Lecture 3: C++ Jump Statement (Continue & Goto)

Jump Statement:

Jump statement is used to terminating or continues the loop inside a program or to stop the execution of a function.

Jump statements in C++ mainly have four types:

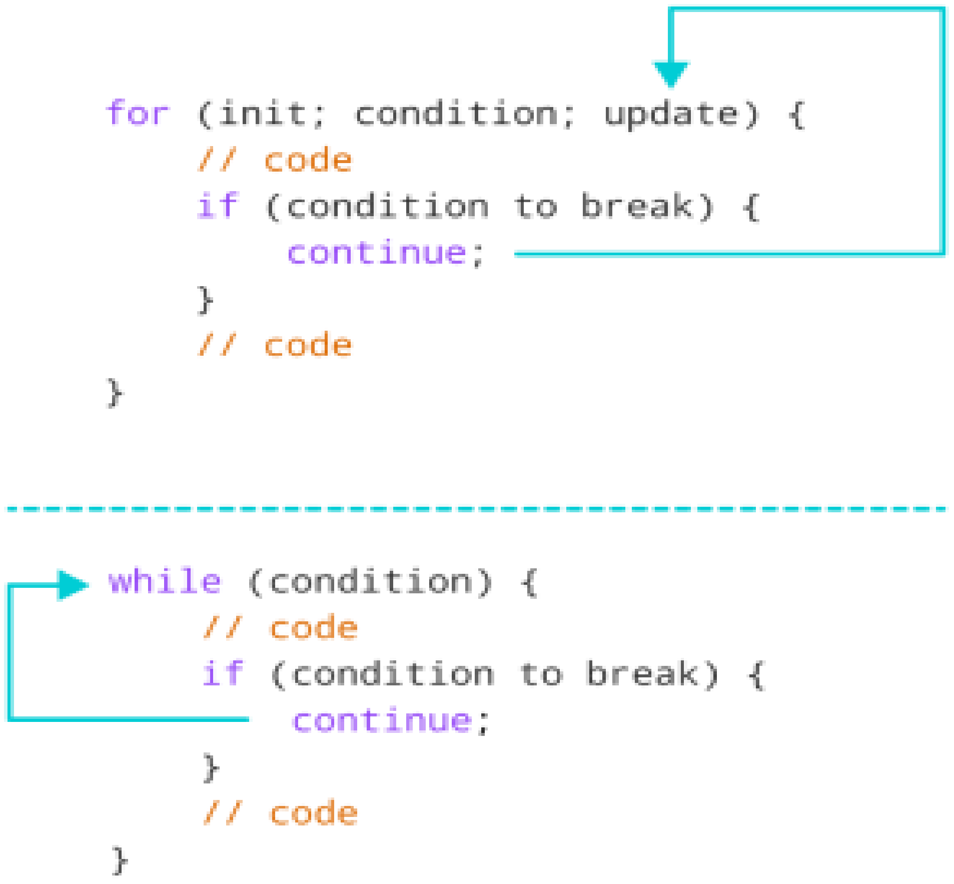
- Break statements.
- Continue statements.
- Goto statements
- Return statements.

Continue Statement:

In computer programming, the continue statement is used to skip the current iteration of the loop and the control of the program goes to the next iteration.

The syntax of the continue statement is:

```
continue;
```



```
for (init; condition; update) {
    // code
    if (condition to break) {
        continue;
    }
    // code
}
```

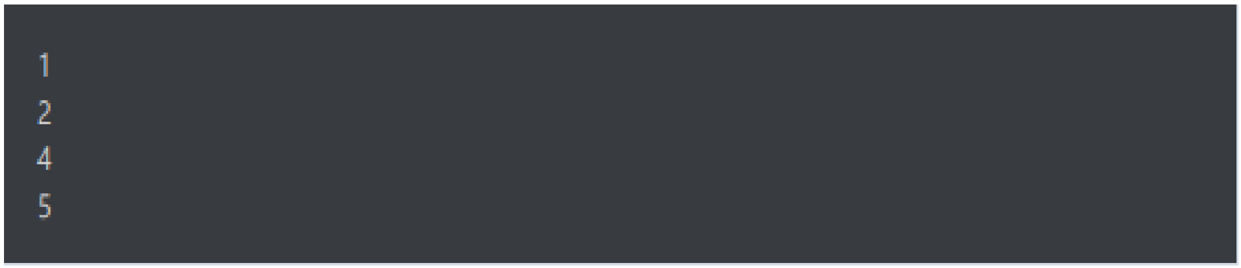
```
while (condition) {
    // code
    if (condition to break) {
        continue;
    }
    // code
}
```

Example 1: continue with for loop:

In a for loop, continue skips the current iteration and the control flow jumps to the update expression

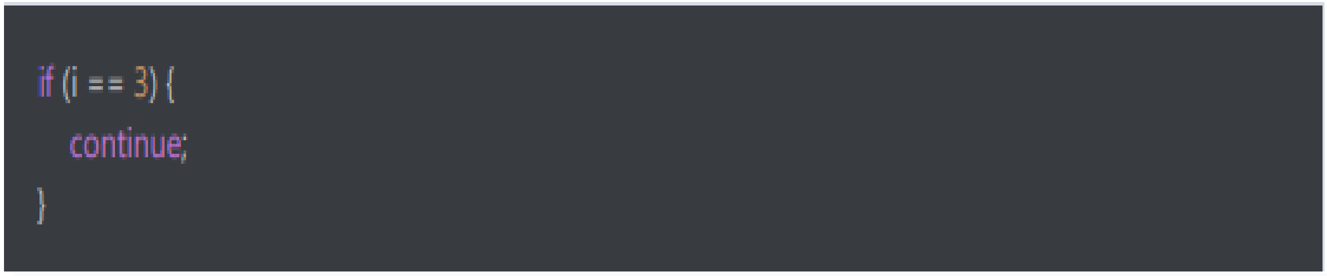
```
// program to print the value of i
#include <iostream>
using namespace std;
int main() {
 for (int i = 1; i <= 5; i++) {
 // condition to continue
 if (i == 3) {
 continue;
 }
 cout << i << endl;
 }
 return 0;
}
```

Output:



```
1
2
4
5
```

In the above program, we have used the the for loop to print the value of i in each iteration. Here, notice the code



```
if (i == 3) {
    continue;
}
```

This means

• When i is equal to 3, the continue statement skips the current iteration and starts the next iteration

• Then, i becomes 4, and the condition is evaluated again

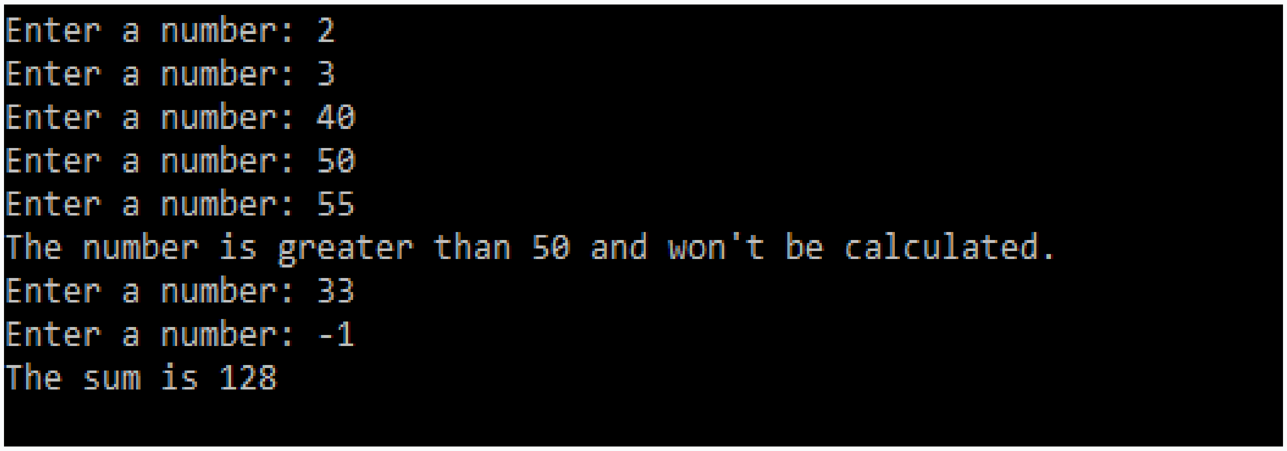
• Hence, 4 and 5 are printed in the next two iterations.

Example 2: continue with while loop:

In a while loop, continue skips the current iteration and control flow of the program jumps back to the while condition.

```
// program to calculate positive numbers till 50 only
// if the user enters a negative number,
// that number is skipped from the calculation
// negative number -> loop terminate
// numbers above 50 -> skip iteration
#include <iostream>
using namespace std;
int main() {
 int sum = 0;
 int number = 0;
 while (number >= 0) {
 // add all positive numbers
 sum += number;
 // take input from the user
 cout << "Enter a number: ";
 cin >> number;
 // continue condition
 if (number > 50) {
 cout << "The number is greater than 50 and won't be calculated." << endl;
 number = 0; // the value of number is made 0 again
 continue;
 }   }
 // display the sum
 cout << "The sum is " << sum << endl;
 return 0;
}
```

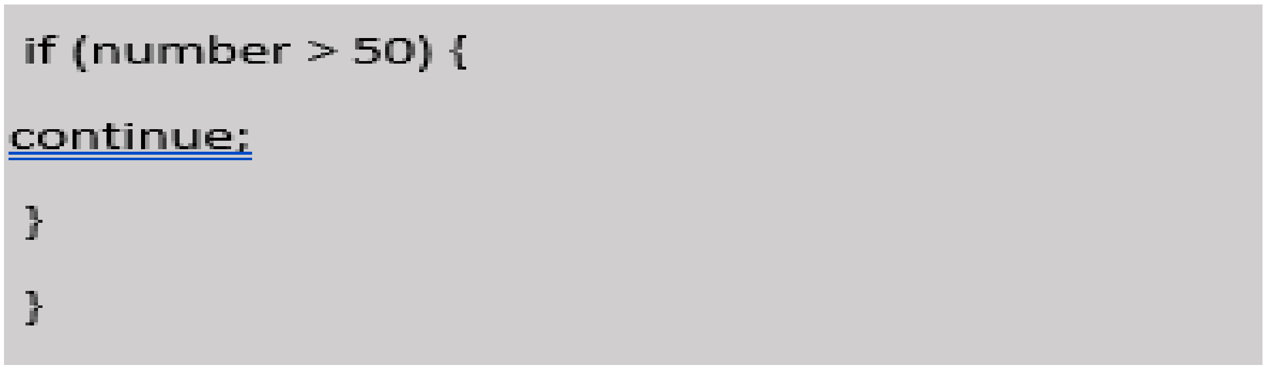
Output:



```
Enter a number: 2
Enter a number: 3
Enter a number: 40
Enter a number: 50
Enter a number: 55
The number is greater than 50 and won't be calculated.
Enter a number: 33
Enter a number: -1
The sum is 128
```

In the above program, the user enters a number. The while loop is used to print the total sum of positive numbers entered by the user, as long as the numbers entered are not greater than 50.

Notice the use of the continue statement.



```
 if (number > 50) {
continue;
 }
 }
```

• When the user enters a number greater than 50, the continue statement skips the current iteration. Then the control flow of the program goes to the condition of while loop.

• When the user enters a number less than 0, the loop terminates

Note: The continue statement works in the same way for the do...while loops.

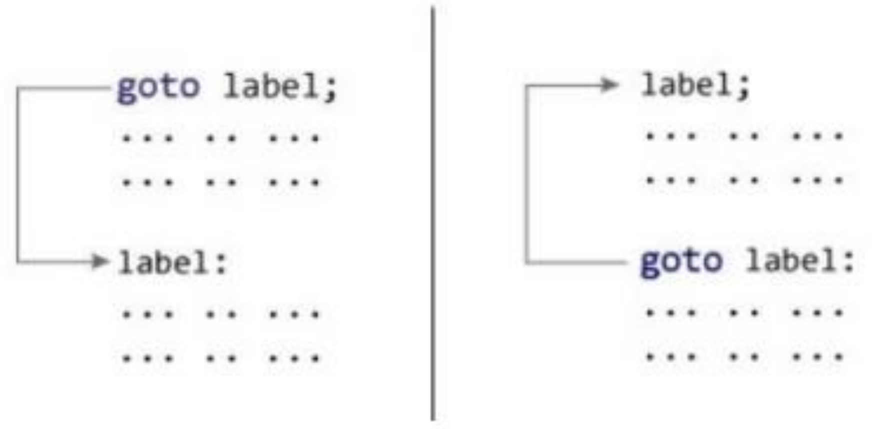
Goto Statement:

In C++ programming, goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

Syntax of goto Statement:

```
goto label;
... .. ...
... .. ...
... .. ...
label:
statement;
... .. ...
```

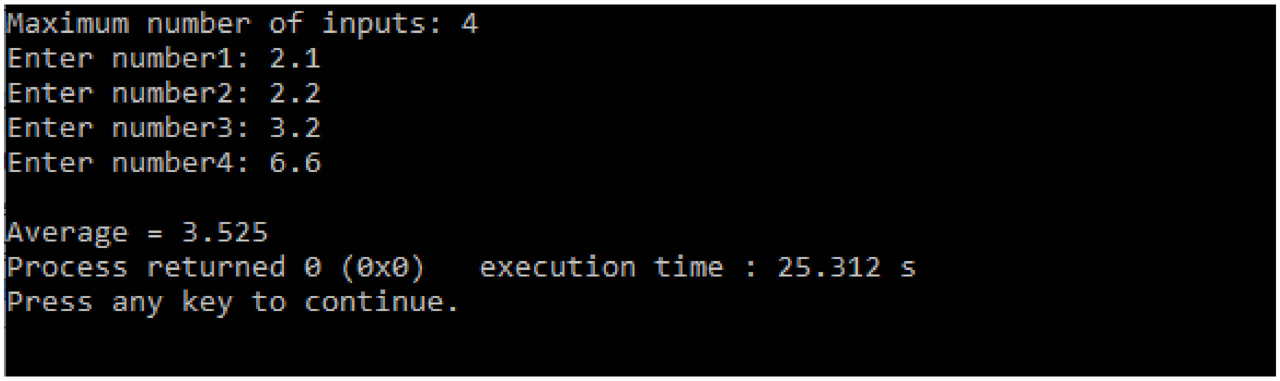
In the syntax above, label is an identifier. When goto label; is encountered, the control of program jumps to label: and executes the code below it.



Example: goto Statement:

```
// This program calculates the average of numbers entered by the user.
// If the user enters a negative number, it ignores the number and
// calculates the average number entered before it.
# include <iostream>
using namespace std;
int main() {
 float num, average, sum = 0.0;
 int i, n;
 cout << "Maximum number of inputs: ";
 cin >> n;
 for(i = 1; i <= n; ++i) {
 cout << "Enter number" << i << ": ";
 cin >> num;
 if(num < 0.0) {
 // Control of the program move to jump:
 goto jump;
 }
 sum += num;
 }
jump:
 average = sum / (i - 1);
 cout << "\nAverage = " << average;
 return 0;
}
```

Output:



```
Maximum number of inputs: 4
Enter number1: 2.1
Enter number2: 2.2
Enter number3: 3.2
Enter number4: 6.6

Average = 3.525
Process returned 0 (0x0)   execution time : 25.312 s
Press any key to continue.
```

Note: You can write any C++ program without the use of goto statement and is generally considered a good idea not to use it.

Reason to Avoid goto Statement

- The goto statement gives the power to jump to any part of a program but, makes the logic of the program complex and tangled.
- In modern programming, the goto statement is considered a harmful construct and a bad programming practice.
- The goto statement can be replaced in most of C++ program with the use of break and continue statements.