Learnem.com

Programming Courses

Learn'em

Programming In C

By: Siamak Sarmady

LEARN'EM PROGRAMMING COURSES

Programming in C

Ver. 2.08.01

By: Siamak Sarmady

Table of Contents

Lesson 1

Quick Start with C

C programming language is perhaps the most popular programming language. C was created in 1972 by Dennis Ritchie at the Bell Labs in USA as a part of UNIX operating system. C was also used to develop some parts of this operating system. From that time C programming language has been the de facto programming language when fast programs are needed or the software needs to interact with the hardware in some way. Most of the operating systems like Linux, Windows™, and Mac™ are either developed in C language or use this language for most parts of the operating system and the tools coming with it.

This course is a quick course on C Programming language. In our first lesson we will first write our first C program. We will then learn about printing to screen, variables and functions. We assume that you are familiar with at least one of the popular operating systems.

For this course you can use the following compilers or Programming Environments.

- Gcc and cc in Unix and Linux operating systems
- Borland C or Turbo C in DOS operating system or in Command line environment of windows operating system
- "Bloodshed Dev-Cpp" integrated development environment (IDE) gives you a complete and compact programming environment. It comes with "MinGW" and "GCC" C Compilers and you should not need anything else for this course.

We use "Bloodshed Dev-Cpp" for this course and we suggest you also use it. "Bloodshed Dev-Cpp" is free and it can be downloaded from the website http://www.bloodshed.net (currently under the URL http://www.bloodshed.net/dev/devcpp.html).

Your first C program

Let's write our first C program.

Example 1-1: example1-1.c

```
#include <stdio.h>
main()
{
 printf("Hello World!\n");
```

```
 system("pause"); //this line is only needed under windows
}
```

First step for running this program is to make a text file containing above code. Be sure that the file is a pure text file. You must save the text file with .c extension.

Then you must compile the source code. The result of compile step is an executable file that you can run it.

If you are running your example on Unix/Linux type operating systems you can remove the line which is commented to be necessary just under Windows.

Compiling and Running on Unix/Linux

To compile program under Unix operating system use the command:

```
$ cc test.c
```

and under linux type

```
$ gcc test.c
```

The resulting executable file is a.out file. To run this executable you must type:

```
$./a.out
```

Program output must appear on your screen.

```
Hello World!
```

Compiling and Running under Windows with Dev-CPP

To develop, compile and run the program in Bloodshed environment follow below steps. If you have problem working with your compiler you may ask your problem in our support forums.

1- Run bloodshed and select "New -> Project" from File menu. In the appeared window enter a name for your project (Figure 1.1). Also select "Console Application", "C Project" and "Make default Language" as the settings of your application.

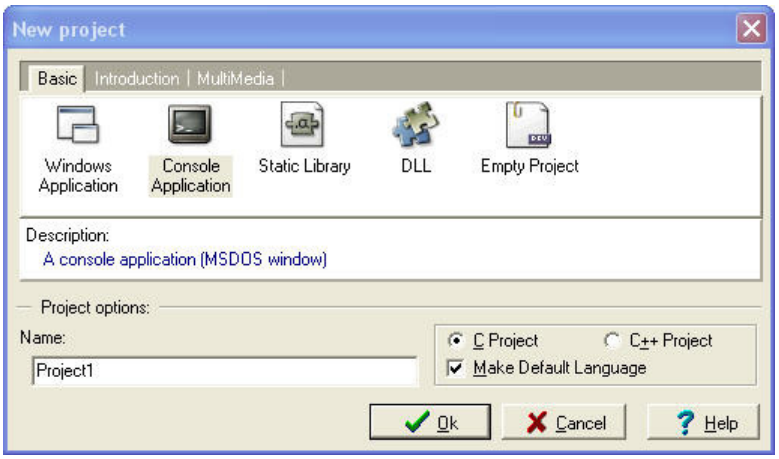

FIGURE 1.1: Creating a new project in Bloodshed Dev-CPP.

2- A window will open and ask for a place to save the project. We suggest that you create a separate directory for each project to avoid their files being mixed with each other (or worse, overwrite each other). A project is a set of related C programming language files. In this case we just have a single C file in our project but big projects may have even hundreds of C files.

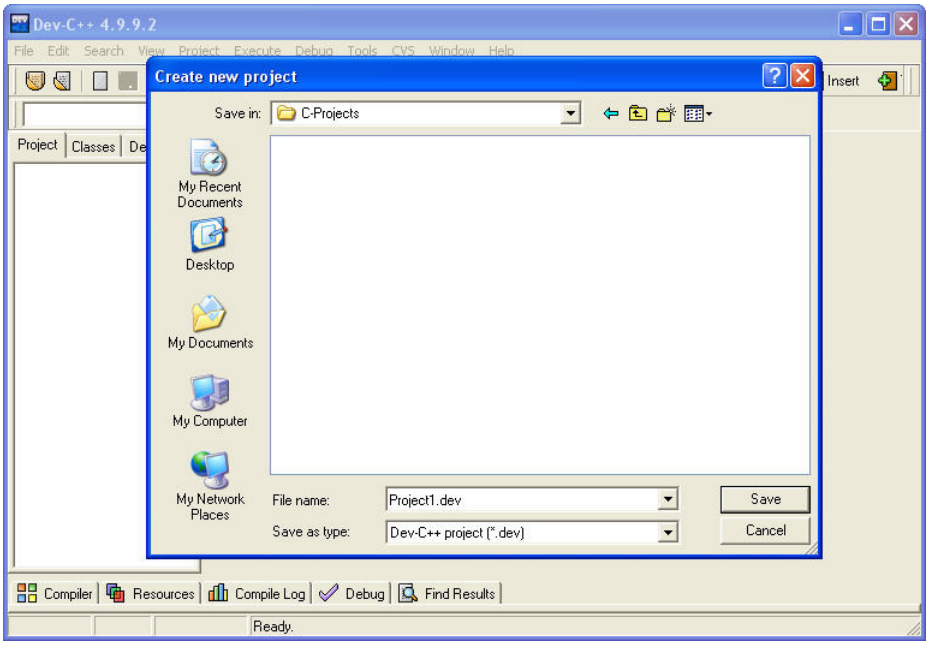

FIGURE 1.2: Saving a project in Bloodshed Dev-CPP.

3- Dev-CPP creates the project and generates a sample C language file for you. Dev-CPP creates this first sample C program with the file name "main.c". You should change the source code to the source code of our Example 1-1 (Figure 1.3 and 1.4). After changing the code, press "Save File" button. This will give you the opportunity to change the default "main.c" file name to whatever file name you prefer. You might want to save the file with the related example number (**example1-1.c** in this case) or you can leave it as it is.

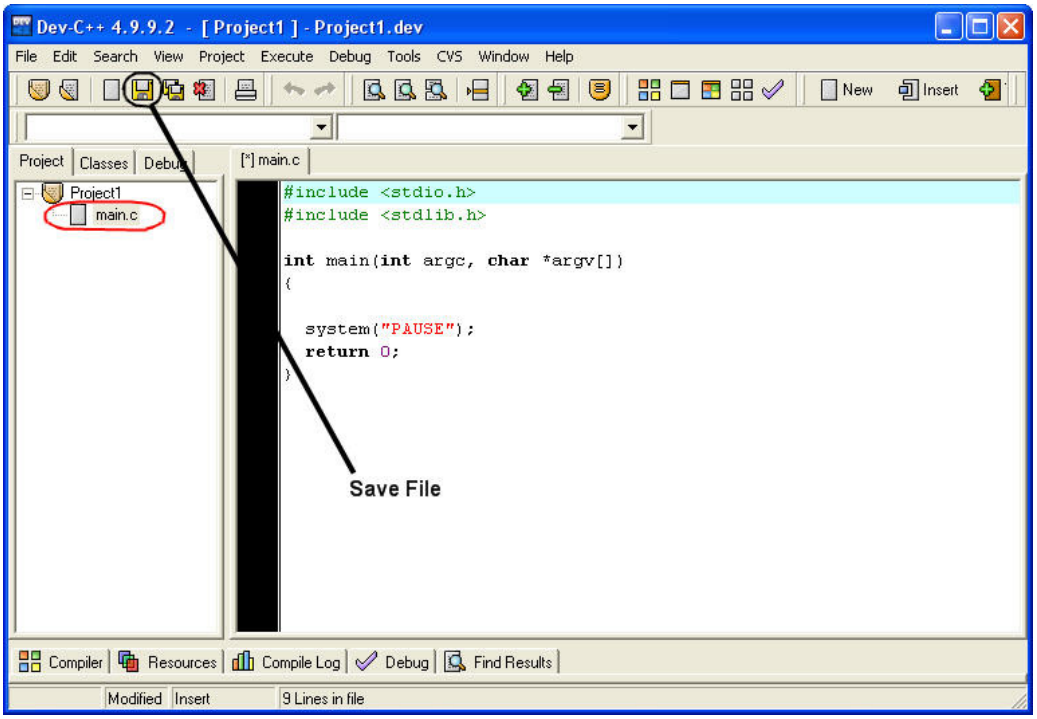

FIGURE 1.3: New project is created in Bloodshed Dev-CPP and a simple code is generated.

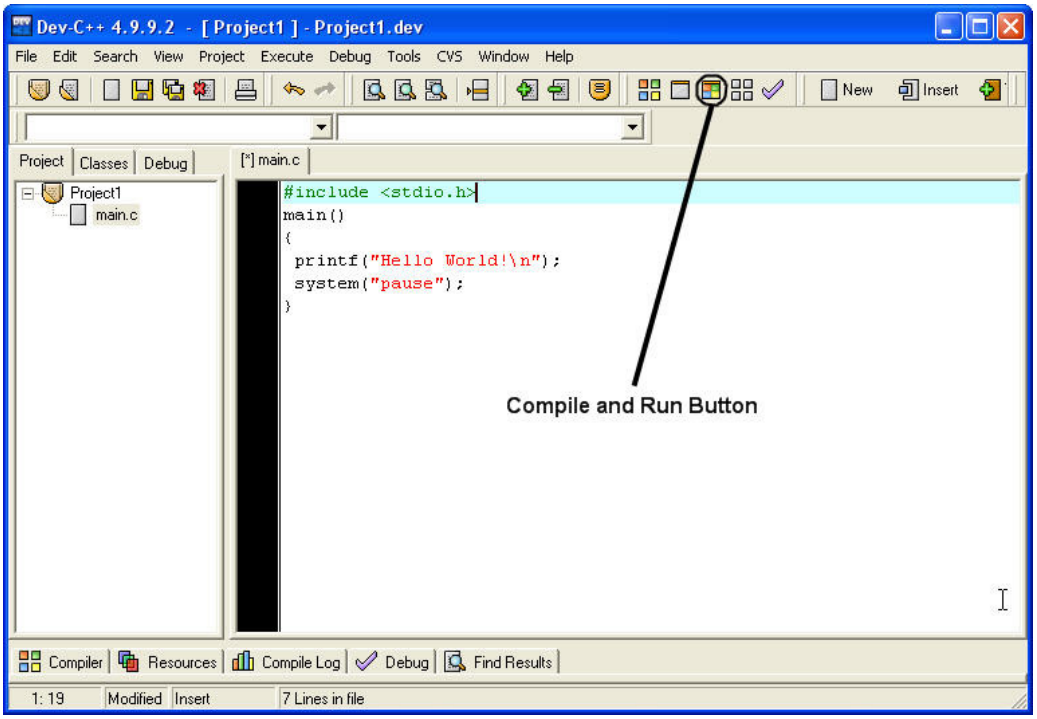

FIGURE 1.4: Change the code to our Example 1-1.

4- Click on "Compile and Run" button (or press F9 key). This will compile the code to machine executable format and run the program (Figure 1.5). To close the output window press any key in output window.

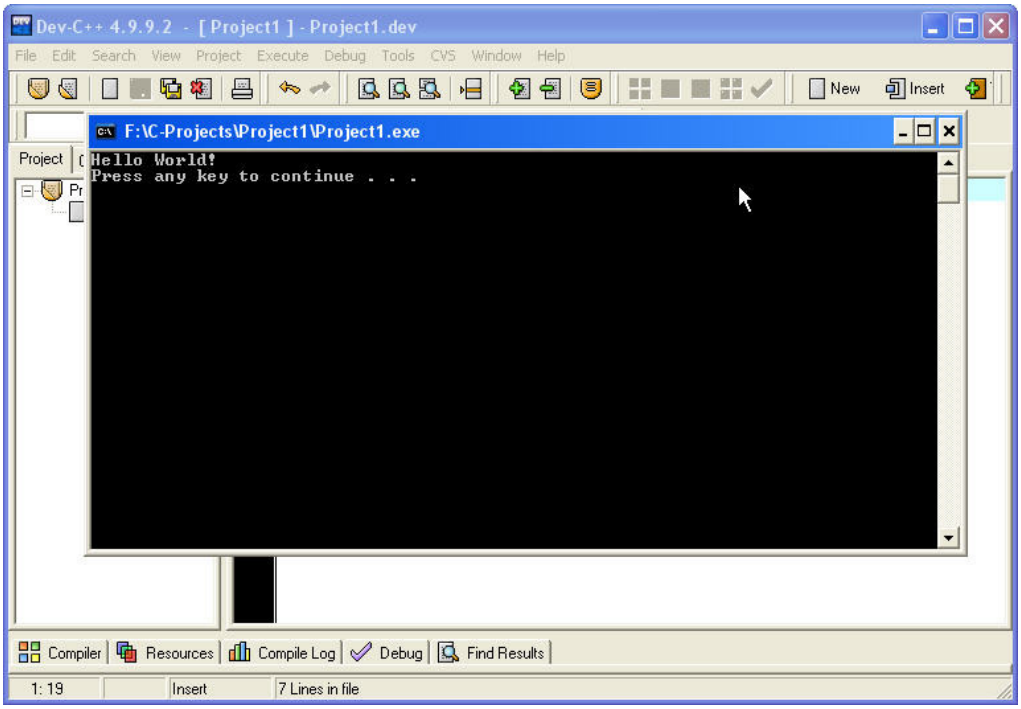

FIGURE 1.5: Output window shows the result of the program.

If you look into the directory which you saved your project, you will find 5 different files:

- **main.c** (main C program file)
- **main.o** (intermediate compile file called "object file")
- **Makefile.win** (Make file is used by Dev-CPP compile settings of your project)
- **Project1.dev** (Project File contains Dev-CPP settings of your project)
- **Project1.exe** (Executable file which can be run independent from Dev-CPP or C Compiler)

The final product of your C program is the windows executable (.exe) file. You will normally distribute the executables of your software to users and keep the source code (*.c) for your own.

Details of Test program

- **#include <stdio.h>**
  Tells C compiler to include the file "stdio.h" in this point of your C program before starting compile step. This "include file" contains several definitions, declarations etc.

- **main()**
  C program consist of one or more functions. Functions are building blocks of C programs. main() function is different from other functions by that it is the start point of program

execution. Our program contains only function while complicated programs may contain thousands.

- **{**
  Opening brace marks the start of a block. Closing brace will mark its end. This one marks main () function start

- **printf("Hello world!");**
  This line of code prints the statement between quotation marks on your output screen. \n tells program to start a new line in output screen.

- Each command line in C ends with ";" character. Control statements are exceptions. You will soon be able to determine when you must use ; to end a line of code.

- **system("pause");**
  The output window will close in Windows™, immediately after program execution has been finished. In this way you will not be able to see results of the execution (as it happens very fast). We have put this command to pause the window and wait for a keystroke before closing the window. **You can remove this line from our examples if you do not use Windows operating system.** This command actually sends the "pause" command to windows operating system and windows runs the its "pause" command at this point. We will learn more about this command in later lessons.

- **}**
  closes main() function.

This program contains only one function while complicated programs may contain several functions.

Data Types and Variables

C uses several data types of data. These include characters, integer numbers and float numbers. In C language you must declare a variable before you can use it. By declaring a variable to be an integer or a character for example will let computer to allocate memory space for storing and interpreting data properly.

Naming a variable

It is better that you use meaningful names for your variables even if this causes them to become long names. Also take this in mind that C is case sensitive. A variable named "COUNTER" is different from a variable named "counter".

Functions and commands are all case sensitive in C Programming language. You can use letters, digits and underscore _ character to make your variable names. Variable names can be up to 31 characters in ANSI C language.

The declaration of variables must take place just after the opening brace of a block. For example we can declare variables for main() function as below code:

```
main()
{
int count;
float sum,area;
.
.
.
}
```

First character in a variable name must be a letter or an underscore character. It cannot be a C programming language-reserved word (i.e. Commands and pre defined function names etc). An example for using variables comes below:

Example 1-2: example1-2.c

```
#include<stdio.h>
main()
{
 int sum;
 sum=12;
 sum=sum+5;
 printf("Sum is %d",sum);
 system("pause");
}
```

General form for declaring a variable is:

Type name;

The line sum=sum+5; means: Increase value of sum by 5. We can also write this as sum+=5; in C programming language. printf function will print the following:

```
Sum is 17
```

In fact %d is the placeholder for integer variable value that its name comes after double quotes.

Common data types are:

```
int      integer
long     long integer
float    float number
double   long float
char     character
```

Other placeholders are:

```
%d     decimal integer
%ld    decimal long integer
%s     string or character array
%f     float number
%e     double (long float)
```

printf () function used in this example contains two sections. First section is a string enclosed in double quotes. It is called a format string. It determines output format for printf function. Second section is "variable list" section.

We include placeholders for each variable listed in variable list to determine its output place in final output text of printf function.

Control characters

As you saw in previous examples \n control character makes a new line in output. Other control characters are:

```
\n New line
\t tab
\r carriage return
\f form feed
\v vertical tab
```

Multiple functions

Look at this example:

Example 1-3: example1-3.c

```
#include<stdio.h>
main()
{
  printf("I am going inside test function now\n");
  test();
  printf("\nNow I am back from test function\n");
  system("pause");
}

test()
{
  int a,b;
  a=1;
  b=a+100;
  printf("a is %d and b is %d",a,b);
}
```

In this example we have written an additional function. We have called this function from inside main function. When we call the function, program continues inside test () function and after it reached end of it, control returns to the point just after test() function call in main(). You see declaring a function and calling it, is an easy task. Just pay attention that we used ";" when we called the function but not when we were declaring it.

We finish this lesson here. Now try to do lesson exercises and know the point that you will not learn anything if you do not do programming exercises.

Exercises

- Write, compile and test your programs under "Bloodshed Dev-CPP" under windows or Gcc under Linux/UNIX.
- Paid students need to submit their exercises inside e-learning virtual campus. Corrected exercises will be available inside virtual campus.
- If you have obtained the e-Book only, you can discuss your homework questions in Learnem.com support forums (in registered e-book users section).

---

1. **What is the exact output result of this code?**

```
#include <stdio.h>
main()
{
printf("Hi\nthere\nWhat is the output\n?");
}
```

2. **Write a program that declares two floating numbers. Initialize them with float values. Then print their sum and multiplication in two separate lines.**

3. **Write the output result of "multiple function example" in this lesson (run the program and see it).**

4. **Why these variable names are not valid?**

```
test$var
my counter
9count
Float
```