



# Coding-bootcamps.com

## Introduction to C Programming Language



# Coding Bootcamps

By Tom Brownlee  
from [Coding Bootcamps](https://coding-bootcamps.com)

# Session 10

## Flow Control Constructs

# SESSION 9 RECAP

## Operators and Expressions

- Arithmetic, Logical, and Bit Operators
- Precedence and Associativity
- Assignment and Casting
- The Conditional Operator

# SESSION 10 OUTLINE

## Flow Control Constructs

- Conditional Constructs: if, switch
- Looping Constructs: while, do, for
- Programming Style

# Flow Control Constructs

## Conditional Constructs: if, switch

An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

### Syntax

The syntax of an **if...else** statement in C programming language is –

```
if(boolean_expression) {  
    /* statement(s) will execute if the boolean expression is true */  
}  
else {  
    /* statement(s) will execute if the boolean expression is false */  
}
```

If the Boolean expression evaluates to **true**, then the **if block** will be executed, otherwise, the **else block** will be executed.

C programming language assumes any **non-zero** and **non-null** values as **true**, and if it is either **zero** or **null**, then it is assumed as **false** value.

# Flow Control Constructs

## Conditional Constructs: if, switch

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each **switch case**.

## Syntax

The syntax for a **switch** statement in C programming language is as follows –

```
switch(expression) {  
  
    case constant-expression  :  
        statement(s);  
        break; /* optional */  
  
    case constant-expression  :  
        statement(s);  
        break; /* optional */  
  
    /* you can have any number of case statements */  
    default : /* Optional */  
        statement(s);  
}
```

# Flow Control Constructs

## Looping Constructs: while, do, for

A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

## Syntax

The syntax of a **while** loop in C programming language is –

```
while(condition) {  
    statement(s);  
}
```

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any nonzero value. The loop iterates while the condition is true.

When the condition becomes false, the program control passes to the line immediately following the loop.

# Flow Control Constructs

## Looping Constructs: while, do, for

Unlike **for** and **while** loops, which test the loop condition at the top of the loop, the **do...while** loop in C programming checks its condition at the bottom of the loop.

A **do...while** loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

## Syntax

The syntax of a **do...while** loop in C programming language is –

```
do {  
    statement(s);  
} while( condition );
```

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.



# Flow Control Constructs

## Looping Constructs: while, do, for

### Syntax

The syntax of a **for** loop in C programming language is –

```
for ( init; condition; increment ) {  
    statement(s);  
}
```

Here is the flow of control in a 'for' loop –

- The **init** step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the **condition** is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and the flow of control jumps to the next statement just after the 'for' loop.
- After the body of the 'for' loop executes, the flow of control jumps back up to the **increment** statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.
- The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the 'for' loop terminates.

# Flow Control Constructs

## Programming Style

This is a basic introduction to a good code style in the C Programming Language. It is designed to provide information on how to effectively use indentation, comments, and other elements that will make your C code more readable. It is not a tutorial on actually programming in C.

As a beginning programmer, the point of creating structure in the programs' code might not be clear, as the compiler doesn't care about the difference. However, as programs become complex, chances are that writing the program has become a joint effort. (Or others might want to see how it was accomplished.)

Therefore, the code is no longer designed purely for a compiler to read.

Keep in mind there are many different styles for writing C code. Different teams, companies, and people may have different style preferences. Tools exist within text editors and IDEs to style code automatically for you. My personal advice: Find a commonly-used style you like, stick to it, and if necessary use these tools to change to your organization's preferred style.

# Flow Control Constructs

## Programming Style

Brace style: Do opening curly braces { go on their own line, or do they share a line with the structure that requires them?

Naming conventions: CamelCase, snake\_case, sHungarianNotation

Line length: 80 characters? 100 characters? More?

Indentation: 2 spaces, 4 spaces, tabs?

Blank lines: Where do they look best?

# Summary

# Live private coaching sessions for C

- Private tutoring sessions for software design and engineering- Weekly and monthly plans
- C and C++ programming languages- Private tutoring sessions

# More software engineering training

- [Introduction to Python Programming](#)
- [Introduction to Java Programming](#)
- [Introduction to Go Programming](#)
- [Learn Kotlin Programming by Examples](#)

# Follow up classes

- [Intermediate level C programming language with hands-on examples](#)
- [Learn C++ Programming by Examples](#)



Coding-bootcamps.com

Thank You



**Coding**  
Bootcamps