University of Batna 2
Institute of safety and security
1st year LMD

Chapter 4: C++ Loops

Introduction

- In computer programming, loops are used to repeat a block of code. For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.
- That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.
- There are 3 types of loops in C++:
  - for loop
  - while loop
  - do...while loop

For Loop

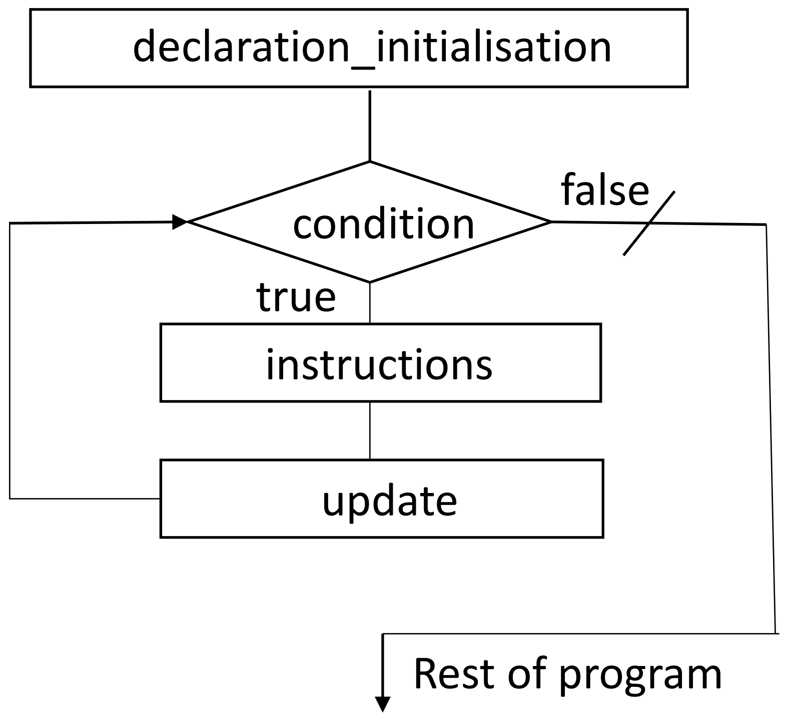
- A For loop is a repetition control structure that allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.
- **Syntax:**

```
for (declaration_initialisation; condition; update) {
      instructions;
   }
```

- `declaration_initialisation` : declares and initializes variable and is executed only once
- `condition:` if true, the body of for loop is executed if false, the for loop is terminated
- `Update`: updates the value of initialized variables and again checks the condition

For Loop

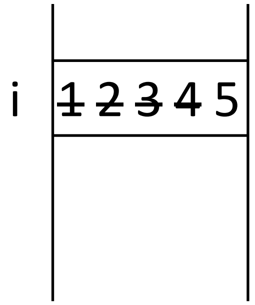
- **Flowchart:**

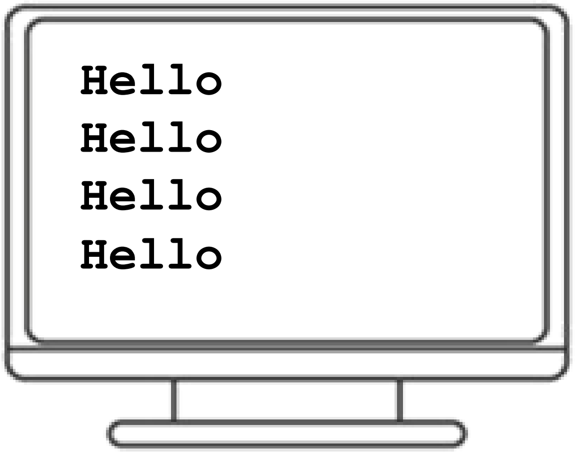


For Loop

- Example:

```
#include <iostream>
using namespace std;
int main() {
 int i;
 for(i=1;i<5;i=i+1)
 cout<<“Hello"<<endl;
 return 0;
}
```





```
Hello
Hello
Hello
Hello
```

For Loop

- **Notes:**
  - The initialization and increment statements can perform operations unrelated to the condition statement, or nothing at all – if you wish to do. But the good practice is to only perform operations directly relevant to the loop.
  - A variable declared in the initialization statement is visible only inside the scope of the for loop and will be released out of the loop.
  - Don’t forget that the variable which was declared in the initialization statement can be modified during the loop, as well as the variable checked in the condition.
  - Don’t write a semicolon after the end of parenthesis.

C++ while Loop

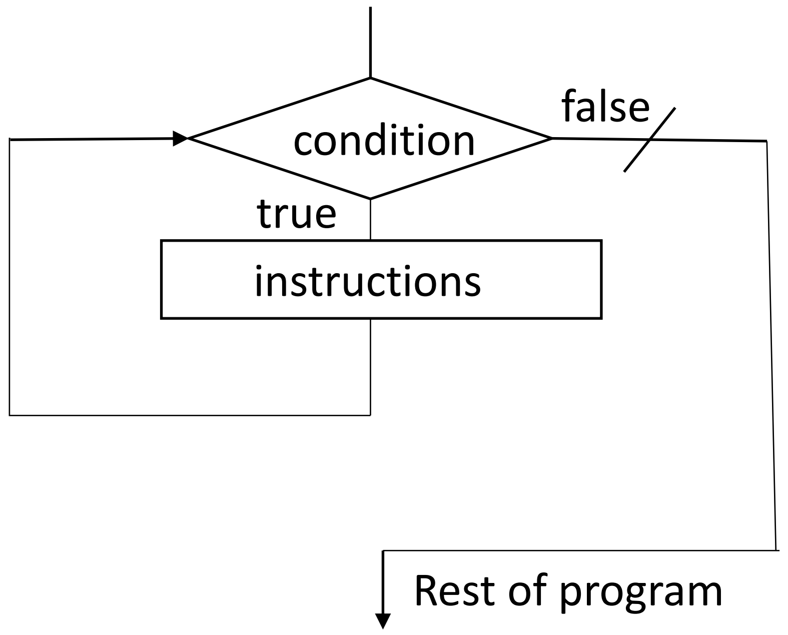
Syntax:

```
while (condition){
      instructions;
  }
```

- A while loop evaluates the condition
- If the condition evaluates to true, the code inside the while loop is executed.
- The condition is evaluated again.
- This process continues until the condition is false.
- When the condition evaluates to false, the loop terminates.

C++ while Loop

- Flowchart:



C++ while Loop

- Example:

```
#include <iostream>
using namespace std;
int main()
{
 int i=0;
 while(i<10)
 {
 cout<<"i="<<i<<endl;
 i++;
 }
 cout<<"The last value of i is:
"<<i<<endl;
 return 0;
}
```

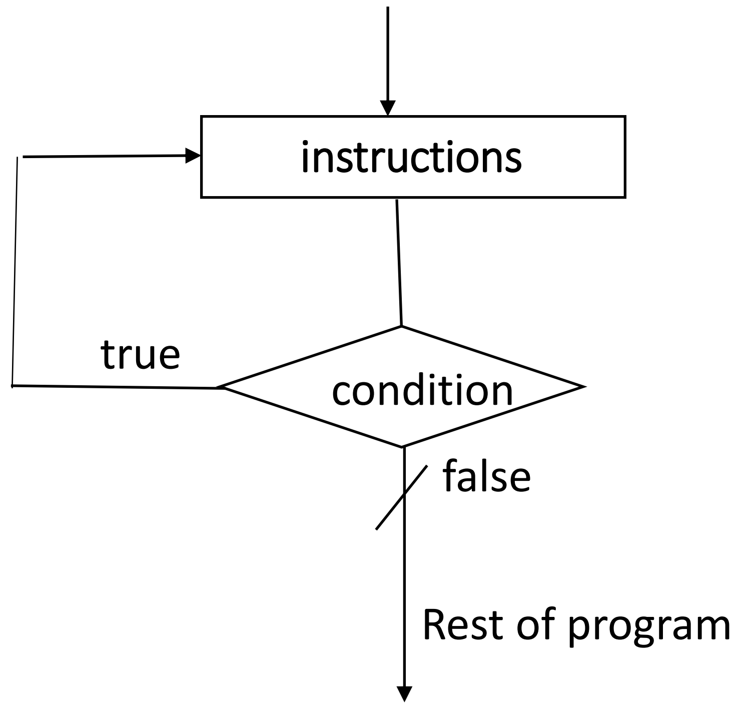
C++ do...while Loop

- The do...while loop is a variant of the while loop with one important difference: the body of do...while loop is executed once before the condition is checked.
- **Syntax:**

```
do{
    instructions;
}while(condition);
```

C++ do...while Loop

- **Flowchart**



C++ do...while Loop

- **Example:**

```
#include <iostream>
using namespace std;
int main()
{
 int pass;
 do{
 cout<<“Please enter passe word";
 cin>>pass;
 }while(pass !=10);
 return 0;
}
```

C++ Infinite loop

- If the condition in a loop is always true, it runs forever (until memory is full). For example (using for loop):

```
for(int i = 1; i > 0; i++) {
        instructions;
}
```

- In the above program, the condition is always true which will then run the code for infinite times.

Nested Loops:

- It is also possible to place a loop inside another loop. This is called a nested loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop“, for example:

```
for (int i = 1; i <= 2; ++i) {
       cout << "Outer: " << i << endl;
       for (int j = 1; j <= 3; ++j) {
              cout << " Inner: " << j << endl;
       }
}
```