FLOW CONTROL

Sometimes we need to execute a block of statements only when a particular condition is met or not met. This is called **decision making**, as we are executing a certain code after making a decision in the program logic. For decision making in C++, we have four types of control statements (or control structures), which are as follows:

a) if statement
b) nested if statement
c) if-else statement
d) if-else-if statement

C++ supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

If statement in C++

If statement consists a condition, followed by statement or a set of statements as shown below:

```
if(condition){
  Statement(s);
}
```

The statements inside **if** parenthesis (usually referred as if body) gets executed only when the given condition is true. If the condition is false then the statements inside if body are completely ignored.

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text:

Example

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

int main()

{

if (20 > 18) {
  cout << "20 is greater than 18";
}

return 0;

}
```

We can also test variables:

Example

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

int main()

{

int x = 20;
int y = 18;
if (x > y) {
  cout << "x is greater than y";
}

return 0;

}
```

Example

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

int main()

{

int age = 50;
if (age <=50) {
  cout << "I am young";
}

return 0;

}
```

Nested if statement in C++

When there is an if statement inside another if statement then it is called the **nested if statement**.
The structure of nested if looks like this:

```
if(condition_1) {
   Statement1(s);

   if(condition_2) {
      Statement2(s);
   }
}
```

Statement1 would execute if the condition_1 is true. Statement2 would only execute if both the conditions(condition_1 and condition_2) are true.

Example of Nested if statement

```
#include <iostream>
using namespace std;
int main(){
   int num=90;

   if( num < 100 ){
      cout<<"number is less than 100"<<endl;

 if(num > 50){
         cout<<"number is greater than 50";
      }
   }
   return 0;
}
```

Output:

```
number is less than 100
number is greater than 50
```

If else statement in C++

Sometimes you have a condition and you want to execute a block of code if condition is true and execute another piece of code if the same condition is false. This can be achieved in C++ using if-else statement.

This is how an if-else statement looks:

```
if(condition) {
   Statement(s);
}
else {
   Statement(s);
}
```

The statements inside “if” would execute if the condition is true, and the statements inside “else” would execute if the condition is false.

Example

```
#include <iostream>
using namespace std;
int main(){


int time = 20;
if (time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
return 0;

}
```

Output: Good evening.

Example

```
#include <iostream>
using namespace std;
int main(){
   int num=66;
   if( num < 50 ){
      cout<<"num is less than 50";
   }
   else {
      cout<<"num is greater than or equal 50";
   }
   return 0;
}
```

Output:

```
num is greater than or equal 50
```

Example

```
#include <iostream>
using namespace std;
int main(){
  float mark;
  cout<<"Enter score:";
  cin>> mark;
  if( mark >= 50 ){
    cout<<"You have Passed";
  }
  else {
    cout<<"You have failed";
  }
  return 0;
}
```

if-else-if Statement in C++

if-else-if statement is used when we need to check multiple conditions. In this control structure we have only one “if” and one “else”, however we can have multiple “else if” blocks. This is how it looks:

```
if(condition_1) {
  statement(s);
}
else if(condition_2) {
  statement(s);
}
else if(condition_3) {
    statement(s);
}
else {
  statement(s);
}
```

Note: The most important point to note here is that in if-else-if, as soon as the condition is met, the corresponding set of statements get executed, rest gets ignored. If none of the condition is met then the statements inside “else” gets executed.

Example

```
#include <iostream>
using namespace std;
int main(){


int time = 22;
if (time < 10) {
  cout << "Good morning.";
} else if (time < 20) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}

   return 0;
}
```

Output: Good evening.

In the example above, time (22) is greater than 10, so the **first condition** is false. The next condition, in the else if statement, is also false, so we move on to the else condition since **condition1** and **condition2** is both false - and print to the screen "Good evening".

However, if the time was 14, our program would print "Good day."

Example

```
#include <iostream>

using namespace std;

int main(){



int weight = 80;

if (weight < 50) {

  cout << "Underweight.";

} else if (weight>50 && weight<= 70) {
  cout << "Normal.";

} else {

  cout << "Overweight.";

}

   return 0;

}
```

Example of if-else-if

```
#include <iostream>
using namespace std;
int main(){
   int num;
   cout<<"Enter an integer number between 1 & 99999: ";
   cin>>num;
   if(num <100 && num>=1) {
      cout<<"Its a two digit number";
   }
   else if(num <1000 && num>=100) {
      cout<<"Its a three digit number";
   }
   else if(num <10000 && num>=1000) {
      cout<<"Its a four digit number";
   }
   else if(num <100000 && num>=10000) {
      cout<<"Its a five digit number";
   }
   else {
      cout<<"number is not between 1 & 99999";
   }
   return 0;
}
```

Output:

```
Enter an integer number between 1 & 99999: 8976
Its a four digit number
```

C++ Switch

C++ Switch Statements

Use the `switch` statement to select one of many code blocks to be executed.

Syntax

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

This is how it works:

- The `switch` expression is evaluated once
- The value of the expression is compared with the values of each `case`
- If there is a match, the associated block of code is executed
- The `break` and `default` keywords are optional, and will be described later in this chapter

The example below uses the weekday number to calculate the weekday name:

Example

```
int day = 4;
switch (day) {
  case 1:
    cout << "Monday";
    break;
  case 2:
    cout << "Tuesday";
    break;
  case 3:
    cout << "Wednesday";
    break;
  case 4:
    cout << "Thursday";
    break;
  case 5:
    cout << "Friday";
    break;
  case 6:
    cout << "Saturday";
    break;
  case 7:
    cout << "Sunday";
    break;
}
// Outputs "Thursday" (day 4)
```

The break Keyword

When C++ reaches a `break` keyword, it breaks out of the switch block.

This will stop the execution of more code and case testing inside the block.

When a match is found, and the job is done, it's time for a break. There is no need for more testing.

A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

The default Keyword

The `default` keyword specifies some code to run if there is no case match:

Example

```
int day = 4;
switch (day) {
  case 6:
    cout << "Today is Saturday";
    break;
  case 7:
    cout << "Today is Sunday";
    break;
  default:
    cout << "Looking forward to the Weekend";
}
// Outputs "Looking forward to the Weekend"
```

Example

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

int main () {
   char grade = 'D';

   switch(grade) {
      case 'A' :
         cout << "Excellent!" << endl;
         break;
      case 'B' :
      case 'C' :
         cout << "Well done" << endl;
         break;
      case 'D' :
         cout << "You passed" << endl;
         break;
      case 'F' :
         cout << "Better try again" << endl;
         break;
      default :
         cout << "Invalid grade" << endl;
   }
   cout << "Your grade is " << grade << endl;

   return 0;
}
```

```
Output
You passed
Your grade is D
```

Example

```
#include <iostream>

using namespace std;
int main( ) {

int mark;
cout << "Enter mark" << endl;
cin >> mark;

switch (mark >= 50){
      case 1:
            cout << "I have passed" << endl;
      break;
    default:
            cout << "I have failed" << endl;
}
      return 0;
}
```