**C for Loop**

**In this tutorial, you will learn to create for loop in C programming with the help of examples.**

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

C programming has three types of loops:

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

We will learn about for loop in this tutorial. In the next tutorial, we will learn about while and do...while loop.

**for Loop**

The syntax of the for loop is:

1. for (initializationStatement; testExpression; updateStatement)
2. {
3. // statements inside the body of loop
4. }

**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 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).

**for loop Flowchart**



**Example 1: for loop**

1. # Print numbers from 1 to 10
2. #include <stdio.h>
3. int main() {
4. int i;
5. for (i = 1; i < 11; ++i)
6. {
7. printf("%d ", i);
8. }
9. return 0;
10. }

**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.

**Example 2: for loop**

1. // Program to calculate the sum of first n natural numbers
2. // Positive integers 1,2,3...n are known as natural numbers
3. #include <stdio.h>
4. int main()
5. {
6. int num, count, sum = 0;
7. printf("Enter a positive integer: ");
8. scanf("%d", &num);
9. // for loop terminates when num is less than count
10. for(count = 1; count <= num; ++count)
11. {
12. sum += count;
13. }
14. printf("Sum = %d", sum);
15. return 0;
16. }

**Output**

Enter a positive integer: 10

Sum = 55

The value entered by the user is stored in the variable num. Suppose, the user entered 10.

The count is initialized to 1 and the test expression is evaluated. Since the test expression count<=num (1 less than or equal to 10) is true, the body of for loop is executed and the value of sum will equal to 1.

Then, the update statement ++count is executed and the count will equal to 2. Again, the test expression is evaluated. Since 2 is also less than 10, the test expression is evaluated to true and the body of for loop is executed. Now, the sum will equal 3.

This process goes on and the sum is calculated until the count reaches 11.

When the count is 11, the test expression is evaluated to 0 (false), and the loop terminates.

Then, the value of sum is printed on the screen.

We will learn about while loop and do...while loop in the next tutorial.

**while loop**

The syntax of the while loop is:

1. while (testExpression)
2. {
3. // statements inside the body of the loop
4. }

**How while loop works?**

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

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).

**Flowchart of while loop**



**Example 1: while loop**

1. // Print numbers from 1 to 5
2. #include <stdio.h>
3. int main()
4. {
5. int i = 1;
6.
7. while (i <= 5)
8. {
9. printf("%d\n", i);
10. ++i;
11. }
12. return 0;
13. }

**Output**

1

2

3

4

5

Here, we have initialized i to 1.

1. When i is 1, the test expression i <= 5 is true. Hence, the body of the while loop is executed. This prints 1 on the screen and the value of i is increased to 2.
2. Now, i is 2, the test expression i <= 5 is again true. The body of the while loop is executed again. This prints 2 on the screen and the value of i is increased to 3.
3. This process goes on until i becomes 6. When i is 6, the test expression i <= 5 will be false and the loop terminates.

**do...while loop**

The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.

The syntax of the do...while loop is:

1. do
2. {
3. // statements inside the body of the loop
4. }
5. while (testExpression);

**How do...while loop works?**

* The body of do...while loop is executed once. Only then, the test expression is evaluated.
* If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
* This process goes on until the test expression becomes false.
* If the test expression is false, the loop ends.

**Flowchart of do...while Loop**



**Example 2: do...while loop**

1. // Program to add numbers until the user enters zero
2. #include <stdio.h>
3. int main()
4. {
5. double number, sum = 0;
6. // the body of the loop is executed at least once
7. do
8. {
9. printf("Enter a number: ");
10. scanf("%lf", &number);
11. sum += number;
12. }
13. while(number != 0.0);
14. printf("Sum = %.2lf",sum);
15. return 0;
16. }

**Output**

Enter a number: 1.5

Enter a number: 2.4

Enter a number: -3.4

Enter a number: 4.2

Enter a number: 0

Sum = 4.70

C break and continue

**We learned about loops in previous tutorials. In this tutorial, we will learn to use break and continue statements with the help of examples.**

**C break**

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

1. break;

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

**How break statement works?**



**Example 1: break statement**

1. // Program to calculate the sum of a maximum of 10 numbers
2. // If a negative number is entered, the loop terminates
3. # include <stdio.h>
4. int main()
5. {
6. int i;
7. double number, sum = 0.0;
8. for(i=1; i <= 10; ++i)
9. {
10. printf("Enter a n%d: ",i);
11. scanf("%lf",&number);
12. // If the user enters a negative number, the loop ends
13. if(number < 0.0)
14. {
15. break;
16. }
17. sum += number; // sum = sum + number;
18. }
19. printf("Sum = %.2lf",sum);
20.
21. return 0;
22. }

**Output**

Enter a n1: 2.4

Enter a n2: 4.5

Enter a n3: 3.4

Enter a n4: -3

Sum = 10.30

This program calculates the sum of a maximum of 10 numbers. Why a maximum of 10 numbers? It's because if the user enters a negative number, the break statement is executed. This will end the for loop, and the sum is displayed.

In C, break is also used with the switch statement. This will be discussed in the next tutorial.

**C continue**

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

1. continue;

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

**How continue statement works?**



**Example 2: continue statement**

1. // Program to calculate the sum of a maximum of 10 numbers
2. // Negative numbers are skipped from the calculation
3. # include <stdio.h>
4. int main()
5. {
6. int i;
7. double number, sum = 0.0;
8. for(i=1; i <= 10; ++i)
9. {
10. printf("Enter a n%d: ",i);
11. scanf("%lf",&number);
12. if(number < 0.0)
13. {
14. continue;
15. }
16. sum += number; // sum = sum + number;
17. }
18. printf("Sum = %.2lf",sum);
19.
20. return 0;
21. }

**Output**

Enter a n1: 1.1

Enter a n2: 2.2

Enter a n3: 5.5

Enter a n4: 4.4

Enter a n5: -3.4

Enter a n6: -45.5

Enter a n7: 34.5

Enter a n8: -4.2

Enter a n9: -1000

Enter a 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.