# Getting started with C

## Comments

A comment is preceded with //

Comments are just used for a human readable description, and has no effect on the program execution.

***// This is a comment***

Another way of writing a comment is to put /\* before and \*/ after the text. Such a comment can then span several lines (compared to // which is only for one line comments).

## Statement

Simply put, it is a line of code that performs something, e.g. add two numbers.

All statements end with a semicolon **;**

**myVar = y + z;** *// Adds y and z and puts the result into myVar*

## Declaring variables

**int myVar;**  // Declares a variable with the name *myVar* and of the type *int* (such a variable can

// contain positive or negative integer values).

**unsigned int mySecondVar;**  // Declares a variable of type *unsigned int* (such a variable can

// only contain positive integer values)

## Using the variables

**myVar = 10;** // Assign the value 10 to the variable *myVar*

**mySecondVar = myVar + 5;**  // mySecondVar will after this contain the value 15

**int x = myVar \* 100;** // Declare the variable *x* and assign a value to it on the same line.

## Constants

**const int myC = 100;**  // Declares a constant named *MyC* of type *int*. The value of a constant

// cannot be changed after it has been declared, otherwise you can use it in

// the same way as a variable.

## Calling functions

The line below is calling the function *print\_long*, with the parameter *mySecondVar*:

**print\_long(mySecondVariable);** // This would print the value 15 on the robot display (due to the

// code above, where mySecondVar is assigned the value 10 + 5).

The function needs to be defined somewhere else, and contain some code to execute when it is called.

## The if statement (and printing, in our robots)

**if(mySecondVariable > 10)** // Check if mySecondVariable is larger than 10

**{**  // if it is larger, execute the code within the first pair of { }

**print(“mySecondVariable is larger than 10”)**

**}**

**else** // otherwise, execute the code below (i.e. inside the next pair of { } )

**{**

// The else statement is optional

**print(“mySecondVariable is less than or equal to 10”)**

**}**

## Boolean expressions and logical operators

mySecondVariable > 10 as shown above is an example of a Boolean expression.

The result evaluates to **true** if it is fulfilled, and **false** otherwise

The result can for example be used in if-statements (as shown above) or while-loops (see further below).

This table shows other Boolean expressions that can be used:

|  |  |  |
| --- | --- | --- |
| **Expression** | **Meaning** | **Example** |
| a < b | a is smaller than b | 5 < 10 is **true** |
| a == b | a is equal to b | 4 == 5 is **false** |
| a != b | a is not equal to b | 4 != 5 is **true** |
| a <= b | a is smaller than or equal to b | 2 <= 2 is **true** |
| a >= b | a is larger than or equal to b | 2 >= 3 is **false** |

You can also combine Boolean expressions using logical operators. For example:

(a < b) && (x == 10)

means that “a is less than b” **and** “x is equal to 10”.

|  |  |
| --- | --- |
| **Logical operator** | **Meaning** |
| && | Logical AND |
| || | Logical OR |
| ! | Example: **!a**  “not a“, which means:   * if a is **true** or **non-zero** this evaluates to **false** * if a is **false** or **zero** this evaluates to **true** |

## The while loop

**int x = 50;**

**while (x < 100)** // Stay inside this loop as long as *x* is smaller than 100.

**{** // The code between { and } will be executed repeatedly until leaving the loop

**x = x + 1;** // Add 1 to the variable *x*

**}**

The code inside the loop above will be executed 50 times. (*x* begins at 50 and is then incremented by one until it reaches 100).

Another example:

**while(1)**

**{**

// We will stay inside this loop forever, because anything that is **non-zero** also evaluates to **true**.

**}**

## Compiling

Finally, after writing your code, you need to compile it, before you can download to the robot. This basically means “translate the human readable text to something that the robot can understand and execute”.