Chapter 4 – Decisions

Now we will learn about a handy piece of code essential to programming, the "if-else statement". Take a look at the following program. It asks the user to enter a whole number and the program will tell the user whether the number is positive, negative or zero.

```
1   #include<iostream.h>
2
3   int main()
4   {
5   int x;
6
7   cout<< "Enter an integer" <<endl;
8
9   cin>>x;
10
11      if(x>0)
12      {
13      cout<< "That number is positive!" <<endl;
14      }
15
16      else if(x<0)
17      {
18      cout<< "That number is negative!" <<endl;
19      }
20
21      else
22      {
23      cout<< "That number is 0!" <<endl;
24      }
25
26  return 0;
26
28  }
```

Example 4a

When you enter a number, the computer first checks to see if the number is greater than zero (line 11). If the number is greater than zero then the code in the brackets after the "if statement" will be executed and the program will finish. The computer does not even look at the other options since a true condition has already been met.

If the number is not greater than zero then the computer will check the next option i.e. if it is less than zero (line 16). If x<0 the computer will print "That number is negative!"

Finally if the number is neither positive nor negative then obviously it is equal to zero, and the computer will tell you this by choosing to execute the code after the final else statement.

You can have as many "if or if else statements" you like in your program. You may only have one "if statement" in your program if you like, but it's sometimes good to have an "else statement" at the end to cover all circumstances that may occur when a program is running.

For example if you use only the following "if statement" in your program:

```
cout<< "Enter an number" <<endl;

        cin>>x;

                if(x>0)
                {
                cout<< "That number is positive!" <<endl;
                }
```

Then you will ONLY receive a response when you enter a positive number. The program will just finish otherwise.

You can do a few logic tests on a number if you like. For instance you could include the code:

```
        if(x>0 && x<10)
        {
        cout<<"x is between 0 and 10"<<endl;
        }
```

This will test if x is greater than 0 **AND** less than 10.

```
        if(x<100 || x>500)
        {
        cout<<"x is less than 100 or it is greater than 500"<<endl;
        }
```

This will test if x is greater than 500 **OR** less than 100.

Nested If Statements

You can even have an "if statement" inside another "if statement". These are called "nested if statements". Here is an example.

```
#include<iostream.h>
int main()
{
int x;

cout<< "Enter a number" <<endl;
cin>>x;

        if(x>0)
        {
                if(x==1)
                {
                cout<< "number is one" <<endl;
                }

                else if(x==2)
                {
                cout<< "number is two" <<endl;
                }

                else
                {
                cout<< "number is positive" <<endl;
                }
        }

        else if(x<0)
        {
        cout<< "number is negative" <<endl;
        }

        else
        {
        cout<< "number is zero" <<endl;
        }

return 0;
}
```

Example 4b

This program may look long at first sight but it's pretty simple when you break it down. If we enter a negative number, then when the computer reaches line 9 it will jump immediately to line 27 where it will agree that x<0. If we enter 0, then the computer will skip nearly everything. First it will stop at line 9 and not agree that x>0, then it will jump to line 27 and not agree that x<0 and finally when it gets to line 32 it will execute the code for the "else statement". But what happens when we enter a positive number?

Well, if x>0 then the code in the brackets after this statement will be executed. But this is even more "if statements"! At this stage the computer knows the number is positive and now it is about to test whether it is exactly equal to 1, exactly equal to 2 or something else. It's sort of like our variable x is falling through a sieve, with the computer narrowing down the possibilities. The computer will choose one of the "nested if statements", depending on the number you entered, then it will exit the main "if statement" and finish the program.

Note above that x = = 1 means x is equal to 1. You use **TWO** equal signs to test for equality. x = 1, on the other hand, *assigns* x the value of 1.

TIP! Be careful when using "nested if statements" or lots of "if statements". The use of all the brackets can get confusing. In this program I have colour-coded and indented the brackets to make it easier to see which pair of brackets correspond to which "if statement".