Práctica 3: Sentencias de Selección en C++.

Los programas definidos hasta este punto se ejecutan de modo secuencial, es decir, una sentencia después de otra. La ejecución comienza con la primera sentencia del programa y prosigue hasta la última sentencia, cada una de las cuales se ejecuta una sola vez. Esta forma de programación sólo permite resolver programas sencillos. Sin embargo, para la resolución de problemas de tipo general se necesita la capacidad de controlar cuáles son las sentencias que se ejecutan, y en qué momentos. Recordemos que la arquitectura de Von Newman exigía la necesidad de una sentencia de salto condicional.

Estructuras de Selección

Las estructuras de selección o condicionales controlan si una sentencia o secuencia de sentencias se ejecutan, en función del cumplimiento o no de una condición o expresión lógica. C++ tiene dos estructuras de control para la selección, ***if*** y ***switch***.

Sentencia *if*

La sentencia ***if*** elige entre varias alternativas en base al valor de una o más expresiones *lógicas.* La notación BNF de esta sentencia es la siguiente(los símbolos terminales se han escrito en negrita):

```
<sent_if> ::=     if (<expres_log>)
                  (<sent>|<bloque>)
                  {else if (<expres_log>) (<sent>|<bloque>)}
                  [else (<sent>|<bloque>)]
```

donde `<expres_log>` es una expresión lógica que ha de ir entre paréntesis, y `<sent>` es una sentencia, y `<bloque>` es un bloque de sentencias que comeinza en '{' y termina en '}'. Por ejemplo:

```
if (condicion_SI)
{  // Sentencias SI
}
else if (condicion_SINO)
{ // Sentencias SINO
}
else
{ // Sentencias EN OTRO CASO
}
```

Sentencia *switch*

La sentencia ***switch*** es una sentencia de C++ que se utiliza para seleccionar una de entre múltiples alternativas. Esta sentencia es especialmente útil cuando la selección se basa en el valor de una variable de un tipo simple o de una expresión de un tipo simple denominada expresión de control o selector. La notación BNF de la sentencia es la siguiente (los símbolos terminales se han escrito en negrita):

```
<sent_case>::=    switch (<expres>)
                  {
                     { <rama> }
                  [default : {<sent>} ; break;]
                  }
<rama> ::= case <exp_const> : { case <exp_const> :} {<sent>}; break;
```

donde `<expres>` es una expresión, `<sent>` es una sentencia o secuencia de sentencias terminadas en punto y coma (`;`), y `<exp_const>` es una expresión constante. Por ejemplo:

```
switch (numero)
{ case 1:
  case 3:
  case 5:   // Sentencias para los valores 1,3 y 5
            break;
  casee 2:  // Sentencias para el  valor   2
            break;
  case 4:   // Sentencias para el  valor   4
```

```
                    break;
            default : // Sentencias EN OTRO CASO
                    break;
          }
```

EJEMPLO. Veamos el siguiente ejemplo en el que usan sentencias `if` y `switch`
Escribir un programa que lea un número entre 1 y 5. Si el número es menor que 1 o mayor que 5 el programa debe avisar y no hacer nada y en caso de que sea válido debe mostrar el número romano que lo representa.

```
/*-----------------------------------------------------------------
|     Autor:                                                      |
|     Fecha:                          Versión: 1.0                |
|-----------------------------------------------------------------|
|     Descripción del Programa:                                   |
|                                                                 |
| ----------------------------------------------------------------*/

// Incluir E/S y Librerías Standard
#include <iostream>
#include <cstdlib>
using namespace std;

// Zona de Declaración de CONSTANTES
const int MIN = 1;
const int MAX = 5;

int main()
{ // Zona de Declaración de VARIABLES
  int num;

  cout << "Introduzca un número entre " << MIN
       << " y " << MAX << ": ";
  cin >> num;

  if (num<MIN)
  { cout << "Error: El Número debe ser mayor que " << MIN << endl;
  }
  else if (num>MAX)
  { cout << "Error: El Número debe ser menor que " << MAX << endl;
  }
  else
  { switch(num)
    { case 1: cout << num << " = I " << endl;
              break;
      case 2: cout << num << " = II " << endl;
              break;
      case 3: cout << num << " = III " << endl;
              break;
      case 4: cout << num << " = IV " << endl;
              break;
      case 5: cout << num << " = V " << endl;
              break;
    }
  }

  system("PAUSE");              // Hace una pausa
  return 0;                     // Valor de retorno a S.O.
}
```

ENUNCIADO PRÁCTICA.

1. Escriba un programa que lea un número natural por teclado y nos diga si es par o impar.
2. Escriba un programa que lea un carácter desde teclado (que puede ser mayúscula o minúscula) y nos diga si es una vocal, una consonante, un numero o un carácter especial.
3. Escriba un programa que lea un número natural que corresponde al número de lados de un polígono y nos mostrará por pantalla el tipo de polígono que es: triángulo (3 lados), cuadriláteros (4 lados), pentágonos (5 lados), hexágonos (6 lados), heptágonos (7 lados), octágonos (8 lados) y "polígono de más de 8 lados" para el resto. Tenga en cuenta también que no existen polígonos de menos de 3 lados.