Iteration structures (loops)

Loops have as purpose to **repeat** a statement a certain number of times **or** while a condition is satisfied.

The while loop

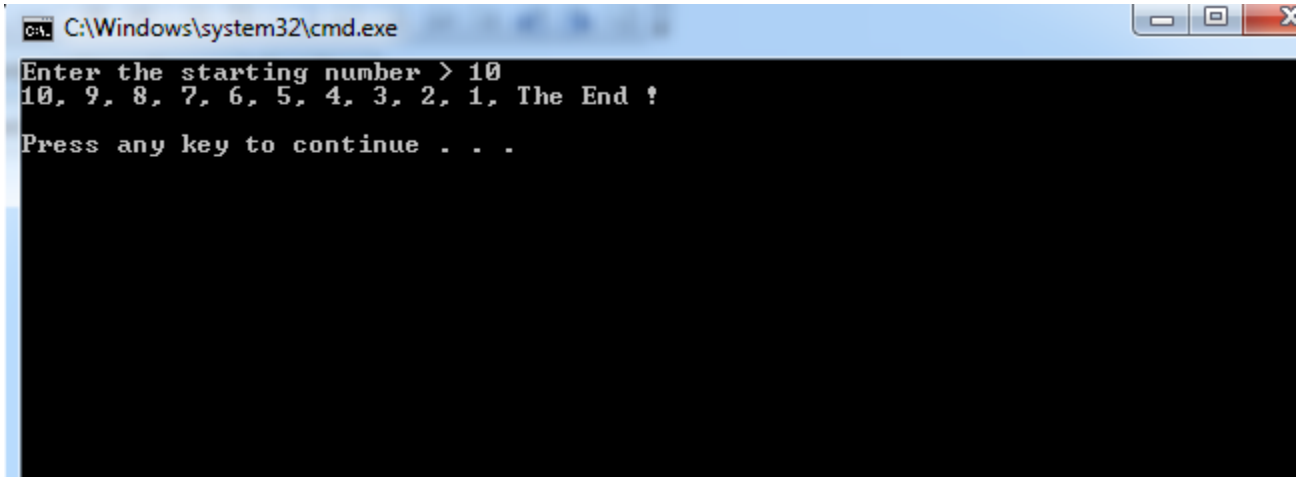
Its format is:

while (expression) statement

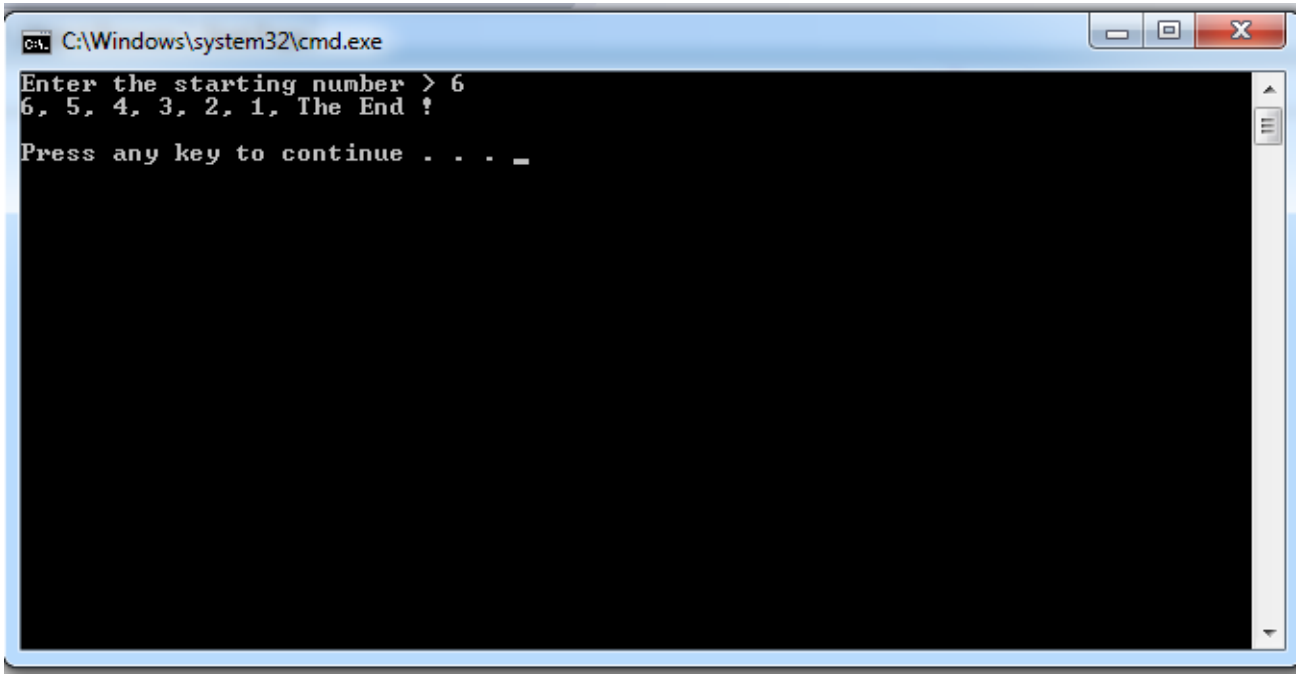
and its functionality is simply to repeat statement while the condition set in expression is true.

For example, we are going to make a program to countdown (العد التنازلي) using a **while-loop**:

```
// custom countdown using while
#include <iostream>
using namespace std;
int main ()
{
     int number;
     cout << "Enter the starting number > ";
     cin >> number;
     while (number>0)
          {
               cout << number << ", ";
               number=number-1;
          }
     cout << "The End !\n \n";
     return 0;
}
```



When input = 10



When input = 6

When the program starts the user is prompted to insert a starting number for the **countdown**. Then the while loop begins, if the value entered by the user fulfills the condition number>0 (**that number is greater than zero**) the block that follows the condition will be **executed** and **repeated** while the condition (number>0) remains being true. The whole process of the previous program can be interpreted according to the following script (beginning in main):

1. User assigns a value to number
2. The while condition is checked (number >0). At this point there are two possibilities :
*** condition is true: statement is executed (to step 3)**
*** condition is false: ignore statement and continue after it (to step 5)**
3. Execute statement:

```
cout << n << ", ";
number = number-1;
```

(prints the value of number on the screen and decreases number by 1)
4. End of block. Return automatically to step 2
5. Continue the program right after the block: print The End ! and end program.

When creating a **while-loop**, we must always consider that it has to **end** at some point, therefore we **must** provide within the **block** some method to **force** the condition to become **false** at some point, **otherwise** the loop will **continue** **looping** **forever**.

In this case we have **included** **number=number-1;** that **decreases** the value of the **variable** that is being **evaluated** in the condition (number) by **one** - this will **finally** make the condition (number>0) to become **false** **after a certain number of loop iterations**: to be more specific, when **number** **becomes 0**, that is where our while-loop and our **countdown** **end**.

Of course this is such a simple action for our computer that the whole **countdown** is performed instantly without any practical delay between numbers.

The do-while loop

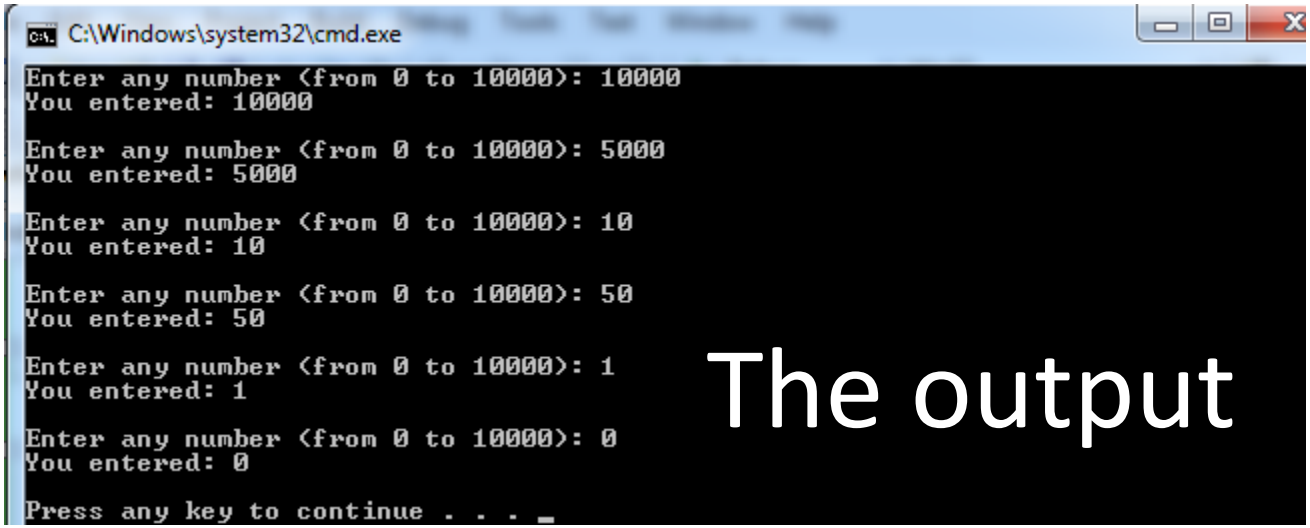
Its format is:

do statement while (condition);

Its functionality is **exactly** the same as the while loop, **except** that condition in the do-while loop is **evaluated** ***after*** the execution of statement ***instead of before***, allowing **at least one execution of statement** even if condition is **never** fulfilled. For **example**, the following example program echoes any number you enter until you enter 0.

The do-while loop is usually used when the condition that has to determine the end of the loop is determined within the loop statement itself, like in the next case, where the user input within the block is what is used to determine if the loop has to end. In fact if you **never** enter the *value 0* in the next example you can be prompted for more numbers forever.

```cpp
// number echoer
#include <iostream>
using namespace std;
int main ()
{
    int n;
    do {
    cout << "Enter any number (from 0 to 10000): ";
    cin >> n;
    cout << "You entered: " << n << "\n \n";
    } while (n != 0);
    return 0;
}
```



The output