Programming challenges in C++

2. Flow control structures in C++

Nacho Iborra

IES San Vicente

Table of Contents

Programming challenges in C++

1. Selective structures

In this session we are going to learn the flow control structures available in C++. As you will see, they are more or less the same that we can find in other languages, such as Java or C#. Let's start with the selective structures...

1.1. if .. else if .. else

You can use the *if* and *if .. else if .. else if .. else* structures as in many other languages.

```
if (number > 0)
{
    cout << "It is positive" << endl;
}
else if (number < -10)
{
    cout << "It is under -10" << endl;
}
else
{
    cout << "It is between -10 and 0" << endl;
}
```

1.2. switch

Besides, there's a *switch* clause, with their corresponding *cases* and a *default* case. The data managed in the *switch* clause must be a primitive type (such as integer values, characters or floating point numbers... strings are NOT allowed). The *break* at the end of each *case* is not compulsory, but if you don't put it, you go to next *case*.

```
switch(number)
{
    case 0: cout << "It is 0" << endl; break;
    case 1: cout << "It is 1" << endl;
    case 2: cout << "It is 2" << endl; break;
    default: cout << "Unknown number" << endl;
}
```

In previous example, if number is 1, it would output the messages "It is 1" and "It is 2", since there is no *break* clause at *case 1*.

2. Loops

C++ has the most common iterative structures that (almost) every programming language has: *while, do..while* and *for*. The way you use them is similar to other languages such as C or C#.

2.1. while

We will use this loop when we don't know how many iterations are expected, and we don't even know if there will be one iteration. This example counts from 1 to 10:

```
int n = 1;
while (n <= 10)
{
    cout << n << endl;
    n++;
}
```

2.2. do..while

We will use this loop when we don't know how many iterations are expected, but we know that there will be (at least) one iteration. It is very usual when we ask the user to type something and then check the input and ask the user again. If we do the same loop than in previous example with a do..while structure, it would look like this.

```
int n = 1;
do
{
    cout << n << endl;
    n++;
} while (n <= 10);
```

2.3. for

We will use this loop when we know how many iterations are expected. The counter from 1 to 10 should be done with this structure preferably, and it would look like this:

```
for (int n = 1; n <= 10; n++)
{
    cout << n << endl;
}
```

Note that we can declare variables in *for* loops (and in the middle of other code, as in other languages such as Java or C#).

3. Challenges for this session

3.1. Sample challenge: Contando en la arena

As a sample of how to use flow control structures in a challenge, let's face this one. It tells us to read a number N and write the number 1 a total of N times. We must repeat this process until we read N = 0.

A possible solution to this challenge would be this one:

```
#include <iostream>

using namespace std;

int main()
{
    int number;

    do
    {
        cin >> number;
        if (number > 0)
        {
            for (int i = 0; i < number; i++)
                cout << 1;
            cout << endl;
        }
    } while (number > 0);
    return 0;
}
```

Note that we use a `do..while` statement to process each number until we read a 0. Then, we use an `if` statement to determine if the number that we have just read is a 0. If it is not, then we write N times the number 1, using a `for` statement.

3.2. Try yourself: Los problemas de ser rico

Let's practice on your own with this challenge from Acepta el reto.

3.3. Try yourself: Sobre la tela de una araña

To finish with this session, if you have some extra time after finishing previous challenge, let's try this last one.