Presented By :Mostafa Saqly

C Flow Control

Content

- C if...else Statement
- C switch Statement
- C for Loop
- C while and do...while Loop
- C break and continue

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

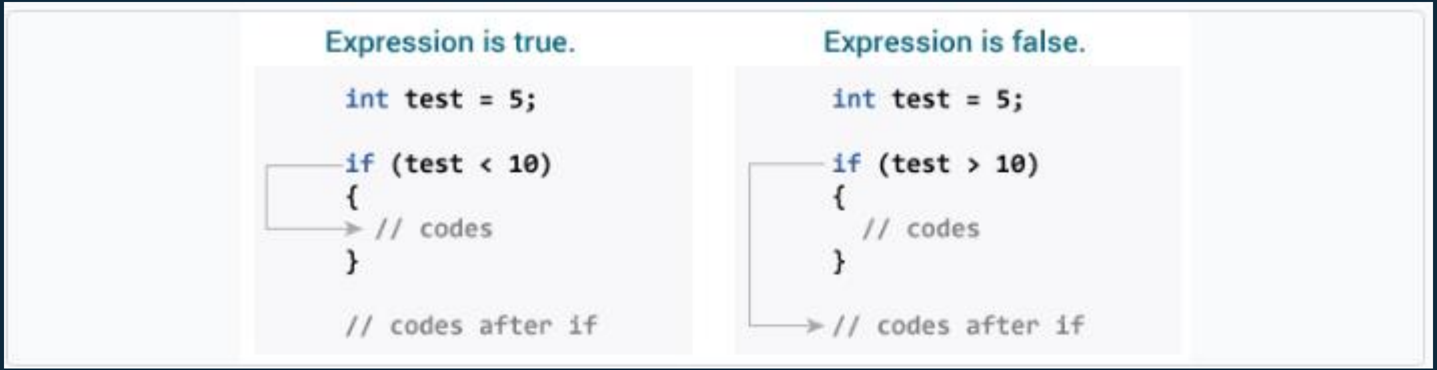
}

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.

How if statement works?



```
int test = 5;

if (test < 10)
{
   // codes
}

// codes after if
```

```
int test = 5;

if (test > 10)
{
   // codes
}

// codes after if
```

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?

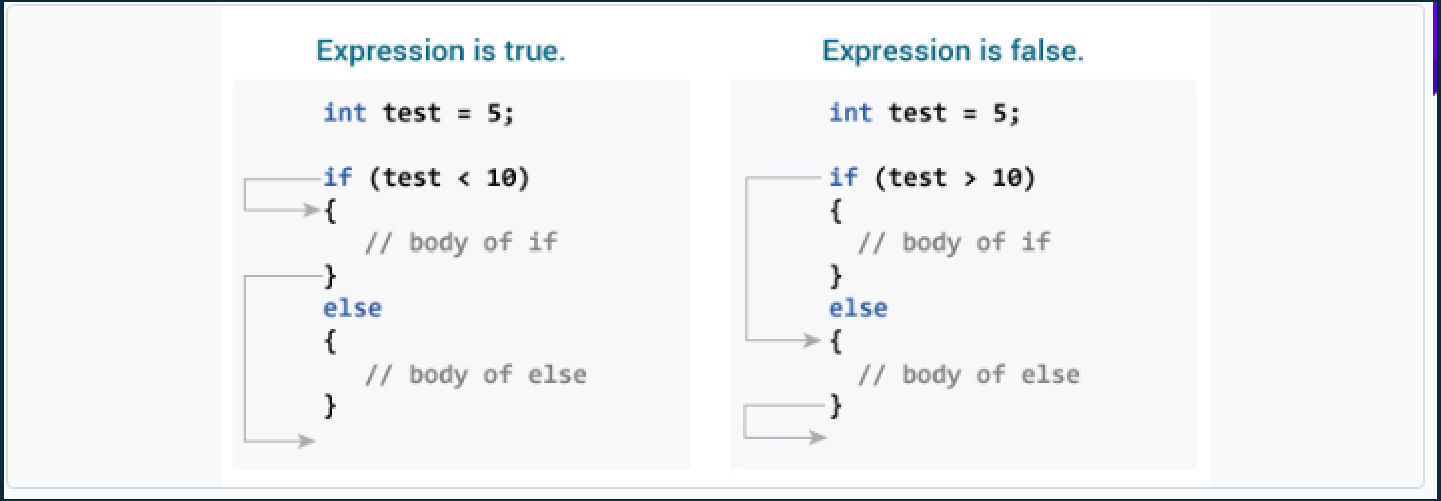
If the test expression is evaluated to true,

- 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.

How if...else statement works?



Expression is true.

```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

Expression is false.

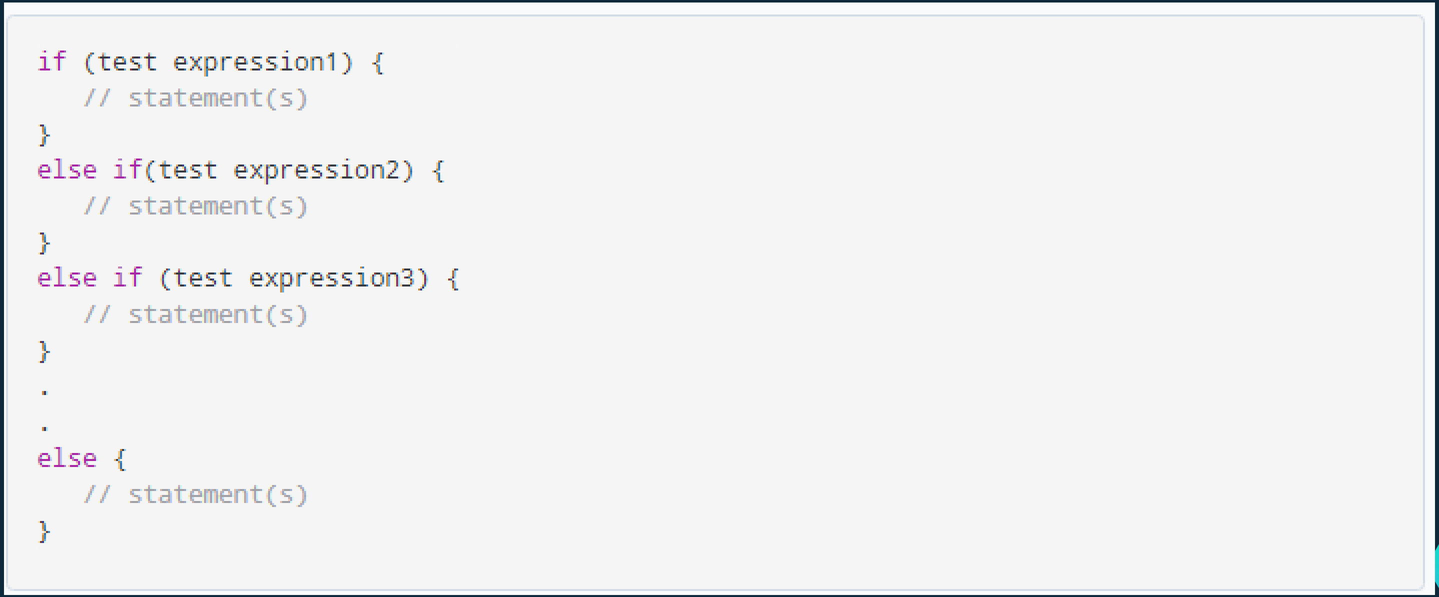
```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

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.

C if...else Ladder



```
if (test expression1) {
   // statement(s)
}
else if(test expression2) {
   // statement(s)
}
else if (test expression3) {
   // statement(s)
}
.
.
else {
   // statement(s)
}
```

Nested if...else

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

C switch Statement

- The switch statement allows us to execute one code block among many alternatives.
- You can do the same thing with the if...else..if ladder. However, the syntax of the switch statement is much easier to read and write.

Syntax of switch...case

- switch (expression) {
- case constant1: // statements
- break;
- case constant2: // statements
- break; . . .
- default: // default statements }

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.
- If we do not use break, all statements after the matching label are executed.
- By the way, the default clause inside the switch statement is optional.

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:

- ◇ for loop
- ◇ while loop
- ◇ do...while loop

C 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.

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).

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);
```

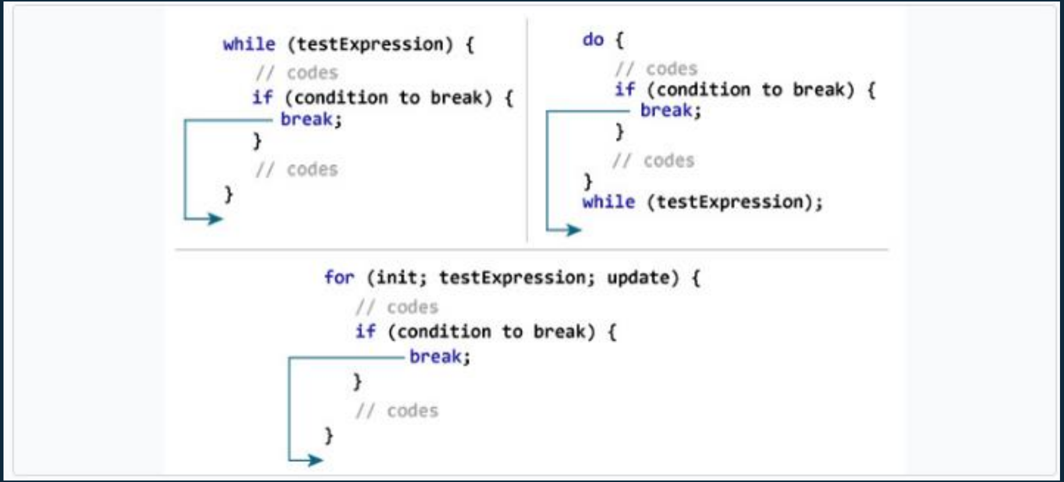
How do...while loop works?

- 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.

C break and continue

C break

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



```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

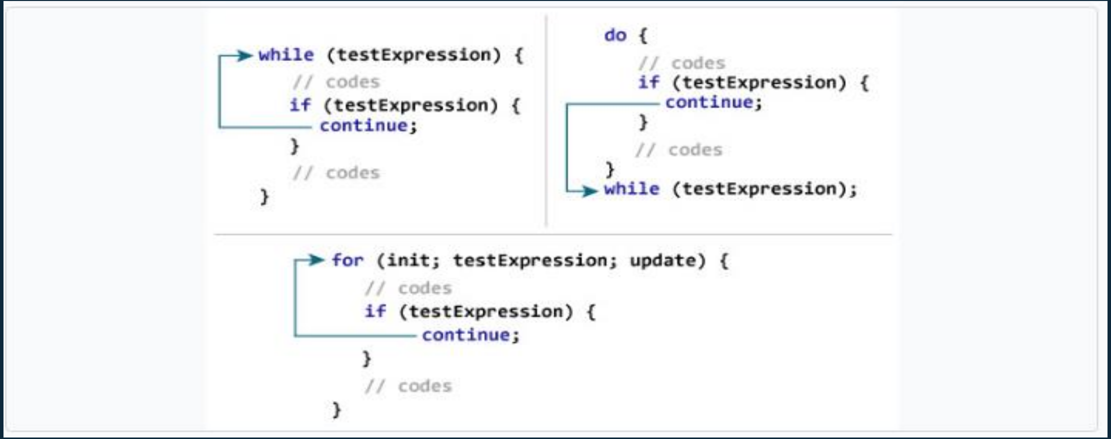
```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

C break and continue

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.



```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

Thanks!

Any questions?