**C if Statement**

The syntax of the if statement in C programming is:

The if statement evaluates the test expression inside the parenthesis ().

* If the test expression is evaluated to true, statements inside the body of if are executed.
* If the test expression is evaluated to false, statements inside the body of if are not executed.
* *example*

#include <stdio.h>

int main() {

int number1, number2;

printf("Enter two integers: ");

scanf("%d %d", &number1, &number2);

if (number1 >= number2) {

if (number1 == number2) {

printf("Result: %d = %d",number1,number2);

}

else {

printf("Result: %d > %d", number1, number2);

}

}

else {

printf("Result: %d < %d",number1, number2);

}

return 0;

}

# C for Loop

In programming, a loop is used to repeat a block of code until the specified condition is met.

C programming has three types of loops:

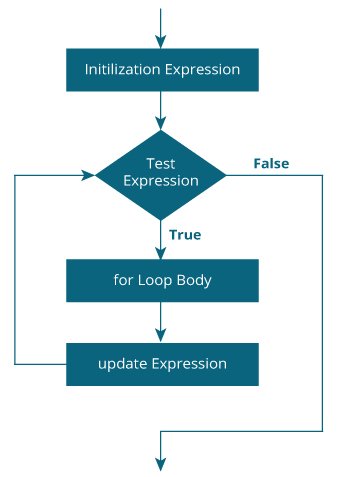
1. for loop
2. while loop
3. do...while loop

### How for loop works?

* The initialization statement is executed only once.
* Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
* However, if the test expression is evaluated to true, statements inside the body of the for loop are executed, and the update expression is updated.
* Again the test expression is evaluated.

This process goes on until the test expression is false. When the test expression is false, the loop terminates.

To learn more about test expression (when the test expression is evaluated to true and false), check out [relational](https://www.programiz.com/c-programming/c-operators#relational) and [logical operators](https://www.programiz.com/c-programming/c-operators#logical).



### Example 1: for loop

// Print numbers from 1 to 10

#include <stdio.h>

int main() {

int i;

for (i = 1; i < 11; ++i)

{

printf("%d ", i);

}

return 0;

}

[Run Code](https://www.programiz.com/c-programming/online-compiler)

**Output**

1 2 3 4 5 6 7 8 9 10

1. i is initialized to 1.
2. The test expression i < 11 is evaluated. Since 1 less than 11 is true, the body of for loop is executed. This will print the **1** (value of i) on the screen.
3. The update statement ++i is executed. Now, the value of i will be 2. Again, the test expression is evaluated to true, and the body of for loop is executed. This will print **2** (value of i) on the screen.
4. Again, the update statement ++i is executed and the test expression i < 11 is evaluated. This process goes on until i becomes 11.
5. When i becomes 11, i < 11 will be false, and the for loop terminates.

## while loop

The syntax of the while loop is:

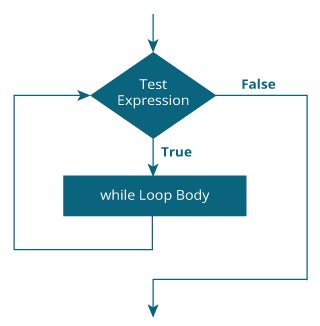
while (testExpression) {

// the body of the loop

}

### How while loop works?

* The while loop evaluates the testExpression inside the parentheses ().
* If testExpression is **true**, statements inside the body of while loop are executed. Then, testExpression is evaluated again.
* The process goes on until testExpression is evaluated to **false**.
* If testExpression is **false**, the loop terminates (ends).



### Example 1: while loop

// Print numbers from 1 to 5

#include <stdio.h>

int main() {

int i = 1;

while (i <= 5) {

printf("%d\n", i);

++i;

}

return 0;

}

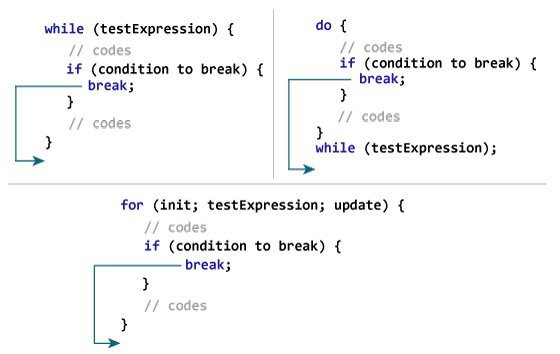
# *C break and continue*

## C break

The break statement ends the loop immediately when it is encountered. Its syntax is:

break;

The break statement is almost always used with if...else statement inside the loop.

**Working of break in C**

### Example 1: break statement

// Program to calculate the sum of numbers (10 numbers max)

// If the user enters a negative number, the loop terminates

#include <stdio.h>

int main() {

int i;

double number, sum = 0.0;

for (i = 1; i <= 10; ++i) {

printf("Enter n%d: ", i);

scanf("%lf", &number);

// if the user enters a negative number, break the loop

if (number < 0.0) {

break;

}

sum += number; // sum = sum + number;

}

printf("Sum = %.2lf", sum);

return 0;

}

[Run Code](https://www.programiz.com/c-programming/online-compiler)

**Output**

Enter n1: 2.4

Enter n2: 4.5

Enter n3: 3.4

Enter n4: -3

Sum = 10.30

**C continue**

The continue statement skips the current iteration of the loop and continues with the next iteration. Its syntax is:

continue;

The continue statement is almost always used with the if...else statement.

// Program to calculate the sum of numbers (10 numbers max)

// If the user enters a negative number, it's not added to the result

#include <stdio.h>

int main() {

int i;

double number, sum = 0.0;

for (i = 1; i <= 10; ++i) {

printf("Enter a n%d: ", i);

scanf("%lf", &number);

if (number < 0.0) {

continue;

}

sum += number; // sum = sum + number;

}

printf("Sum = %.2lf", sum);

return 0;

}

[Run Code](https://www.programiz.com/c-programming/online-compiler)

**Output**

Enter n1: 1.1

Enter n2: 2.2

Enter n3: 5.5

Enter n4: 4.4

Enter n5: -3.4

Enter n6: -45.5

Enter n7: 34.5

Enter n8: -4.2

Enter n9: -1000

Enter n10: 12

Sum = 59.70

In this program, when the user enters a positive number, the sum is calculated using sum += number; statement.

When the user enters a negative number, the continue statement is executed and it skips the negative number from the calculation.

***Switch*  
How does the switch statement work?**

The expression is evaluated once and compared with the values of each case label.

* If there is a match, the corresponding statements after the matching label are executed. For example, if the value of the expression is equal to constant2, statements after case constant2: are executed until break is encountered.
* If there is no match, the default statements are executed.

**Notes:**

* If we do not use the break statement, all statements after the matching label are also executed.
* The default clause inside the switch statement is optional.
* // Program to create a simple calculator
* #include <stdio.h>
* int main() {
* char operation;
* double n1, n2;
* printf("Enter an operator (+, -, \*, /): ");
* scanf("%c", &operation);
* printf("Enter two operands: ");
* scanf("%lf %lf",&n1, &n2);
* switch(operation)
* {
* case '+':
* printf("%.1lf + %.1lf = %.1lf",n1, n2, n1+n2);
* break;
* case '-':
* printf("%.1lf - %.1lf = %.1lf",n1, n2, n1-n2);
* break;
* case '\*':
* printf("%.1lf \* %.1lf = %.1lf",n1, n2, n1\*n2);
* break;
* case '/':
* printf("%.1lf / %.1lf = %.1lf",n1, n2, n1/n2);
* break;
* // operator doesn't match any case constant +, -, \*, /
* default:
* printf("Error! operator is not correct");
* }
* return 0;
* }
* [Run Code](https://www.programiz.com/c-programming/online-compiler)
* **Output**
* Enter an operator (+, -, \*, /): -
* Enter two operands: 32.5
* 12.4
* 32.5 - 12.4 = 20.1