# #C if...else Statement

#### In this tutorial, you will learn about if statement (including if...else and nested if..else) in C programming with the help of examples.

## C if Statement

The syntax of the if statement in C programming is:

if (test expression)

{

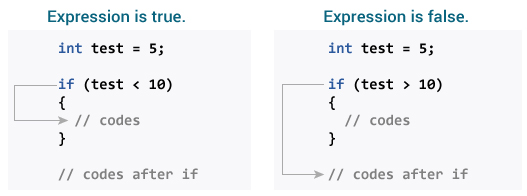
// statements to be executed if the test expression is true

}

### **How if statement works?**

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.



To learn more about when test expression is evaluated to true (non-zero value) and false (0), check [relational](https://www.programiz.com/c-programming/c-operators" \l "relational" \o "C Relational operators) and [logical operators](https://www.programiz.com/c-programming/c-operators" \l "logical" \o "C logical operators).

### **Example 1: if statement**

// Program to display a number if it is negative

#include <stdio.h>int main() {

int number;

printf("Enter an integer: ");

scanf("%d", &number);

// true if number is less than 0

if (number < 0) {

printf("You entered %d.\n", number);

}

printf("The if statement is easy.");

return 0;

}

****Output 1****

Enter an integer: -2

You entered -2.

The if statement is easy.

When the user enters -2, the test expression number<0 is evaluated to true. Hence, You entered -2 is displayed on the screen.

****Output 2****

Enter an integer: 5

The if statement is easy.

When the user enters 5, the test expression number<0 is evaluated to false and the statement inside the body of if is not executed

## C if...else Statement

The if statement may have an optional else block. The syntax of the if..else statement is:

if (test expression) {

// statements to be executed if the test expression is true

}else {

// statements to be executed if the test expression is false

}

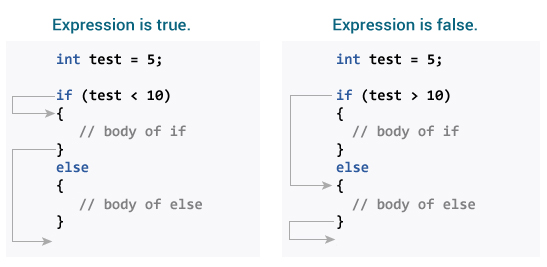
### **How if...else statement works?**

If the test expression is evaluated to true,

* statements inside the body of if are executed.
* statements inside the body of else are skipped from execution.

If the test expression is evaluated to false,

* statements inside the body of else are executed
* statements inside the body of if are skipped from execution.



### **Example 2: if...else statement**

// Check whether an integer is odd or even

#include <stdio.h>int main() {

int number;

printf("Enter an integer: ");

scanf("%d", &number);

// True if the remainder is 0

if (number%2 == 0) {

printf("%d is an even integer.",number);

}

else {

printf("%d is an odd integer.",number);

}

return 0;

}

****Output****

Enter an integer: 7

7 is an odd integer.

When the user enters 7, the test expression number%2==0 is evaluated to false. Hence, the statement inside the body of else is executed.

## C if...else Ladder

The if...else statement executes two different codes depending upon whether the test expression is true or false. Sometimes, a choice has to be made from more than 2 possibilities.

The if...else ladder allows you to check between multiple test expressions and execute different statements.

### **Syntax of if...else Ladder**

if (test expression1) {

// statement(s)

}else if(test expression2) {

// statement(s)

}else if (test expression3) {

// statement(s)

}

.

.else {

// statement(s)

}

### **Example 3: C if...else Ladder**

// Program to relate two integers using =, > or < symbol

#include <stdio.h>int main() {

int number1, number2;

printf("Enter two integers: ");

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

//checks if the two integers are equal.

if(number1 == number2) {

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

}

//checks if number1 is greater than number2.

else if (number1 > number2) {

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

}

//checks if both test expressions are false

else {

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

}

return 0;

}

****Output****

Enter two integers: 12

23

Result: 12 < 23

## Nested if...else

It is possible to include an if...else statement inside the body of another if...else statement.

### **Example 4: Nested if...else**

This program given below relates two integers using either <, > and = similar to the if...else ladder's example. However, we will use a nested if...else statement to solve this problem.

#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;

}

If the body of an if...else statement has only one statement, you do not need to use brackets {}.

For example, this code

if (a > b) {

print("Hello");

}

print("Hi");

is equivalent to

if (a > b)

print("Hello");

print("Hi");

# #C for Loop

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

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:

* for loop
* while loop
* 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:

for (initializationStatement; testExpression; updateStatement)

{

// statements inside the body of loop

}

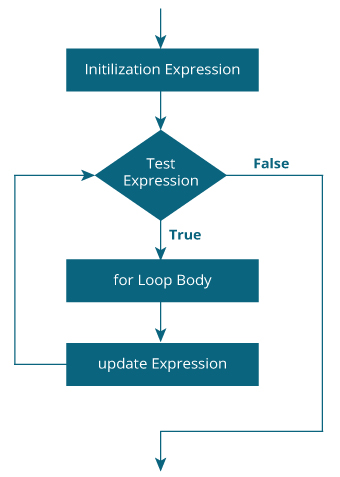
### **How for loop works?**

* 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 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" \l "relational" \o "C Relational operators) and [logical operators](https://www.programiz.com/c-programming/c-operators" \l "logical" \o "C logical operators).

### **for loop Flowchart**



### **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;

}

****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**

// Program to calculate the sum of first n natural numbers// Positive integers 1,2,3...n are known as natural numbers

#include <stdio.h>int main(){

int num, count, sum = 0;

printf("Enter a positive integer: ");

scanf("%d", &num);

// for loop terminates when num is less than count

for(count = 1; count <= num; ++count)

{

sum += count;

}

printf("Sum = %d", sum);

return 0;

}

****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.

# #C while and do...while Loop

#### In this tutorial, you will learn to create while and do...while 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

In the previous tutorial, we learned about for loop. In this tutorial, we will learn about while and do..while loop.

## while loop

The syntax of the while loop is:

while (testExpression)

{

// statements inside the body of the loop

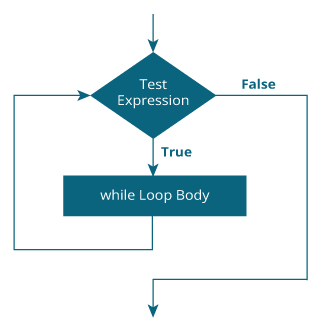
}

### **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" \l "relational" \o "C Relational operators) and [logical operators](https://www.programiz.com/c-programming/c-operators" \l "logical" \o "C logical operators).

### **Flowchart of while loop**



### **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;

}

****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:

do

{

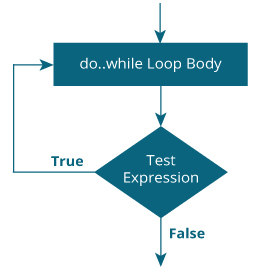
// statements inside the body of the loop

}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**

// Program to add numbers until the user enters zero

#include <stdio.h>int main(){

double number, sum = 0;

// the body of the loop is executed at least once

do

{

printf("Enter a number: ");

scanf("%lf", &number);

sum += number;

}

while(number != 0.0);

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

return 0;

}

****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