Learning to Write Programs in C

Jeffrey La Favre

October 21, 2015

Updated March 13, 2026

Your next project in GEAR will include learning how to program a robot. The robot you will program is named ActivityBot360 (https://www.parallax.com/product/activitybot-360-robot-kit/). ActivityBot360 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 use programming software. Another name for programming software is **Integrated Development Environment** (or **IDE** for short).

If you are using an Apple computer we suggest you install the IDE app named CodeRunner 4 available at this link: https://apps.apple.com/us/app/coderunner-4/id955297617?mt=12

If you don't want to install any software on your computer you can use an online IDE available at this link: https://www.onlinegdb.com/online_c_compiler

If you want to install software on a computer that is running Windows, there is a free IDE program available. The name of this IDE is **CodeBlocks.** As of this writing, the download page is located here: https://www.codeblocks.org/downloads/binaries/

For a Windows computer, you want to select a binary version of the software that includes the GCC/G++/GFortran compiler (currently this link name: codeblocks-25.03mingw-setup.exe).

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 ActivityBot360 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 CodeBlocks will still apply to your robot programs. The IDE for ActivityBot360 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 CodeBlocks, 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. Below you will find the code for “Hello, World.”

```
#include <stdio.h>
int 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 (if you are not using CodeBlocks the steps will be slightly different):

1. Start the **CodeBlocks** program (you should find it on your list of computer programs)

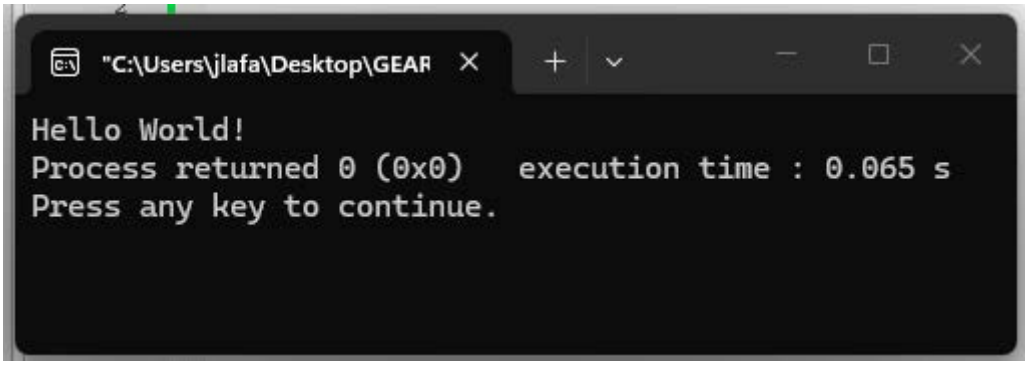
2. Open the **File** menu and select **New**, then select **Empty file**

3. Using your keyboard type the code above into the text window of CodeBlocks .

4. Open the **File** menu and select **Save file**.

5. In the **Save file** dialog box that opens, make sure the **Save as type**: is set for **C/C++ files** 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 **Build** menu and select **Build and Run**. If you don't have any errors in your program, then it should open another window and print the message "Hello, World." (see image below)



Notice in the Terminal window above that the text Hello World! is displayed. After the program runs, additional information is supplied. In this case we learn that the computer took 65 milliseconds to execute the program.

IMPORTANT: Notice the last line of text displayed: Press any key to continue. After you are finished reading the information in the Terminal window, press any key on the keyboard to close the Terminal Window. YOU WON'T BE ABLE TO BUILD AND RUN ANY FURTHER PROGRAMS IN CODEBLOCKS UNTIL YOU CLOSE THE TERMINAL WINDOW.

(continued on next page)

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

(continued on next page)

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 commonly listed before other functions. When the computer starts executing a C program, it starts with the main() function. This is where the action starts. Sometimes, 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>
```

Many 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.