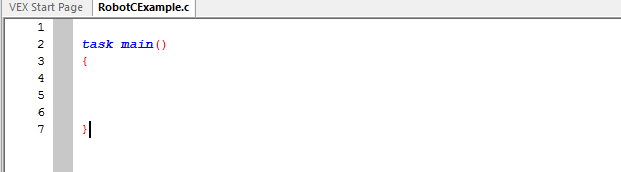
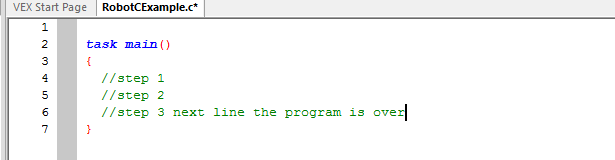
**Basic Programming:**

**Main:**

Now that the motors and sensors are setup we should talk about basic programming. The first thing a program will execute is starting in task main() which is just like a normal main in C/C++, yet in robotC main is a task which is not the same, but for all practical purpose we will consider it the same for now. Any actions you want to happen should be called within here in order.



In the bellow picture the program will do step1, step2, step 3, and then stop, but if you end the program with a motor on it will stay on, so you must shut of all motors when done.



**Variables:**

A variable is just way to store data just like math when you solve for x. In programming a variable has to have a data type which means what does the variable hold. Variables we will use will hold a numerical number or a Boolean which means true or false.

Three parts of declaring a variable: **“Data Type” “Name” “Value”**

**Data Type:** The data type is written as int (integer), bool (Boolean), or float (number with decimal)

**Name:** The variable name can be any name with no spaces, but it is recommended and sometimes enforced to not start the name with a numerical number.

**Value:** The value will be a number or true / flase.

When declaring a variable or assigning a variable a value the line of code must end with a semicolon.

**Declaring a variable:**

First lets create a variable which holds speed of a motor which we will call motorSpeed and set the data type as an integer followed by a semicolon.

int motorSpeed;

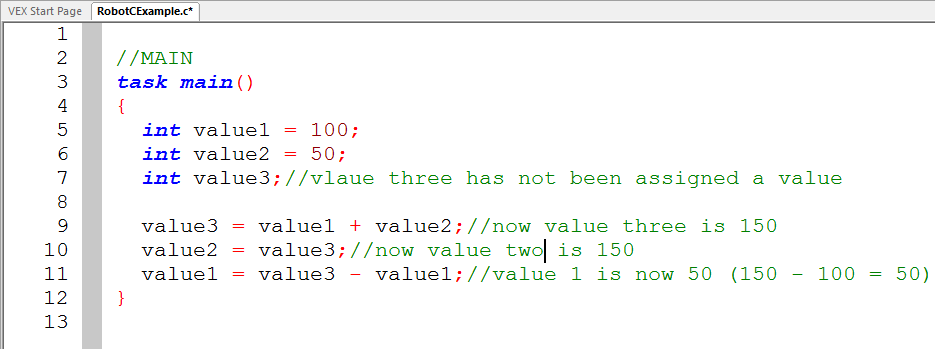
Second we will declare a variable which will hold the state of a button which will call isButtonPushed with will be data type Boolean followed by a semicolon.

Bool isButtonPushed;

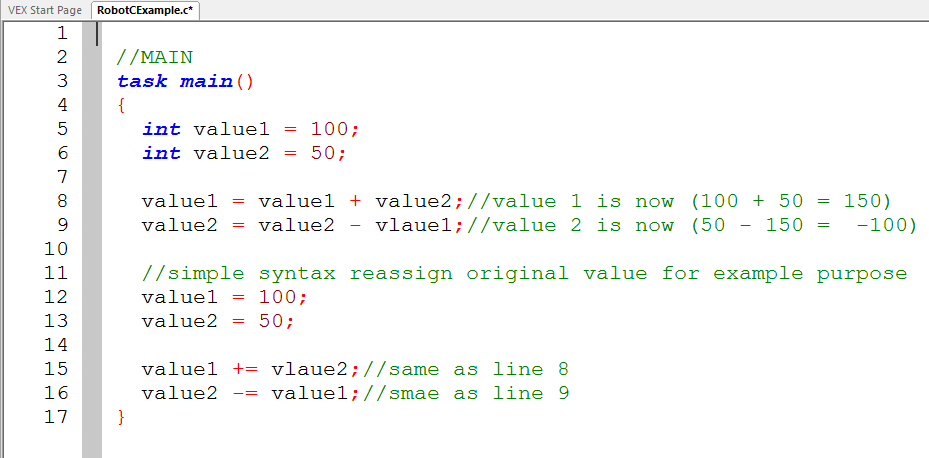
You can also declare and assign a value of a variable at the same time. Here is how to declare and set a speed of 77 to the motor speed variable. Also note you still need a semicolon.

int motorSpeed = 77;

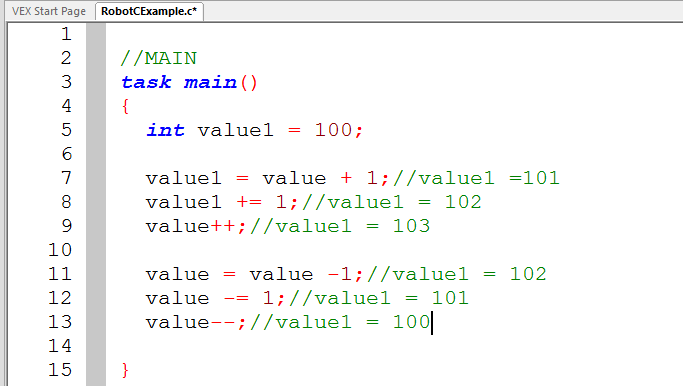
To alter a variable or assign a value to a variable you can assign a value with equals sign, add two values together with addition symbol, and subtract a variable with a minus sign. The variable must already be declared before you assign a value to it. Here is an example of declare two variables, assign them values, adding them and subtracting them.



A simpler syntax you can use to assign and add or decrease a value is += and -= which will add a value to the assigning value and assign the new total to the original variable. The code snippet below shows this example and how its equivalent.

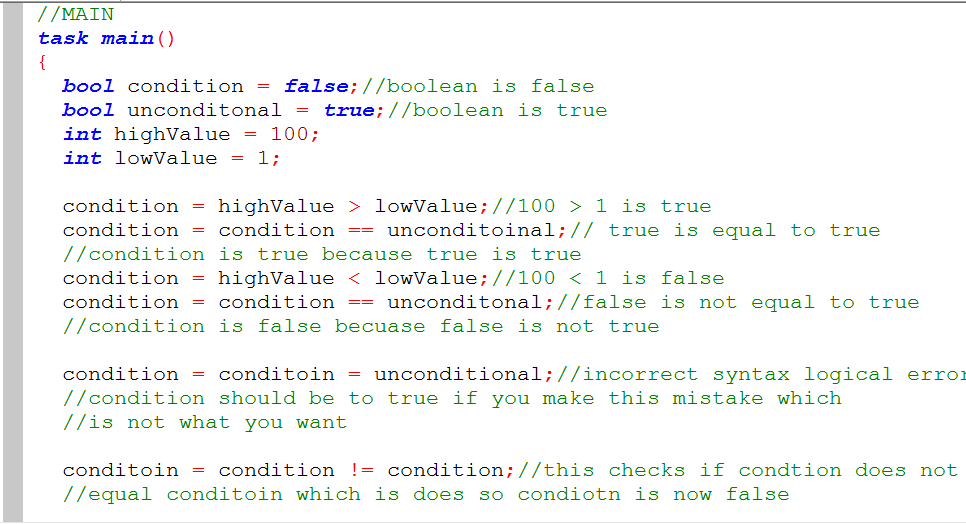


The postfix increment and decrement is an easy way to change a variable by one value. Here is an example of how to do the same assignment three ways. The first three assigments increase the value of value1 by 1 while the next three lines decrease the value of value1 by 1.



**Comparison conditions and the Boolean variable:**

Just like the int variable there is a Boolean variable called bool, but a Boolean variable can only be true or false. You can manually assign the value of the Boolean variable as true or value or you can get the conditional value from a comparison or another variable such as if a button is pushed. To compare integer variable you can use less than “<”, greater than “>”, less than or equal “<=”, greater than or equal “>=”, if the values are equivalent “==” which is two equal sign and should not be confused with the assignment operator “=” one equal sign, or if the two value are not equivalent “!=” which does not equal. Here is an example of comparison and the Boolean variable declaration and assignment.



**If-statements:**

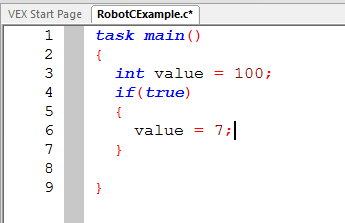
With the use of Boolean variables or conditional statements we can have the program choose what code to execute. When using if-statements you should put the code you want inside curly brackets to make sure the correct code is executed. To make a block of code you use the curly brackets around the code. This is a left curly bracket first followed by the lines of code then a right curly bracket at the end. If-statement do not need a semicolon at the end of the line.

**If, else if, or else statement:** You can check a statement by doing an if-statement. If you wanted to check another statement if the first if failed you would do an else if. You can do as many else ifs as you want. Lastly you can do an else statement which will execute if all if’s and else if’s failed the conditional comparison. This may seem confusing, but the example will clarify how to make this happen.

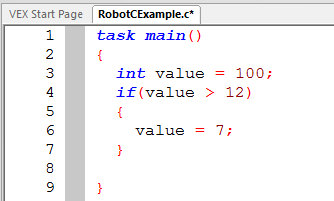
**Condition:** The condition is enclosed inside two parentheses and is a true or false condition. The condition can be a comparison or a Boolean variable. You can also compare more than one condition by a logical and operator “&&” or a logical or operator “||”.

**Code block:** Once the if-statement is true the block of code will execute. Remember the block of code

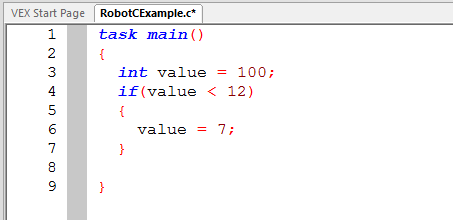
This example simply shows how to create an if-statement. The code will check if true is true which it is and assign the value variable a value of 7.



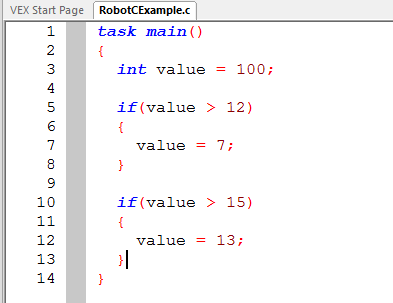
This is the same example, yet now the if- statements checks to see if variable value is greater than 12 which is true because 100 is greater than 12.



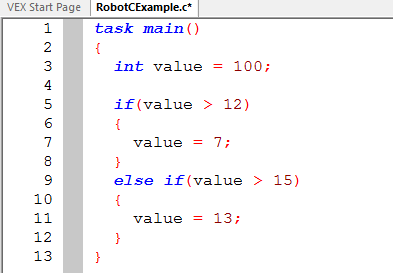
Now the example checks to see if value variable is less than 12 which it is not so the if statement if false and the code inside the curly bracket is not executed meaning the value of variable value is 100 not 7.



Now that the first if statement has failed to be true we can check if another statement is true which an else if-statement. An else if-statement is not the same as a separate if because the else if-statement is only check if the first if-statement fails. In this two if-statement example the first if is true so the variable value is assigned a value of 7 then the next if statement is checked which is true and the variable value is assigned a value of 13.

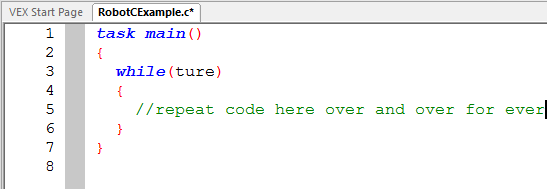


In this if-statement and else if-statement the first if-statement is true assigning a value of 7 to variable value, but the else if-statement is never checked because the first if statement was true, so the value of the variable value at the end is 7 not 13.



**While loops:**

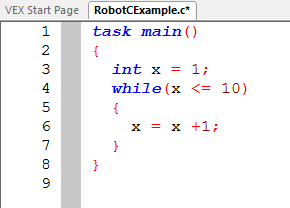
While-loops are used to repeat a portion of code instructions. You can have a while loop repeat forever by having the condition it checks if it should repeat to true. If a while loops repeats forever it is called an infinite loop. To create an infinite loop you can use the following syntax.



The word while indicates we want to create a while loop. Whatever is inside the parentheses is the condition that will be checked just like the if-statement. Since the condition we are checking is true than the while loop is always true and will repeat the code inside the curly brackets forever.

You will be using while loops for check if a sensor read a certain value so the robot will repeat the same action until the sensor value has reached the desired state. You will see this later on, but you must understand what a while loop will do.

A simple example of a while loop is to have the while loop increment a number while the while loop checks to see if the number is below another number. The example below will loop ten times than stop.



**Functions:**

**Four parts of a function: “data type” “name” “parameters” “block of code”**

A function is a piece of code which you can call over and over again without retyping every step which saves time and looks nicer. We will focus on functions which return nothing.

**Data Type**: The data type is what the function returns which we are returning nothing so it will be called and written as *void*.

**Name:** The name of the function will how we will refer to the function to execute within main or another function.

**Parameters:** The parameters are the amount and type of variables we will use to execute the function. The type of variables will be int (integer), bool (true or false), float (number with a decimal). You must also name the parameters which are grouped with the data type and separated by commas. The whole parameters are enclosed in parentheses.

Ex We want a function to turn on motor1 and motor2 with a value.

**Parameters:**  (int motorSpeed1, int motorSpeed2)

So far we have the data type void our name will be “twoMotorFunction” and we wanted two parameters of motor speed so our function will look like this

void twoMotorFunction(int motorSpeed1, int motorSpeed2)

The last part of a function is the code which will be executed. The code will be inside a block of curly brackets meaning you will need a left curly bracket, lines of code, and lastly a right curly bracket.

**Block:**

{

//lines of code

}

The finished function will like :

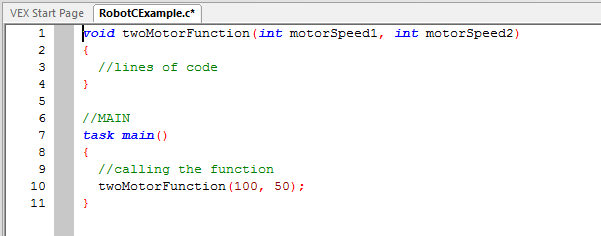
void twoMotorFunction(int motorSpeed1, int motorSpeed2)

{

//lines of code

}

To call the function you put the name of the function, the arguments of the parameters, and end the call by a semicolon for example to call the function twoMotorFunction it would look like this.

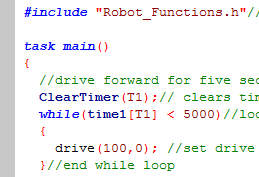


As you can see the function is written above main which must always happen or main will not know what the function is. Line number 10 show the calling of the function which sets the motorSpeed1 to 100 and motorSpeed2 to 50 and then runs any code inside the function.

**Including other files:**

You can include separate code file inside your code so you don’t have to keep writing the same code over and over again. The idea is to have a file that only has functions which will used again and again with different code. To use code from another file you must include the file before you call the code. The syntax to add a file in C is #include “filename.filetype” where filename is the name you gave to the file and the file type will be a header file ‘.h’ or source file ’.c’ without the quotation marks. For are purpose it doesn’t matter what file type you use, but to distinguish the file is not meant to be used by itself I would recommend using ‘.h’.

In the first example labeled Problem 01 you will use this include of file “Robot\_Functions.h” which if you open the file “Robot\_Functions.h” you will see the function drive(short throttle, short steering) declared. Also not the including of the file is before the use of the function with in the file.



**Resources:**

[RobotC Wiki](http://www.robotc.net/wikiarchive/Tutorials/Getting_Started/Getting_Started_with_the_VEX_CORTEX)