Control Structures

A program is **usually** not limited to a linear sequence of instructions. During its process it may branch off, **repeat code** or **take decisions**. For that purpose, C++ provides control structures that serve to specify what has to be done by our program, when and under which conditions.

With the introduction of control structures we are going to have to introduce a new concept: the ***compound statement*** or ***block***. A **block** is a group of statements which are separated by semicolons (;) like all C++ statements, but grouped together in a block enclosed in braces { }:

{ statement1; statement2; statement3; }

Most of the control structures that we will see in this section require a generic statement as part of its syntax. A statement can be **either** a **simple statement** (a simple instruction ending with a semicolon) or a **compound statement** (several instructions grouped in a **block**), like the one just described. In the case that we want the statement to be a simple statement, we do **not** need to enclose it in braces ({}). **But** in the case that we want the statement to be a compound statement it **must be** enclosed between braces ({}), forming a block.

Conditional structure: if and else

The **if** keyword is used to execute a **statement** or **block** only if a condition is fulfilled. Its form is:

if (condition) statement

Where condition is the expression that is being evaluated. If this condition is **true**, statement is executed. If it is **false**, statement is ignored (not executed) and the program continues right after this conditional structure.

For example, the following code fragment prints Z is 300 only if the value stored in the Z variable is indeed 300:

```
if (Z == 300)
cout << "Z is 300";
```

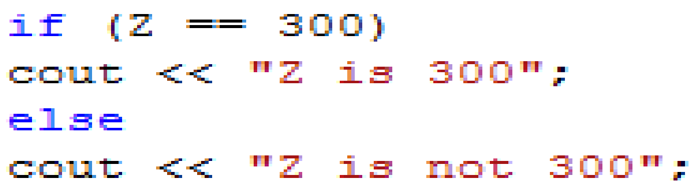
If we want **more than** a single statement to be executed in case that the condition is **true** we can specify a block using braces { }:

```
if (Z == 300)
{
cout << "Z is ";
cout << Z;
}
```

We can additionally specify what we want to happen if the condition is not fulfilled by using the keyword **else**. Its form used in conjunction with if is:

if (condition) statement1 else statement2

For Example :

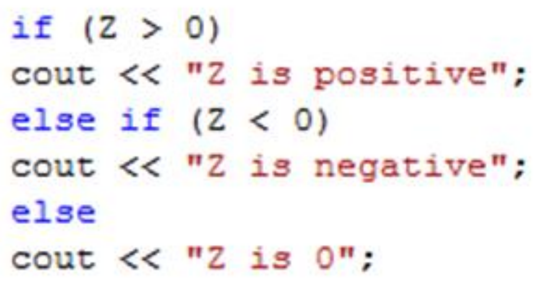


```
if (Z == 300)
cout << "Z is 300";
else
cout << "Z is not 300";
```

prints on the screen Z is 300 if indeed Z has a value of 300, but if it has not -and only if not- it prints out Z is not 300.

The if + else structures can be concatenated with the intention of verifying a range of values.

The following example shows its use telling if the value currently stored in Z is positive, negative or none of them (i.e. zero):

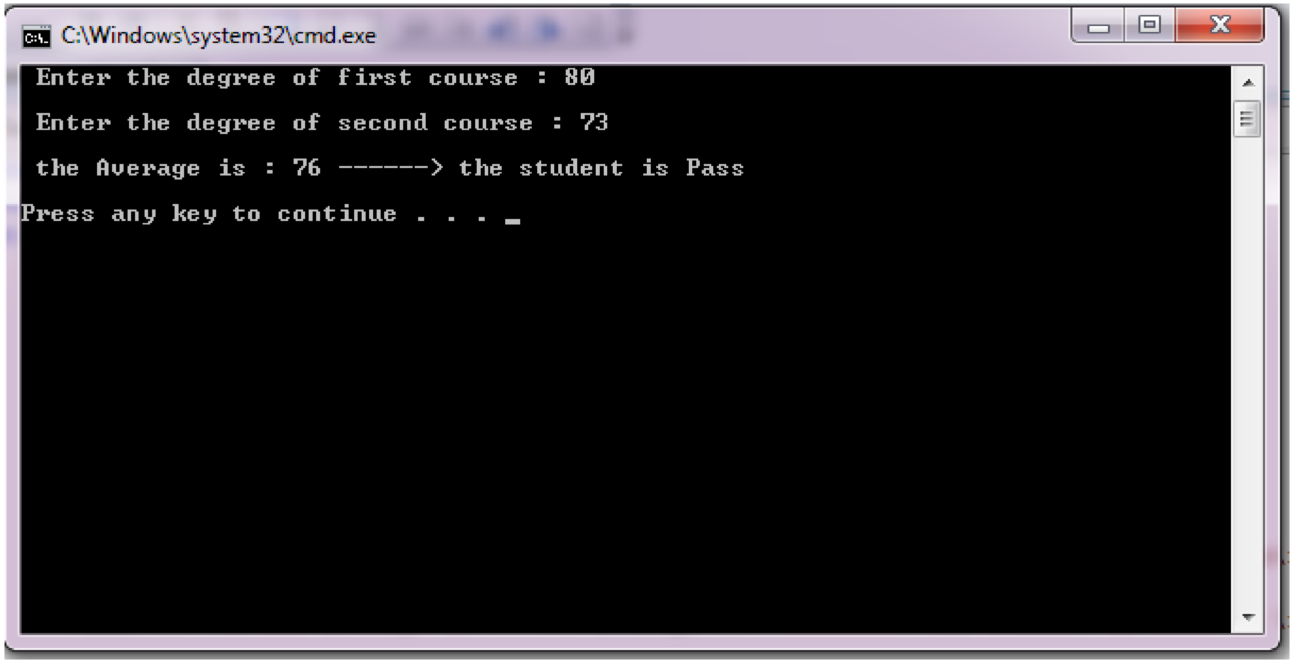


```
if (Z > 0)
cout << "Z is positive";
else if (Z < 0)
cout << "Z is negative";
else
cout << "Z is 0";
```

Remember that in case that we want more than a single statement to be executed, we must group them in a block by enclosing them in braces { }.

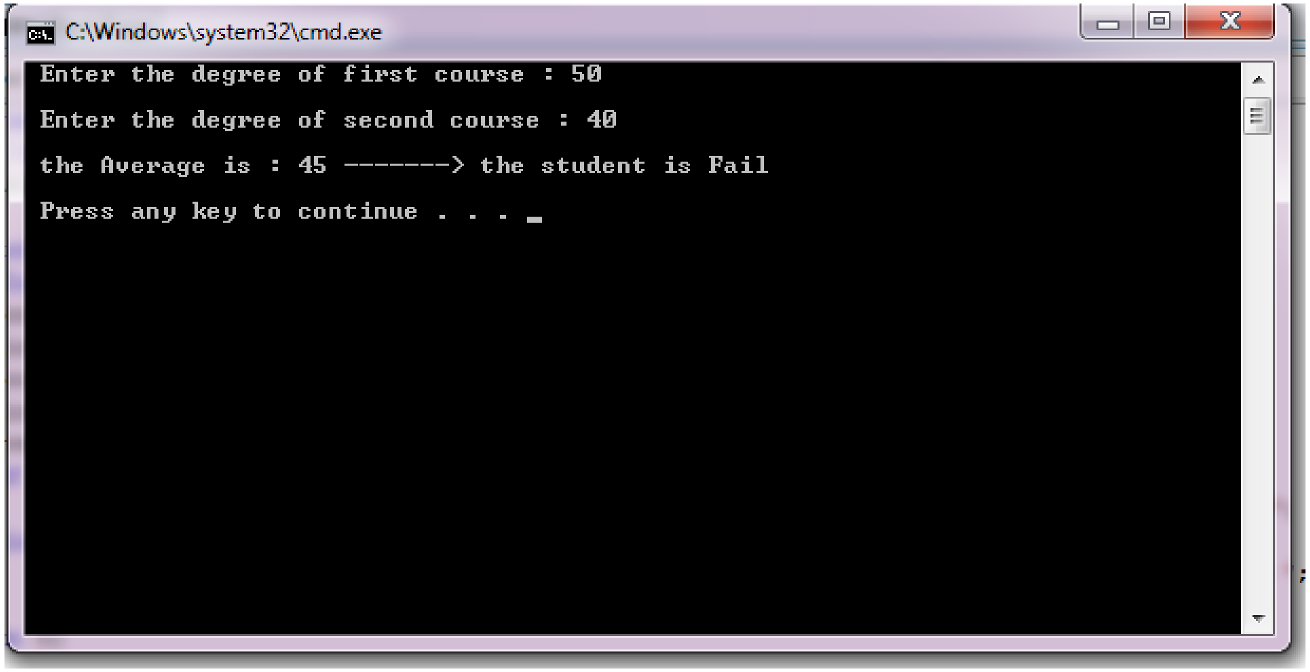
```
#include <iostream>
using namespace std;
int main ()
{
    int C1,C2;
    float average;
    cout << " Enter the degree of first course : ";
    cin >> C1;
    cout << " \n Enter the degree of second course : ";
    cin >> C2;
    average = (C1+C2)/2;
    if ( average >= 50 )
        cout << " \n the Average is : " << average << " ------> the student is Pass \n\n";
    else
       cout << "\n the Average is : " << average << " -------> the student is Fail \n \n ";

return 0;
}
```



```
 Enter the degree of first course : 80
 Enter the degree of second course : 73
 the Average is : 76 ------> the student is Pass
Press any key to continue . . .
```

Output 1 : Pass



```
Enter the degree of first course : 50
Enter the degree of second course : 40
the Average is : 45 -------> the student is Fail
Press any key to continue . . .
```

Output 2 : Fail