CSE 232

Programming with C++

Lecture 6

Control Structures (2)

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
Email: mijan@jkkniu.edu.bd
Web: www.mijanrahman.com

Contents

CONTROL STRUCTURES

- Decision making and looping (Iteration) statement
  - For loop
  - Nested for loop
  - Rage-based for loop
  - Infinite loop
  - While loop
  - Do…While loop

Looping in C++

- In Programming, sometimes there is a need to perform some operation more than once or (say) n number of times. Loops come into use when we need to repeatedly execute a block of statements.

- For example: Suppose we want to print “Hello World” 10 times. This can be done in two ways:
  - Manual Method (Iterative Method)
  - Using Loops

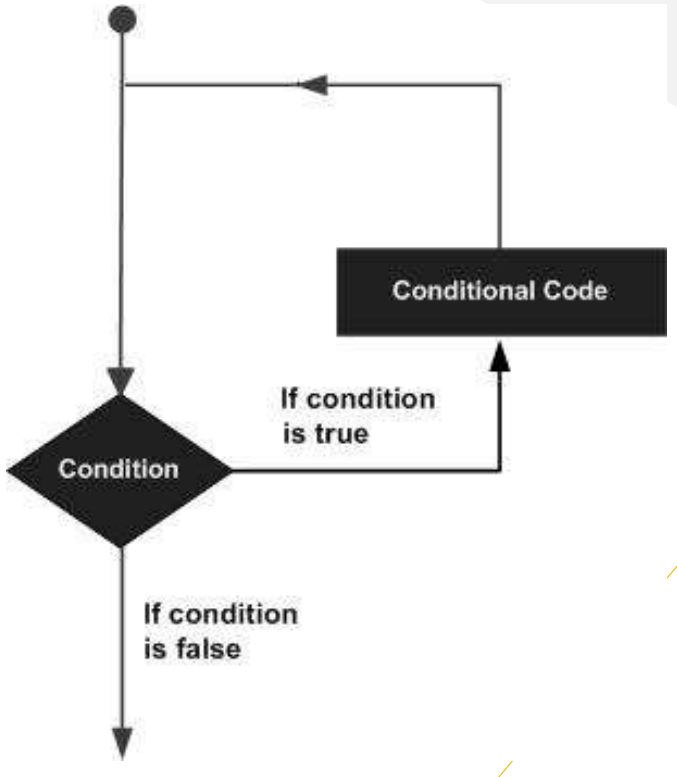
Manual Looping (Iteration)

- Manually we have to write cout for the C++ statement 10 times.
- Let's say you have to write it 20 times (it would surely take more time to write 20 statements) now imagine you have to write it 100 times, it would be really hectic to re-write the same statement again and again. So, here loops have their role.

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      cout << "Hello World\n";
7.      cout << "Hello World\n";
8.      cout << "Hello World\n";
9.      cout << "Hello World\n";
10.     cout << "Hello World\n";
11.     return 0;
12. }
```

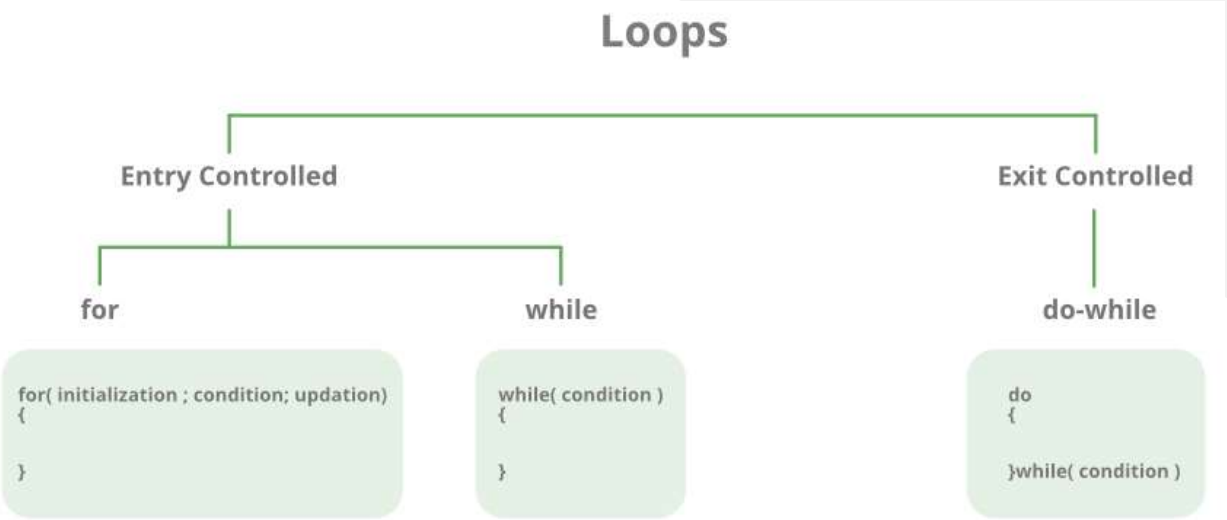
Looping (Iteration) Statements

- In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.



Looping (Iteration) Statements

- There are mainly two types of loops:
  1. **Entry Controlled loops:** The test condition is tested before entering the loop body. For Loop and While Loop is entry-controlled loops.
  2. **Exit Controlled Loops:** The test condition is tested or evaluated at the end of the loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. the do-while loop is exit controlled loop.



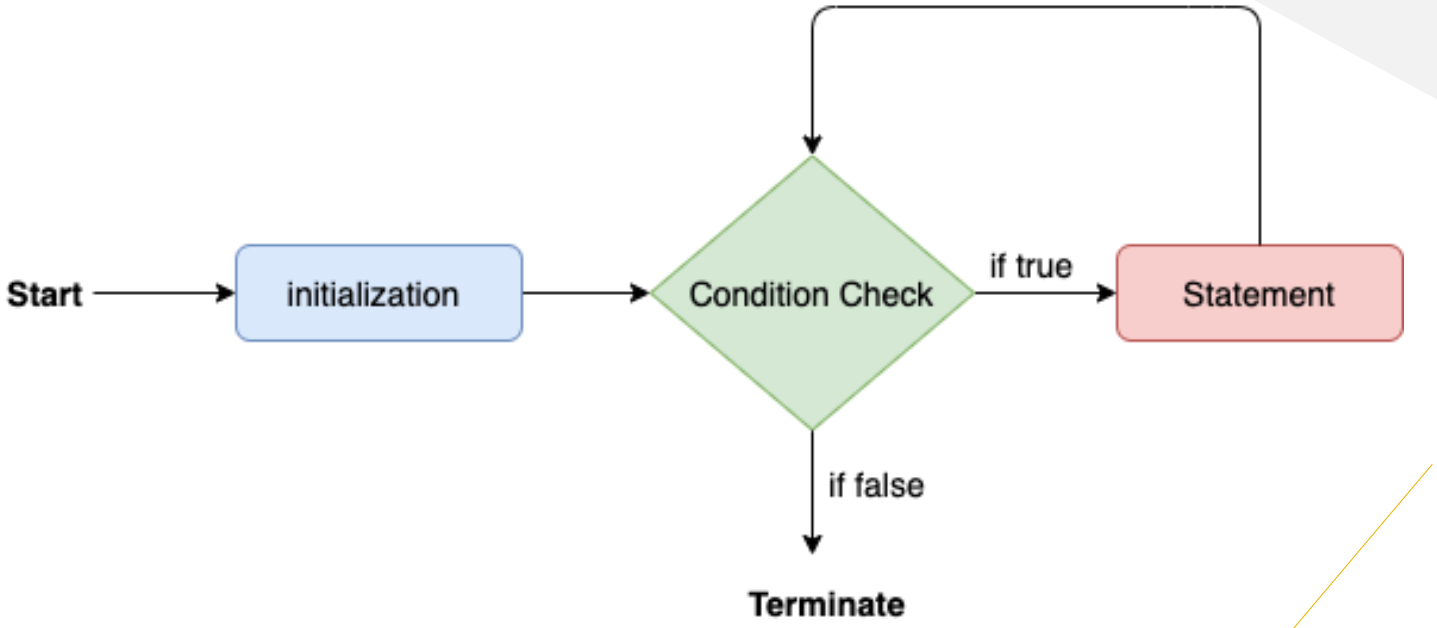
For Loop

- The **For Loop Statement** is an iteration statement that executes a set of code repeatedly, given the initial value, the condition, increment value.
- It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.
- **Syntax:**

```
for(initialization, condition, increment/decrement) {
//block of statements
}
```

For Loop

- The flow chart for the for-loop is given below.



For Loop

- **Example:**

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

int main()
{

   int n = 5;
   int i;
   for (i = 1; i <= n; i++) {
      cout << i << " ";
   }

   return 0;
}
```

For Loop

- **Example:**

```
int main()
{
        int sum = 0;
        for(int j = 1; j<=10; j++) {
                sum = sum + j;
        }
cout<<"The sum of first 10 natural numbers is “<<sum;
return 0;
}
```

- **Output:**

```
The sum of first 10 natural numbers is 55
```

C++ Nested for Loop

- A nested for loop is basically putting one loop inside another loop. Every time the outer loop runs, the inner loop runs all the way through. It’s a way to repeat tasks within tasks in your program.
- **Example of Nested for Loop**

```
int main()
{
   for (int i = 0; i < 4; i++) {
      for (int j = 0; j < 4; j++) {
         cout << "*“ << " ";
      }
      cout << endl;
   }
}
```

Multiple Variables in for Loop

- In the for loop we can initialize, test, and update multiple variables in the for loop.

- **Example of Using Multiple Variables in for Loop**

```
int main()
{
    int m, n;
    for (m = 1, n = 1; m <= 5; m += 1, n += 2) {
      cout << "iteration " << m << endl;
      cout << "m is: " << m << endl;
      cout << "j is: " << n << endl;
   }
   return 0;
}
```

C++ Infinite for Loop

- When no parameters are given to the for loop, it repeats endlessly due to the lack of input parameters, making it a kind of infinite loop.
- **Example of Infinite for Loop**

```
int main()
{
   // skip Initialization, test and update conditions
   for (;;) {
      cout << “CSE" << endl;
   }
   return 0;
}
```

Range-based for loop

- Range-based for loop in C++ has been added since C++ 11. It executes a for loop over a range. Used as a more readable equivalent to the traditional for loop operating over a range of values, such as all elements in a container.

```
for ( range_declaration : range_expression )
   loop_statement
```

Parameters:

range_declaration:
a declaration of a named variable, whose type is the type of the element of the sequence represented by range_expression, or a reference to that type.
Often uses the auto specifier for automatic type deduction.

range_expression:
any expression that represents a suitable sequence
or a braced-init-list.

loop_statement:
any statement, typically a compound statement, which
is the body of the loop.

Range-based for loop

- Example of Range-Based for Loop

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

int main()
{
    int arr[] = { 10, 20, 30, 40, 50 };

    cout << "Printing array elements: " << endl;
    // range based for loop
    for (int& val : arr) {
        cout << val << " ";
    }
    cout << endl;

    cout << "Printing vector elements: " << endl;
    vector<int> v = { 11, 22, 33, 44, 55 };

    // range-based for loop for vector
    for (int& it : v) {
        cout << it << " ";
    }
    cout << endl;

    return 0;
}
```

for_each Loop in C++

- C++ for_each loop accepts a function that executes over each of the container elements.
- **Syntax of for_each Loop**

```
for_each (start_iter, last_iter, fnc);
```

- Parameters
  - start_iter: Iterator to the beginning position from where function operations have to be executed.
  - last_iter: Iterator to the ending position till where function has to be executed.
  - fnc: The 3rd argument is a function or a function object which would be executed for each element.

for_each Loop in C++

- Example of for_each Loop

```
#include <bits/stdc++.h>
using namespace std;

// function to print values passed as parameter in loop
void printValues(int i) {
    cout << i << " " << endl;

}

int main()
{
    // initializing vector
    vector<int> v = { 11, 22, 33, 44, 55 };

    // iterating using for_each loop
    for_each(v.begin(), v.end(), printValues);

    return 0;
}
```

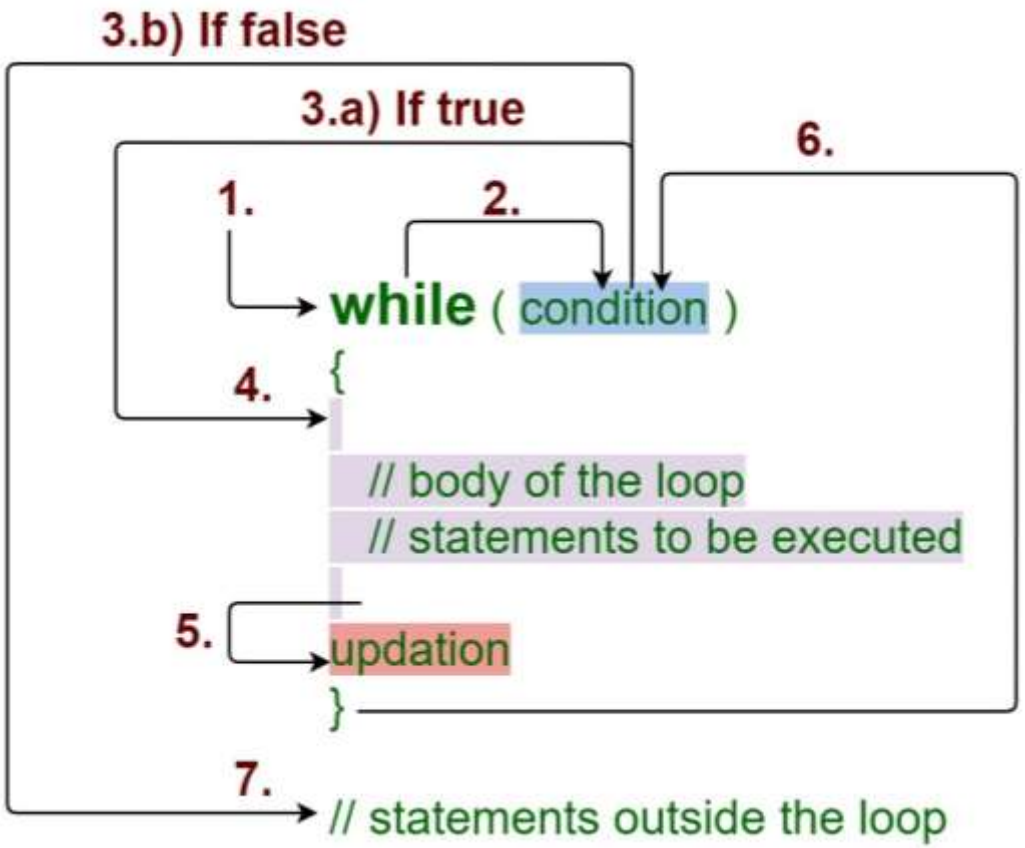
While Loop

- **while** statement is an iteration structure that executes a set of code only when the condition is true. It is also known as "Entry controlled loop" since it is executed only inside the loop if the condition is satisfied. It ends with a ";" after the closing braces.
- **Syntax:** The syntax of the while loop is given below.

```
while(condition){
//looping statements
}
```

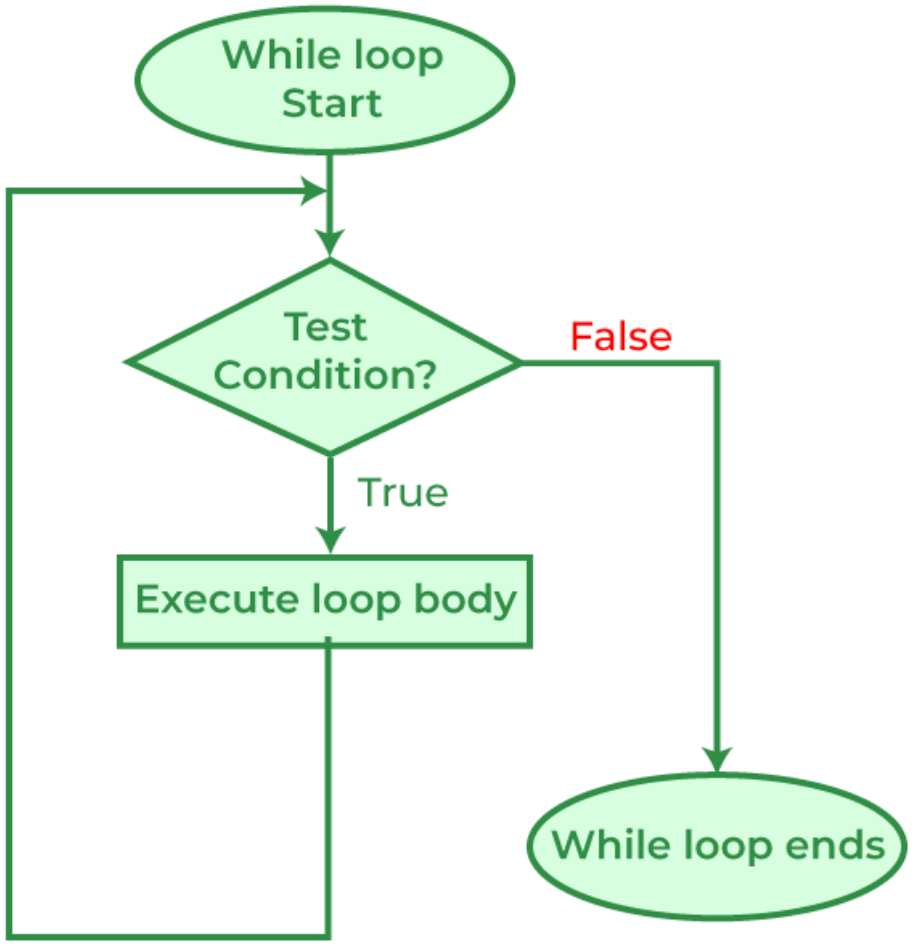
While Loop

- **Syntax:** The syntax of the while loop is given below.



While Loop

- The flow chart for the while loop:



While Loop

- Example:

```
int i = 0;
cout << "Printing the list of even numbers (<=10): ";
while(i<=10) {
        cout << i << “  ”;
        i = i + 2;
}
```

- **Output:**

Printing the list of first 10 even numbers: 0 2 4 6 8 10

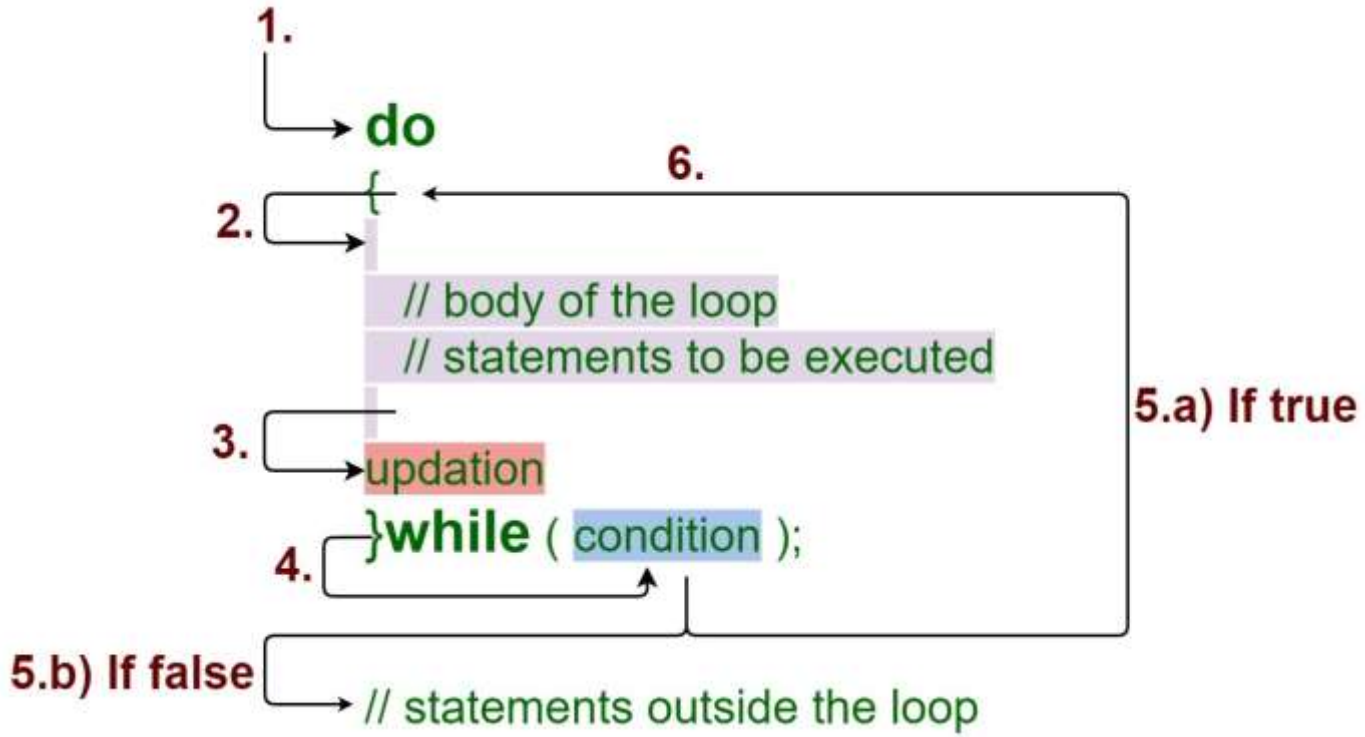
Do…While Loop

- The **Do While** statement is an iteration statement that executes a set of code first, then it checks the condition. This statement is also known as **"Exit controlled loop"** since the condition is checked only after executing the loop at least once.
- **Syntax:** The syntax of the do…while loop is given below.

```
do
{
//statements
} while (condition);
```

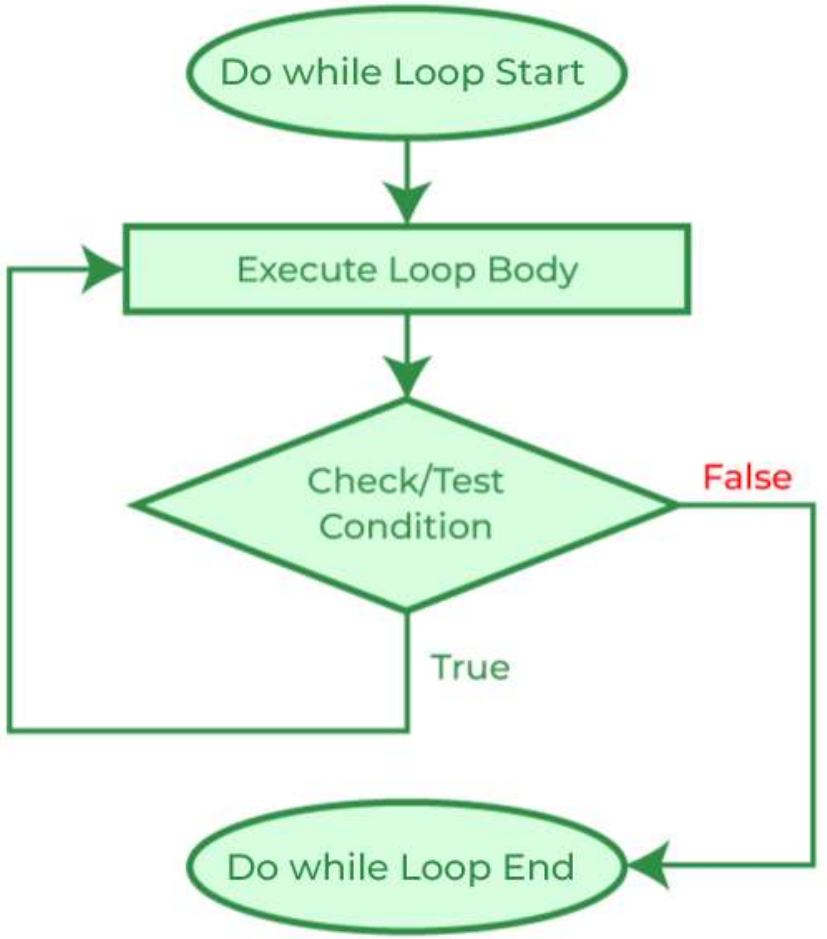
Do…While Loop

- **Syntax:** The syntax of the do…while loop is given below.



Do…While Loop

- The flow chart of the do-while loop:



Do…While Loop

- **Example:**

```
int i = 0;
cout << "Printing the list of even numbers (<=10): ";
do {
        cout << i << “  ”;
        i = i + 2;
}  while(i<=10);
```

- **Output:**

Printing the list of first 10 even numbers: 0 2 4 6 8 10

C++ Programs

- **Printing the multiplication table of a given number:**

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

int main() {
    int number, limit;

    cout << "Enter a number for the multiplication table: ";
    cin >> number;
    cout << "Enter the limit for the multiplication table: ";
    cin >> limit;

    // Print the multiplication table
    cout << "Multiplication table for " << number << ":\n";
    for (int i = 1; i <= limit; ++i) {
        cout << number << " x " << i << " = " << number * i << endl;
    }

    return 0;
}
```

```
Enter a number for the multiplication table: 9
Enter the limit for the multiplication table: 10
Multiplication table for 9:
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

C++ Programs

- Below is a simple C++ program that reads positive numbers from the user and sums them until the user enters a negative number, at which point it will print the total sum.

```
Enter positive numbers (a negative number to stop):
Enter a number: 10
Enter a number: 20
Enter a number: 30
Enter a number: -10
Total sum of positive numbers: 60
```

```cpp
#include <iostream>
using namespace std;
int main() {
    double number, sum = 0;

    cout << "Enter positive numbers (a negative number to stop):\n";

    while (true) {
        cout << "Enter a number: ";
        cin >> number;

        // Check if the number is negative
        if (number < 0) {
            break; // Exit the loop if a negative number is entered
        }

        // Add the positive number to the sum
        sum += number;
    }

    cout << "Total sum of positive numbers: " << sum << endl;

    return 0;
}
```

C++ Programs

- **Finding the factorial of a number:**

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

int main() {
    int number;
    unsigned long long factorial = 1;
    cout << "Enter a positive integer to find its factorial: ";
    cin >> number;

    // Check for negative input
    if (number < 0) {
        cout << "Factorial is not defined for negative numbers." << endl;
    } else {
        int i = 1;
        do {
            factorial *= i;
            i++;
        } while (i <= number);

        cout << "Factorial of " << number << " = " << factorial << endl;
    }

    return 0;
}
```

C++ Programs

- C++ program that generates the Fibonacci sequence up to a specified number of terms:

```
Enter the number of terms for the Fibonacci sequence: 10
Fibonacci sequence up to 10 terms:
0 1 1 2 3 5 8 13 21 34
```

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

int main() {
    int terms;

    cout << "Enter the number of terms for the Fibonacci sequence: ";
    cin >> terms;

    if (terms <= 0) {
        cout << "Please enter a positive integer." << endl;
    } else {
        int first = 0, second = 1;

        cout << "Fibonacci sequence up to " << terms << " terms:\n";

        for (int i = 1; i <= terms; i++) {
            cout << first << " "; // Print the current term
            int next = first + second; // Calculate the next term
            first = second; // Update first to the next term
            second = next; // Update second to the next term
        }
        cout << endl;
    }

    return 0;
}
```

Lecture 6

C++ Control Structures (2)

THE END