imoObjetivo de la práctica:
- Aprender el uso de estructuras condicionales.

NOTA 1: Durante la práctica todos los ejercicios deberán ser guardados ***temporalmente*** *en el directorio c:\tmp. Una vez finalizada la misma y transferidos los ficheros a un disquete, se deberá eliminar dicho directorio.*

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";

//Multiples 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` o `bool`, 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.

PROBLEMAS:

En esta práctica realizaremos un programa que solucione un problema real de gestión de empresas. Se escribirá un único programa sobre el que realizaremos todos los ejercicios planteados guardando en cada caso el ejercicio en un fichero diferente.

El programa tratará del cálculo automático de comisiones a vendedores en función de diferentes parámetros. Para ello tendremos que utilizar estructuras condicionales que discriminen la ejecución del programa y lleven el flujo del código en una dirección o en otras.

Al mismo tiempo veremos como un programa, tal y como pasa en la vida real, se complica conforme se le añaden nuevas funciones.

1. Nuestra empresa calcula las comisiones a sus vendedores en función de la categoría a la que pertenece. Los porcentajes según la categoria a la que pertenece son:

   Categoría A 2%

   Categoria B 8%

   Categoria C 12%

   Realiza un primer programa que, en función de la categoría del trabajador, muestre el porcentaje asociado.

2. La misma empresa anterior nos propone una ampliación para, en función de la categoría y las unidades vendidas, calcule las comisiones a sus vendedores. Estos cálculos los realiza utilizando la siguiente tabla comparativa:

Ventas	**Comisión**
Menos de 100	n_ventas * 100€ * Porcentaje categoría * 2
Entre 100 y 199	n_ventas * 100€ * Porcentaje categoría * 3

Mas de 199	n_ventas * 100€ * Porcentaje categoría * 4

Se debe de ampliar el apartado anterior para que mediante un menú por pantalla se incluya la nueva opción.

3. Supongamos que tras un año de uso del anterior programa la empresa decide que el programa necesita modificar su funcionamiento en base a si el trabajador a superado la media de ventas de los demás trabajadores. Supongamos que el número de trabajadores es 4 y que su comisión se verá reducida a la mitad si no ha llegado a la media de ventas de todos los trabajadores y se multiplicará por 2 si ha llegado o ha superado la media.

   Modificar el programa realizado en el apartado 2 para desarrollar la nueva utilidad introducida en este punto.

4. El director general de la empresa que nos ha contratado el desarrollo, nos pide que modifiquemos el funcionamiento del programa para que si el usuario pertenece a la categoria C y no supera un mínimo de ventas de 100 se muestre un mensaje avisando que este vendedor ya no pertenecerá la categoría C pasando a pertenecer a la categoría B.

   Realizar esta modificación para que, sin perder las opciones de los puntos anteriores, al final del cálculo de la comisión, avise si el trabajador cambia o no de categoría.

5. Una vez desarrollado todos los puntos anteriores y entendiendo como se complica un programa en función de cómo se añade nueva opciones y entendiendo la base de las estructuras condicionales contesta a las siguientes preguntas:

   a. ¿Qué estructuras condicionales conoces? ¿Cuándo es mejor aplicar unas u otras?

   b. ¿Que tenemos que incluir en C y C++ cuando queremos que después de un IF se ejecuten mas de una sentencia? ¿Es necesario incluir algo mas despues de un IF cuando solo queremos ejecutar una sentencia?