Learning to Write Programs in C

Jeffrey La Favre

October 21, 2015

Your next project in GEAR will include learning how to program a robot. The robot you will program is named ActivityBot (https://www.parallax.com/product/32500). ActivityBot is programmed using the **C Programming Language**. A programming language provides a method for writing code that can be translated into a set of machine instructions that a computer or microprocessor will follow.

In order to create programs using a programming language, you need to have programming software installed on your computer. Another name for programming software is **Integrated Development Environment** (or **IDE** for short). There is a free IDE program available on the Internet that you can use to write programs in C and it is available for download at this address:

http://sourceforge.net/projects/orwelldevcpp/

The name of this IDE is **Dev-C++**. With **Dev-C++** you can write code in C language or C++ language. You will just use it to write programs in C for this series of lessons.

Go ahead and install **Dev-C++** on your computer. Then you can start writing your own computer programs in C. However, keep in mind that a different IDE is required when you start writing programs for the ActivityBot robot. For now, we just want to get you started learning C. Soon you will start writing programs for the robot, but what you learn using **Dev-C++** will still apply to your robot programs. The IDE for ActivityBot requires that the robot be connected to your computer to test out the programs you write in C. By starting your study of C utilizing **Dev-C++**, you don't need to have the robot to run a program.

Even a simple program requires these steps to complete:

1. Write code
2. Compile (translates the language code to machine code)
3. Load (put machine code into computer memory)
4. Run (microprocessor of computer reads code and executes it)

Your first program: "Hello, World"

The first program to write when learning a programming language is commonly a program that just outputs the message "Hello, World." By writing this program and running it, you will start your journey in learning the C language. At the top of the next page you will find the code for "Hello, World."

```
#include <stdio.h>

main()
    {
    printf("Hello, World\n");
    }
```

The above program is just about as simple as it gets in C language programming. All this program does is print the message "Hello, World" on the computer screen when it is run. There is nothing particularly exciting about this program. However, if you can make this program work, that is exciting. It is exciting because it means you have started to learn the C language. When you have a good understanding of C, you will be able to write all sorts of interesting computer programs. The possibilities are only limited by your imagination.

Shortly you will learn more about the code for the Hello World program, but first, let's see if you can make this program run. Follow these steps:

1. Start the **Dev-C++** program (you should find it on your list of computer programs, on my Windows 7 computer the actual program is located here: C:\Program Files (x86)\Dev-Cpp\devcpp.exe)
2. Open the **File** menu of **Dev-C++** and select **New**, then select **Source File**
3. Copy the program above and enter it into the text window of **Dev-C++** (you can just copy and paste if you like or type it in yourself using the keyboard, but make sure it is exactly as given above).
4. Open the **File** menu and select **Save**
5. In the **Save** dialog box that opens, open up the drop-down list for **Save as type:** and select **C source files (*.c)**. Then enter a file name for your program in the **File name** slot (I suggest you name it HelloWorld). Check the location where you will be saving the file in the **Save in** slot at the top of the dialog box (you may need to find the file later). Then click the **Save** button.
6. Now you are ready to test your program. Open the **Execute** menu and select **Compile and Run**. If you don't have any errors in your program, then it should open another window and print the message "Hello, World."

Hopefully you were able to get your computer to display "Hello, World." Now, let's start learning about the C code in this program.

Perhaps the best place to start is by defining **Functions**. Functions are things that accomplish specific tasks. In the code for Hello World, there are two functions: **main** and **printf**. The function name is always followed by a set of parentheses (). There may be no contents inside the parentheses, as in **main()**, or there may be **arguments** inside the parentheses as in **printf("Hello, World\n")**. **"Hello, World\n"** is the argument for the function **printf** in the code above. In any case, the parentheses must follow the function name. Some functions require arguments and some do not. There is no particular

rule to remember regarding which functions require arguments and which do not. This is something you need to memorize for each function. If you forget, you can always look the function up in a reference section of a C language book.

The task accomplished by the function **printf** is *output of text to the computer screen*. The text to output is that which is provided by the argument: **("Hello, World\n")**. In this case the double quotes (") indicate that we want the computer to display the actual text given between the double quotes. You may be wondering why **\n** was not displayed on the computer screen after Hello, World. There are a number of invisible characters and **\n** represents the invisible character named **new line**. By including **\n**, we are instructing the computer cursor to jump down to the next line after displaying Hello, World. Why would we want to do this? Well, if our program included more text to display on the screen, but we wanted that text to be displayed on another line, then we need to include the new line character before providing additional text.

Here are some additional invisible characters: **\t** is the tab character, add this if you want the computer cursor to jump one tab distance to the right after displaying some text – this can be useful if you want to put some space between sections of text on one line; **\b** is the backspace character. In addition there are these two: **\"** will actually display the double quotes character; **** will actually display the backslash character.

Now we should go on to the **main()** function. Every C program **MUST** include the **main()** function. The **main()** function is always listed before any other function. When the computer starts executing a C program, it starts with the **main()** function. This is where the action starts. Commonly, the bulk of the C code is found inside the **main()** function. The code that belongs inside the **main()** function is marked by curly braces **{}**. Notice that the first line below **main()** contains the character **{**, which marks the beginning of the code block contained inside the **main()** function. Notice that the last line of code contains the character **}**, which marks the end of the code block contained inside the **main()** function.

Our simple Hello World program contains only one line between the **{** and **}** characters of the **main()** function. However, most programs will contain many lines of code inside the **main()** code block.

Each line of code following a function is called a **statement**. Each statement must have a semicolon (;) to mark the end of the statement. Notice that a semicolon is included at the end of this line of code:

printf(“Hello, World\n”);

Now let’s look at the first line of the code:

#include <stdio.h>

Most of the functions you will use in your program are contained in libraries. The function **printf** is contained in the *standard input/output* library named **stdio.h** for short. Therefore, we need to reference that library before we can use the function **printf**. That is done by this line of code:

#include <stdio.h>

You have now completed your first lesson on learning C! Next you will learn about variables and calculations.