más fácil para evitar errores es el sangrado, con lo que ya se aprecia que la cláusula **`else`** se corresponde a la sentencia que contiene condición: `b > 0`

```
if (a > 0)
     if (b > 0)
          c = a +b;
     else
          c = a * b * c ;
d = a * b;
```

Una de las razones de utilizar la Guía de Estilo es la claridad de los programas y evitar errores innecesarios.

3. El selector de una sentencia **`switch`** debe ser de tipo entero o compatible entero (ordinal). Así las constantes reales, como 2.4, -4.5, 3.1416, no pueden ser utilizadas en el selector.

4. Cuando se utiliza una sentencia **`switch`**, asegúrese que el selector de **`switch`** y las etiquetas **`case`** son del mismo tipo (`int`, `char`, pero no `float`). Si el selector se evalúa a un valor no listado en ninguna de las etiquetas **`case`**, la sentencia **`switch`** no gestionará ninguna acción; por esta causa se suele poner una etiqueta **`default`** para resolver este problema.