# Difference Between Break and Continue Statement

One of the primary reasons to pause a loop early is with the Break keyword. But the Continue statement can also interrupt a loop, though in a way that's quite different from Break. The difference between break and continue statements are as follows:

### Key Differences Between Break and Continue Statement

| Break | Continue |
|---|---|
| The break statement is used to exit from the loop. | The continue statement is not used to exit from the loop. |
| The break statement results in the control transfer out of the loop where the break keyword appears. | Continue statement results in skipping the loop's current iteration and continuing from the next iteration. |
| It results in the loop's early termination. | The continue statement causes the next iteration to start earlier. |
| The 'break' command terminates the loop. | 'continue' does not stop the loop from continuing, merely the current iteration. |
| **Syntax:** break; | **Syntax:** continue; |

## Break and Continue Statement

Various one-token statements are used in a program to break or halt the control flow. Two important one-token statements are:

- Break Statement
- Continue Statement

### What is a Break Statement?

The “break” keyword is used to terminate the execution of the current loop in which it is used. The break keyword is most widely used in control statements of C language, such as with for loop, while loop, do-while loop, and switch statements. Whenever the compiler encounters a break statement, the control is transferred to the statement that follows the loop in which the break statement is present.

The syntax of the break statement is as follows: break; (The keyword “break” followed by a semicolon.)

### Example of Break Statement

```
#include <stdio.h>
```

```
int main()

{

int a=0;

while(a<=10)

{

if(a==5)

break;

printf(“%d”, a);

a=a+1;

}

return 0;

}
```

Output: 0 1 2 3 4

As it can be seen from the above example that as soon as the value of a becomes equal to 5, the break statement is executed, and the control jumps out of the loop, bypassing its normal termination expression.

## What is a Continue Statement?

Like the break statement, the continue statement can only appear in the loop's body. Whenever the compiler encounters a continue statement, then the rest of the statements of the loop are skipped, and the control is transferred to the nearest loop continuation portion of the enclosing loop. The keyword “continue” is also frequently used in control statements like for loop, while loop, and do-while loop. The syntax of the continue statement is as follows: continue; (The keyword “continue” followed by a semicolon.)

## Example of Continue Statement

```
#include <stdio.h>

int main()
```

```
{
int a;
for(a=0; a<=10; a++)
{
if(a==5)
continue;
printf("%d", a);
}
return 0;
}
```

Output: 0 1 2 3 4 6 7 8 9 10

As soon as a becomes equal to 5, the continue statement is encountered, so the printf statement is skipped, and the compiler executes the next iteration.