Condiciones / Bifurcaciones

Operador condicional

El operador condicional es un operador con tres operandos (ternario) que tiene la siguiente forma general:

```
expresion_1 ? expresion_2 : expresion_3;
```

Se evalúa expresion_1. Si el resultado de dicha evaluación es true (!0), se ejecuta expresion_2; si el resultado es false (0), se ejecuta expresion_3.
Ejemplo:

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

int main(void){
   int n;
   cin >> n;
   cout << ((n%2==0)?"Es par":"Es impar");
   system("pause >nul");
   return 1;
}
```

T03CF01.cpp

Sentencia *if*

Esta sentencia de control permite ejecutar o no una sentencia simple o compuesta según se cumpla o no una determinada condición. Esta sentencia tiene la siguiente forma general:

```
if (expresion){
  sentencia;
}
```

Se evalúa expresion. Si el resultado es true (!0), se ejecuta sentencia; si el resultado es false (0), se salta sentencia y se prosigue en la línea siguiente. Hay que recordar que sentencia puede ser una sentencia simple o compuesta.

Sentencia *if…else*

Esta sentencia permite realizar una bifurcación, ejecutando una parte u otra del programa según se cumpla o no una cierta condición. La forma general es la siguiente:

```
if (expresion){
  sentencia_1;
}
else{
  sentencia_2;
}
```

Se evalúa expresion. Si el resultado es true (!0), se ejecuta sentencia_1 y se prosigue en la línea siguiente a sentencia_2; si el resultado es false (0), se salta sentencia_1, se ejecuta sentencia_2 y se

prosigue en la línea siguiente. Hay que indicar aquí también que sentencia_1 y sentencia_2 pueden ser sentencias simples o compuestas.

Sentencia *if ... else* múltiple

Esta sentencia permite realizar una ramificación múltiple, ejecutando una entre varias partes del programa según se cumpla una entre n condiciones. La forma general es la siguiente:

```
if (expresion_1){
  sentencia_1;
}
else if (expresion_2){
  sentencia_2;
}
else if (expresion_3){
  sentencia_3;
}
else if (...){
  ...
}
[else{
  sentencia_n;
}]
```

Se evalúa expresion_1. Si el resultado es true, se ejecuta sentencia_1. Si el resultado es false, se salta sentencia_1 y se evalúa expresion_2. Si el resultado es true se ejecuta sentencia_2, mientras que si es false se evalúa expresion_3 y así sucesivamente. Si ninguna de las expresiones o condiciones es true se ejecuta sentencia_n que es la opción por defecto (puede que no sea necesario una opción por defecto por lo tanto no se escribiría). Todas las sentencias pueden ser simples o compuestas.

Sentencia *switch*

La sentencia que se va a describir a continuación desarrolla una función similar a la de la sentencia if ... else con múltiples ramificaciones, aunque como se puede ver presenta también importantes diferencias. La forma general de la sentencia switch es la siguiente:

```
switch (expresion) {
case expresion_cte_1:
  sentencia_1;
break;
case expresion_cte_2:
  sentencia_2;
  ...
break;
case expresion_cte_n:
  sentencia_n;
break;
[default:
  sentencia_default;
break;]
}
```

Se evalúa expresion y se considera el resultado de dicha evaluación. Si dicho resultado coincide con el valor constante expresion_cte_1, se ejecuta sentencia_1 seguida de sentencia_2, sentencia_3, ..., sentencia_n. Si el resultado coincide con el valor constante expresion_cte_2, se ejecuta sentencia_2 seguida de sentencia_3, ..., sentencia_n. En general, se ejecutan todas aquellas sentencias que están a continuación de la expresion_cte cuyo valor coincide con el resultado calculado al principio. Si ninguna expresion_cte coincide se ejecuta la sentencia que está a continuación de default. Si se desea ejecutar únicamente una sentencia_i (y no todo un conjunto de ellas), basta poner una sentencia break a continuación (en algunos casos puede utilizarse la sentencia return o la función exit()). El efecto de la sentencia break es dar por terminada la ejecución de la sentencia switch. Existe también la posibilidad de ejecutar la misma sentencia_i para varios valores del resultado de expresion, poniendo varios case expresion_cte seguidos.
El siguiente ejemplo ilustra las posibilidades citadas:

```
switch (expresion) {
case expresion_cte_1:
  sentencia_1;
break;
case expresion_cte_2:
case expresion_cte_3:
  sentencia_2;
break;
default:
  sentencia_3;
}
```

Sentencias *if* anidadas

Una sentencia if puede incluir otros if dentro de la parte correspondiente a su sentencia, A estas sentencias se les llama sentencias anidadas (una dentro de otra), por ejemplo,

```
if (a >= b){
  if (b != 0.0){
    c = a/b;
  }
}
```

En ocasiones pueden aparecer dificultades de interpretación con sentencias if...else anidadas, como en el caso siguiente:

```
if (a >= b)
  if (b != 0.0)
    c = a/b;
else
  c = 0.0;
```

En principio se podría plantear la duda de a cuál de los dos if corresponde la parte else del programa. Los espacios en blanco –las indentaciones de las líneas– parecen indicar que la sentencia que sigue a else corresponde al segundo de los if, y así es en realidad, pues la regla es que el else pertenece al if más cercano. Sin embargo, no se olvide que el compilador de C no considera los espacios en blanco (aunque sea muy conveniente introducirlos para hacer más claro y legible el programa), y que si se quisiera que el else perteneciera al primero de los if no bastaría cambiar los espacios en blanco, sino que habría que utilizar llaves, en la forma:

```
if (a >= b) {
  if (b != 0.0)
    c = a/b;
  }
else
  c = 0.0;
```

Bibliografía:

- García de Jalón. J, Rodríguez. J, Sarriegui. J, Goñi. R, Brazales. A, Funes. P, Larzabal. A, Rodríguez. R, Aprenda C++ como si estuviera en primero, San Sebastián, Abril 1998.
- http://www.cplusplus.com/doc/tutorial/control/ consultado el Viernes 20 de noviembre de 2009.
- Peña Basurto. M, Cela Espín. J, Introducción a la programación en C, UPC, Septiembre de 2000, ISBN: 84-8301-429-7.