C++ Foundation with Data Structures

Lecture 3 : Conditionals and Loops

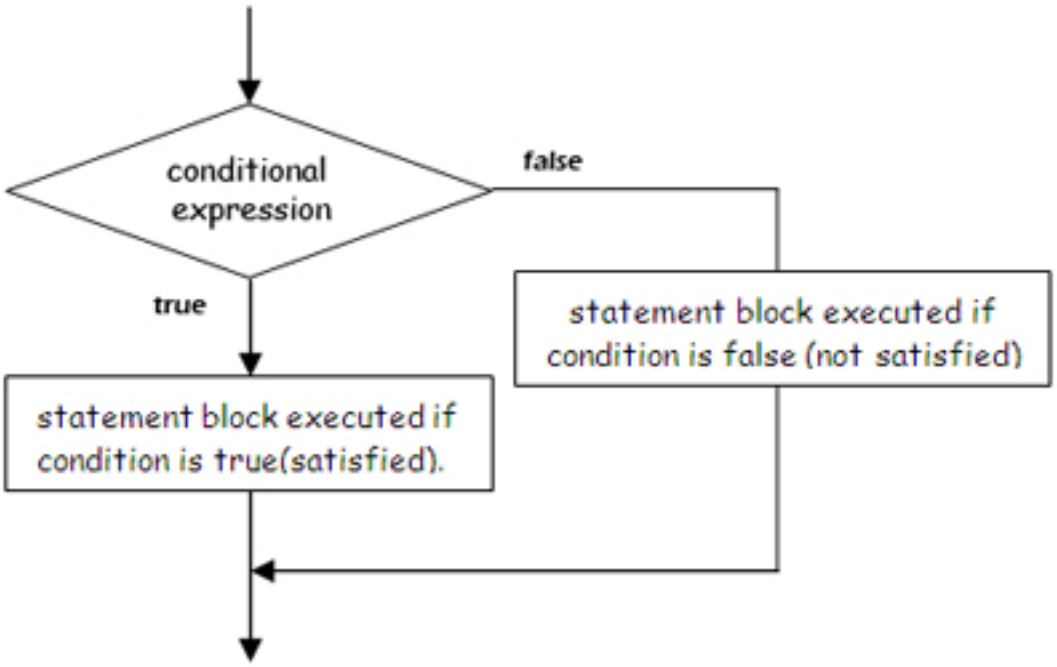
Points :

- cin>>a>>b>>c;
- cout<<sizeof(char);
- ! Negates the value
- int/int = int
- float/int = float
- % can't be used with double or float
- # include<cmath>
  round(....)

Conditional Statements (if else)

Description

Conditionals are used to execute a certain section of code only if some specific condition is fulfilled, and optionally execute other statements if the given condition is false. The result of given conditional expression must be either true or false.



Different variations of this conditional statement are –

- **if statement**

  if statement evaluates the given test expression. If it is evaluated to true, then statements inside the if block will be executed. Otherwise, statements inside if block is skipped.

  Syntax

```
if(test_expression) {
        // Statements to be executed only when test_expression is true
}
```

  Example Code:

```
int main ()
{
	int n=5;
	if ( n<10 )
	{
		cout << "Inside if statement" << endl;
	}
	cout << "Outside if statement" << endl;
}
```

Output

```
Inside if statement
Outside if statement
```

So if the condition given inside if parenthesis is true, then statements inside if block are executed first and then rest of the code. And if the condition evaluates to false, then statements inside if block will be skipped.

- **If – else statement**

if statement evaluates the given test expression. If it is evaluated to true, then statements inside the if block will be executed. Otherwise, statements inside else block will be executed. After that, rest of the statements will be executed normally.

Syntax

```
if(test_expression) {
	// Statements to be executed when test_expression is true

}
else {
	// Statements to be executed when test_expression is false
}
```

Example Code:

```
int main () {
	int a=10,b=20;
	if (a > b){
		cout<< "a is bigger" <<endl;
```

```
        }
        else {
            cout<< "b is bigger" <<endl;
        }
    }
```

Output

b is bigger

- **if – else – if**

Using this we can execute statements based on multiple conditions.

Syntax

```
if(test_expression_1) {
    // Statements to be executed only when test_expression_1 is true
}
else if(test_expression_2) {
    // Statements to be executed only when test_expression_1 is false and test_expression_2 is true
}
else if(test_expression_2) {
    // Statements to be executed only when test_expression_1 & test_expression_2 are false and test_expression_3 is true
}
....
....
else {
    // Statements to be executed only when all the above test expressions are false
}
```

Out of all block of statements, only one will be executed based on the given test expression, all others will be skipped. As soon as any expression evaluates to ttue, that block of statement will be executed and rest will be skipped. If none of the expression evaluates to true, then the statements inside else will be executed.

Example Code:

```
int main() {
	int a = 5;
	if(a < 3) {
		cout<<"one"<< endl;
	}
	else if(a < 10) {
		cout<<"two"<< endl;
	}
	else if(a < 20) {
		cout<<"three"<< endl;
	}
	else {
		cout<<"four"<< endl;
	}
}
```

Output :

two

- **Nested if statement**

We can put another if – else statement inside an if.

Syntax

```
if(test_expression_1) {
	// Statements to be executed when test_expression_1 is true
	if(test_expression_2) {
		// Statements to be executed when test_expression_2 is true
	}
	else {
		// Statements to be executed when test_expression_2 is false
	}
}
```

Example Code:

```
int main() {
    int a = 15;
    if(a > 10) {
        if(a > 20) {
            cout<<"Hello"<<endl;
        }
        else {
            cout<<"Hi"<<endl;
        }
    }
}
```

Output :
Hi

return keyword

return is a special keyword, when encountered ends the main. That means, no statement will be executed after return statement. We'll study about return in more detail when we'll study functions.

Example Code:

```
int main() {
    int a = 10;
    if(a > 5) {
        cout<<Hello"<<endl;
        return 0;
    }
    cout<<"Hi"<<endl;
}
```

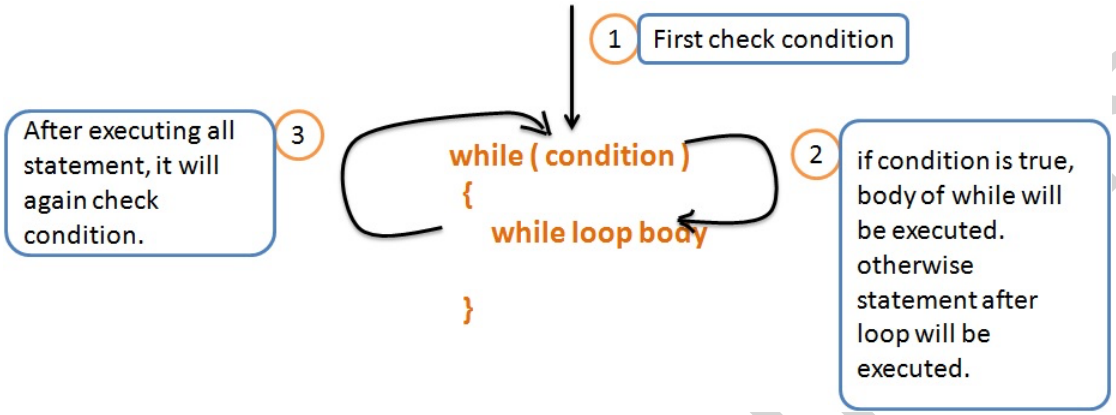
Output:
Hello

while loop

Loop statements allows us to execute a block of statements several number of times depending on certain condition. **while** is one kind of loop that we can use. When executing, if the *test expression* result is true, then the actions inside the loop will be executed. This will continue as long as the expression result is true.

Syntax

```
while(test_expression) {
        // Statements to be executed till test_expression is true
}
```



Example Code:

```
int main() {
        int i = 1;
        while(i <= 5) {
                cout<<i<<endl;
                i++;
        }
}
```

Output:

1
2
3
4
5

In while loop, first given test expression will be checked. If that evaluates to be true, then the statements inside while will be executed. After that, the condition will be checked again and the process continues till the given condition becomes false.