Introduction to Programming

Lecture while , do-while Loops

Muhammad Salman

ABASYN UNIVERSITY

Summary

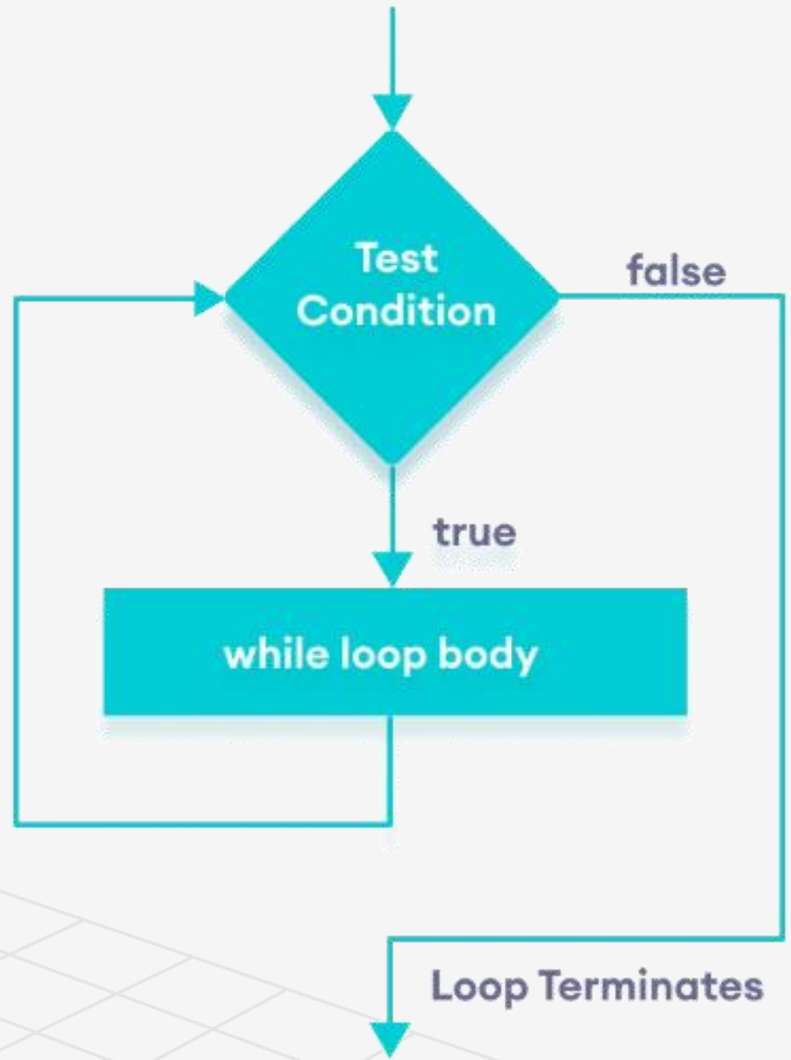
- The “while” loop
- Flow chart
  - A program
  - Continue statement
  - Break statement
- “do-while” loop
- Flow chart
  - A program

Loop

- A statement or set of statement that is executed repeatedly is called loop. The statement(s) in a loop are executed for a specified number of times or until some given condition remain true.
  - In C++ there are three kind of loop statements these are:
    - The while loop
    - The do-while loop
    - The for loop

Flowchart of while Loop

Check the Condition first If true, execute the loop body; else terminate the loop body



The while loop

- It is conditional loop statement. It used to execute a statement or set of statements as long as the given condition is remain true.

The syntax of the while loop is :

while (condition)
statement ;

Where
Condition: it consists of relational expression. If it is true, the statement given in the while loop is executed.

Statement: it represents the body of the loop the compound statements or a set of statements are written in braces. { }.

The while loop

while (condition)
statement ;

- When the while loop statement is executed, the computer first evaluates the given condition. If the given condition is true , the statement in the body of the while executed.

- The body of the loop must contain a statement so that the condition on the loop becomes false during the execution of the loop body.

- If the condition of the loop never becomes false, the loop never ends. It becomes an infinite loop.

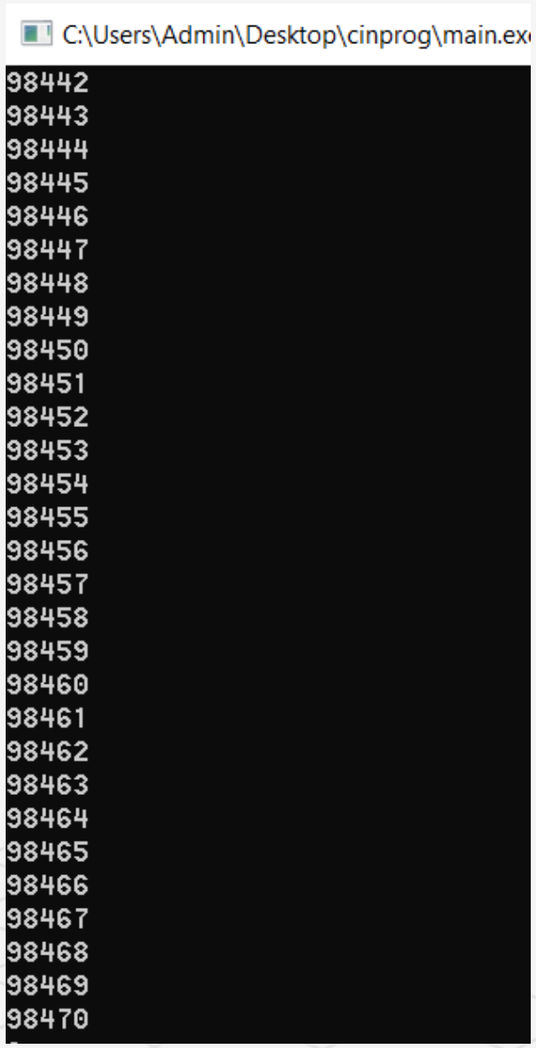
The while loop (Infinite)

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



int main() {
    int i = 0;
while (true) {
   cout << i << "\n";
   i++;
}

    return 0;
}
```



The while loop

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


int main() {
    int i = 0;
while (i<5) {
   cout << "i is now " <<i <<endl;
   i++;
}

    return 0;
}
```

```
C:\Users\Admin\Desktop\cinprog\main.exe
i is now 0
i is now 1
i is now 2
i is now 3
i is now 4

Process returned 0 (0x0)   execution time : 0.046 s
Press any key to continue.
```

The while loop

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


int main() {
    int i = 4;
while (i<5) {
  cout << "i is now " <<i <<endl;
  i++;
}

   return 0;
}
```

```
C:\Users\Admin\Desktop\cinprog\main.exe
i is now 4

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

The while loop

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


int main()
 {
     int n =10;
     while (n!=0){

     cout <<"n is now : "<<n <<endl;
     n =n-1;
}

return 0;
}
```

```
C:\Users\Admin\Desktop\cinprog\main.exe
n is now : 10
n is now : 9
n is now : 8
n is now : 7
n is now : 6
n is now : 5
n is now : 4
n is now : 3
n is now : 2
n is now : 1

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

The while loop

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


int main()
 {
     int n =10;
     while (n!=0){

     cout <<"n is now : "<<n <<endl;
     n =n-1;
}

return 0;
}
```

```
C:\Users\Admin\Desktop\cinprog\main.exe
n is now : 10
n is now : 9
n is now : 8
n is now : 7
n is now : 6
n is now : 5
n is now : 4
n is now : 3
n is now : 2
n is now : 1

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

The while loop

This program works fine if a positive number is entered, but in case of a nonpositive number, the while loop keeps execution for an unlimited number of times, and that's the default behavior of the loop.

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

int main() {
    int number;
    cout << "Enter a positive number: ";
    cin >> number;
    // Loop until a valid positive number is entered.
    while (number <=0) {
            cout << "Invalid input. Please enter a positive number: "<<endl;
            cin >>number;
            cout << "You entered: " << number << endl;
    }

    return 0;
}
```

Output

```
Enter a positive number: 1


=== Code Execution Successful ===
```

The while loop

From the picture below, it is quite clear that the while loop will only terminate if we enter a positive number, otherwise, it will keep executing…….

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

int main() {
    int number;
    cout << "Enter a positive number: ";
    cin >> number;
    // Loop until a valid positive number is entered.
    while (number <=0) {
            cout << "Invalid input. Please enter a positive number: "<<endl;
            cin >>number;
            cout << "You entered: " << number << endl;
    }

    return 0;
}


```

Output

```
Enter a positive number: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
0
You entered: 0
Invalid input. Please enter a positive number:
```

The while loop allows only three attempts or terminates it by entering a positive number

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

int main() {
    int number;
    int limit = 3;        // Maximum number of attempts
    int limit_count = 0; // Counter for attempts

    cout << "Enter a positive number: ";
    cin >> number;
    // Loop until a valid positive number is entered or the limit is reached
    while (number <=0)
    {
        if (limit_count < limit)
        {
            cout << "Invalid input. Please enter a positive number: ";
            cin >> number;
            limit_count++;

            // Print a warning message when limit_count is 2 (one attempt left)
            if (limit_count == 2)
            {
                cout << "Warning: You have only 1 attempt left!" << endl;
            }
        }
        else
        {
            cout << "Limit exceeded. The limit is " << limit << " attempts." << endl;
            break;
        }
    }

    if (number > 0)
    {
        cout << "You entered: " << number << endl;
    }

    return 0;
}
```

```
C:\Users\Admin\Desktop\while_loop\main.exe
Enter a positive number: 0
Invalid input. Please enter a positive number: 0
Invalid input. Please enter a positive number: 0
Warning: You have only 1 attempt left!
Invalid input. Please enter a positive number: 0
Limit exceeded. The limit is 3 attempts.

Process returned 0 (0x0)   execution time : 10.044 s
Press any key to continue.
```

We have seen in the previous example that the while loop keeps executing utill we enter to any positive number to falsify the given condition. In this program, we have made some changes to only allow three 3 attempts, with out entering any positive number.

The while loop to validate Username and Password.

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

int main() {

    while (true) {

        cout << "Enter your username: ";
        string name="abasyn";
        string u_name;

        getline(cin, u_name);

        if (u_name != name) {
            cout << "No, this is not you." << endl;
            continue; // Skip the rest of the loop and start over
        }

        cout << "Enter your password: ";
        string password ="abasyn@123";
        string u_password;
        getline(cin, u_password);

        if (u_password == password) {
            cout << "Congrats!!! Access granted." << endl;
            break;
        } else {
            cout << "Incorrect username or password. Try again." << endl;

        }
    }

    return 0;
}
```

```
C:\Users\Admin\Desktop\NamePassword\main.exe
Enter your username: abasyn
Enter your password: abasyn@123
Congrats!!! Access granted.

Process returned 0 (0x0)   execution time : 11.365 s
Press any key to continue.
```

This program is technically correct, but there is no restriction on how many times a user can try to log in. refer to the next slides to learn how to restrict a user for few attempts.

The while loop to validate Username and Password.

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

int main() {
    int limit = 3;
    int count_limit = 0;

    while (true) {
        if (count_limit == limit) {
            cout << "You have reached the maximum number of attempts. Access denied." << endl;
            break;
        }
        cout << "Introduce yourself please: " << endl;
        cout << "Enter your username: ";
        string username;
        getline(cin, username);

        if (username != "abasyn") {
            cout << "No, this is not you." << endl;
            count_limit++;
            continue; // Skip the rest of the loop and start over
        }

        cout << "Enter your password: ";
        string password;
        getline(cin, password);

        if (password == "abasyn@123") {
            cout << "Congrats!!! Access granted." << endl;
            break;
        } else {
            cout << "Incorrect password. Try again." << endl;
            count_limit++;
        }
    }

    return 0;
}
```

```
C:\Users\Admin\Desktop\NamePassword\main.exe
Introduce yourself please:
Enter your username: abasyn
Enter your password: abasyn@123
Congrats!!! Access granted.

Process returned 0 (0x0)   execution time : 11.056 s
Press any key to continue.
```

This program is technically ok, if but instead of assigning the username “abasyn” and password to variables, we have t assign the values manually, which is not good practice (hardcoded) Refer to a lecture on **why we need variables.**

The while loop to validate Username and Password

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

int main() {
    int limit = 3;
    int count_limit = 0;

    while (true) {
        if (count_limit == limit) {
            cout << "You have reached the maximum number of attempts. Access denied." << e
            break;
        }
        cout << "Introduce yourself please: " << endl;
        cout << "Enter your username: ";
        string name="abasyn";
        string u_name;

        getline(cin, u_name);

        if (u_name != name) {
            cout << "No, this is not you." << endl;
            count_limit++;
            continue; // Skip the rest of the loop and start over
        }

        cout << "Enter your password: ";
        string password ="abasyn@123";
        string u_password;
        getline(cin, u_password);

        if (u_password == password) {
            cout << "Congrats!!! Access granted." << endl;
            break;
        } else {
            cout << "Incorrect password. Try again." << endl;
            count_limit++;
        }
    }
```

```
C:\Users\Admin\Desktop\NamePassword\main.exe
Introduce yourself please:
Enter your username: ABASYN
No, this is not you.
Introduce yourself please:
Enter your username: Abasyn
No, this is not you.
Introduce yourself please:
Enter your username: ABASYn
No, this is not you.
You have reached the maximum number of 3 attempts. Access denied.

Process returned 0 (0x0)   execution time : 23.158 s
Press any key to continue.
```

```
C:\Users\Admin\Desktop\NamePassword\main.exe
Introduce yourself please:
Enter your username: abasyn
Enter your password: abasyn@123
Congrats!!! Access granted.

Process returned 0 (0x0)   execution time : 9.785 s
Press any key to continue.
```

The if statement checks if u_name is not equal to name.

Continue How does it work ???

```
if (u_name != name) {
    cout << "No, this is not you." << endl;
            count_limit++;
            continue;
}

        cout << "Enter your password: ";
        string password ="abasyn@123";
        string u_password;
        getline(cin, u_password);
```

- The continue statement skips the rest of the code in the current iteration of the loop and immediately moves to the next iteration.
- It does not exit the loop; it just skips the remaining code for the current iteration and re-evaluates the loop condition to start the next iteration.

- The if statement checks if u_name is not equal to name.
- If the condition is true (u_name != name), the code inside the if block is executed.
- continue; is executed, which skips the rest of the code in the current iteration of the loop.

The if statement checks if u_name is not equal to name.

Boolean Datatype

- The boolean data type in C++ is used to store true/false values. It's represented by the bool keyword and can only take two possible values:
- true (which is internally represented as 1)
- false (which is internally represented as 0)

```cpp
#include <iostream>

using namespace std;

int main()
{
    bool is_in_islamabad = false;
    bool is_student =false ;
    if (is_in_islamabad && is_student){
        cout <<"Your are living in islamabad and living in islamabad "<<endl;
    }
    else if (is_in_islamabad && ! is_student){
        cout <<"You are in islamabad but you are not student"<<endl;
    }
    else if (!is_in_islamabad && is_student){
        cout <<"you are not in islamabad but you are student"<<endl;
    }
    else {
        cout <<"You are neither in islamabad nor a student"<<endl;
    }
    return 0;
}
```

```
C:\Users\Admin\Desktop\boolean\bin\Debug\boolean.exe
You are neither in islamabad nor a student

Process returned 0 (0x0)   execution time : 0.014 s
Press any key to continue.
```

The if statement checks if u_name is not equal to name.

Guessing game.

```cpp
#include <iostream>

using namespace std;

int main() {
    string mySecret = "Moon";
    int guess_count = 0;
    string guess;
    int guess_limit = 3;
    bool out_of_guess = false;

    while (guess != mySecret && !out_of_guess) {
        if (guess_count < guess_limit) {
            cout << "Guess my secret: ";
            cin >> guess;

            guess_count++;
        if (guess_count ==2)
        {
        cout <<"Last warning"<<endl;
       }
        }
         else {
            out_of_guess = true;
        }
    }

    if (out_of_guess) {
        cout << "Nah, it is not !!!" << endl;
    } else {
        cout << "Yeah !!! you got it " << endl;
    }

    return 0; // Success
}
```

C:\Users\Admin\Desktop\guessinggame\main.exe

```
Guess my secret: moon
Guess my secret: star
Last warning
Guess my secret: nature
Nah, it is not !!!

Process returned 0 (0x0)   execution time : 24.513 s
Press any key to continue.
```

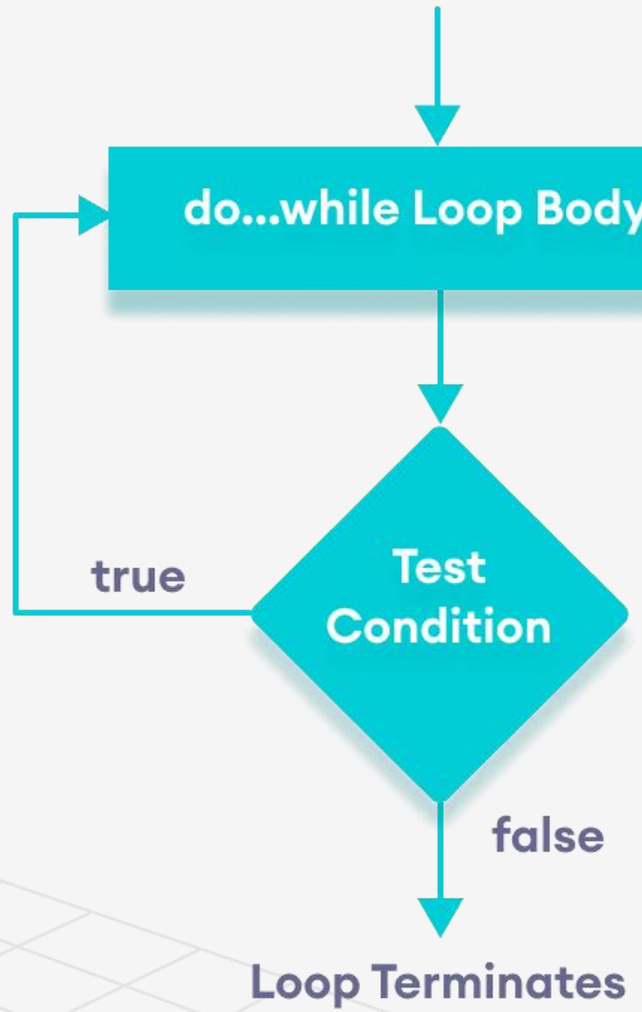
C:\Users\Admin\Desktop\guessinggame\main.exe

```
Guess my secret: Moon
Yeah !!! you got it

Process returned 0 (0x0)   execution time : 3.464 s
Press any key to continue.
```

Flowchart of do while Loop

Execute the loop body, then check the condition. If it is true, loop through the loop body; otherwise, terminate the loop body.



The do while loop

- The **do-while** loop is also a conditional statement. It is like a while loop, but this loop condition is tested after executing the loop body.

The syntax is :

```
 do
{
Statement ;
While (condition);
```

do→ The keyword do indicates the starting of the do-while loop.
The statement → enclosed in braces { } represents the body of the loop.
Condition → It is the condition that must remain true for the execution of the loop body.

The Do While First Program

```cpp
#include <iostream>

using namespace std;

int main() {
    int i = 1;
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);

    return 0;
}
```

```
1 2 3 4 5

=== Code Execution Successful ===
```

The Condition is true (i<=5), and the program executed successfully

The Do While “The loop body execute at least Once”

```cpp
#include <iostream>

using namespace std;

int main() {
    int i = 10;

    // do...while loop from 1 to 5
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);

    return 0;
}
```

```
Output

10

=== Code Execution Successful ===
```

Although i =10 and the given condition is (i<=5), in this case, the condition is false because i >5, but the loop body gets executed.

The Do While “The loop body execute at least Once”

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

int main() {
    int marks ;
    char name[15], address [15], op;
    do {

        cout <<"Enter student name"<<endl;
        cin>>name;
        cout <<"Enter address"<<endl;
        cin>> address;
        cout <<"Enter Obtained marks"<<endl;
        cin>>marks;
        cout<<"student record is: "<<endl;
        cout <<name <<"\t" <<address<<"\t" <<marks<<endl;
        cout <<"more record [Y/N]"<<endl;
        cin >>op;
    }
    while (op =='y' || op =='Y');
    cout <<"Ok";

     return 0;
}
```

```
Enter student name
Ali
Enter address
Islamabad
Enter Obtained marks
800
student record is:
Ali Islamabad   800
more record [Y/N]
```

A good example of do-while loop as can be seen in the program. That loop will start executing if we enter ‘y’ or ‘Y’, but since we know that the do-while once execute before checking the condition. So it executes the “do” part before checking the while part, which is a condition for executing. In this case, the while loop starts execution if we supply/enter ‘y’ or ‘Y’. After completing the first execution again it will need an input (‘y’ or ‘Y’).

Write a program to find factorial of a number

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

int main() {
    int number;
    int factorial = 1;
    cout << "Enter a positive integer: ";
    cin >> number;
    if (number < 0) {
        cout << "Factorial is not defined for negative numbers." << endl;
    } else {
        // Calculate factorial using a while loop
        int i = number;
        while (i > 1) {
            //variable = variable * expression;
            //x *= 3;  // Equivalent to: x = x * 3;
            factorial *= i;  // Multiply factorial by i
            //factorial = factorial * i i.e.,4,3,2,1
            i--;
            cout <<factorial<<endl;
            //cout<<i;
        }
        cout << "Factorial of " << number << " is " << factorial << endl;
    }

    return 0;
}
```

```
C:\Users\Admin\Desktop\factorail\main.exe
Enter a positive integer: 4
4
12
24
Factorial of 4 is 24

Process returned 0 (0x0)   execution time : 2.010 s
Press any key to continue.
```

Assignment # 2
Deadline next Wednesday

Write a program to calculate the electricity bill. The rates of the electricity per unit are as follows:

- If the unit consumed is less than or equal to 200, the cost is Rs 65/per unit.

- If the units consumed are more than 200, the cost is Rs 75/per unit. and a surcharge of 5% of the bill is added.

Thanks