Break and Continue Statements in C++ | C++ Tutorials for Beginners #11

Overview Q&A Files Announcements

Break and Continue Statements in C++ | C++ Tutorials for Beginners #11

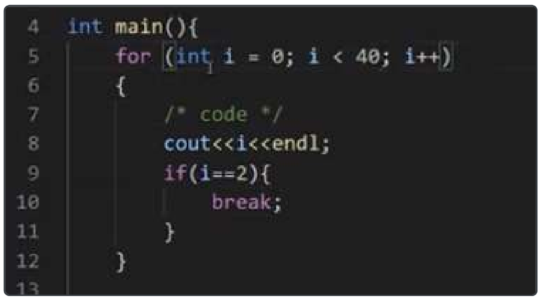
In this series of our C++ tutorials, we will visualize Break and continue statements in C++ language in this lecture. In our last lesson, we discussed for loop, while loop and do-while loop structures in C++.

In this C++ tutorial, the topics which we are going to cover today are given below:

- **Break Statements in C++**
- **Continue Statements in C++**

Break Statements

We had already discussed a little bit about break statements in switch statements. Today we will see the working of break statements in loops. Break statements in loops are used to terminate the loop. An example program for Break's statement is shown in figure 1.



```
int main(){
    for (int i = 0; i < 40; i++)
    {
        /* code */
        cout<<i<<endl;
        if(i==2){
            break;
        }
    }
```

Figure 1: Break Statement Program

As shown in figure 1, this is how the break statement program will be executed:

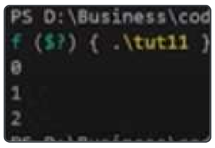
- Initialize integer variable "**i**" with value "**0**"
- Check the condition if the value of the variable "**i**" is smaller than "**40**"
- If the condition is true go into the loop body
- Execute "**cout**" function
- Check the condition if the value of the variable "**i**" is equal to "**2**", if it is equal terminate the loop and get out of loop body
- Update the value of "**i**" by one

Course Content

Hide Player

- Keep repeating these steps until the loop condition gets false, or the “if” condition inside the loop body gets true.

The output of the following program is shown in figure 2.

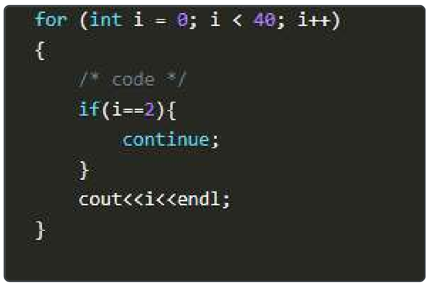


```
PS D:\Business\cod
f ($?) { .\tut11 }
0
1
2
```

Figure 2: Break Statement Program output

Continue Statements in C++

Continue statements are somewhat similar to break statements. The main difference is that the break statement entirely terminates the loop, but the continue statement only terminates the current iteration. An example program for continue statements is shown in figure 3.



```
for (int i = 0; i < 40; i++)
{
    /* code */
    if(i==2){
        continue;
    }
    cout<<i<<endl;
}
```

Figure 3: Continue Statement Program

As shown in figure 3, this is how the continue statement program will be executed:

- Initialize integer variable “**i**” with value “**0**”
- Check the condition if the value of the variable "**i**” is smaller than “**40**”
- If the condition is true go into the loop body
- Check the condition if the value of the variable "**i**” is equal to “**2**", if it is equal terminate the loop for the current iteration and go to the next iteration
- Execute “**cout**” function
- Update the value of “**i**” by one
- Keep repeating these steps until the loop condition gets false.

Code as described/written in the video

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

int main(){
    // for (int i = 0; i < 40; i++)
    // {
    //     /* code */
    //     if(i==2){
    //         break;
    //     }
    //     cout<<i<<endl;
    // }
    for (int i = 0; i < 40; i++)
    {
        /* code */
        if(i==2){
            continue;
        }
        cout<<i<<endl;
    }
```

← Previous   Next →