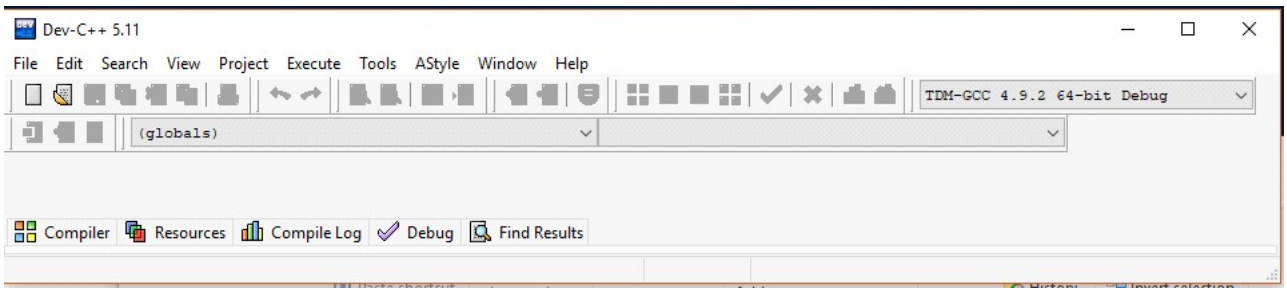
CMPE108 - EXPERIMENT 1
DEVC++ AND C PROGRAMMING[1]

Aims

1. Learning primary functions of DEVC++ 5.11
2. Introduction to C Programming
3. Running C programs using Microsoft Visual Studio

Starting DevCpp

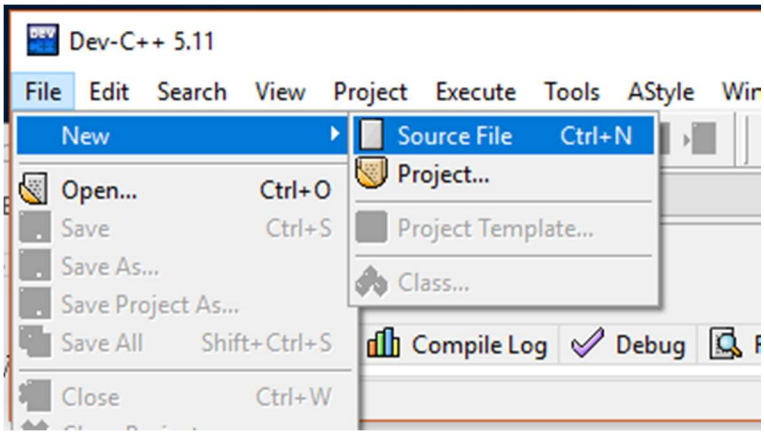
DevCpp 5.11 starts with the following toolbars:



You shall select the suitable compiler for your computer from the menu **Tools – Compiler options** or from the **compiler-select** box on the toolbar. For 64 bit computers, select **TDM-GCC #### 64-bit debug** or for 32 bit computers, you may select **TDM-GCC #### 32-bit debug.**

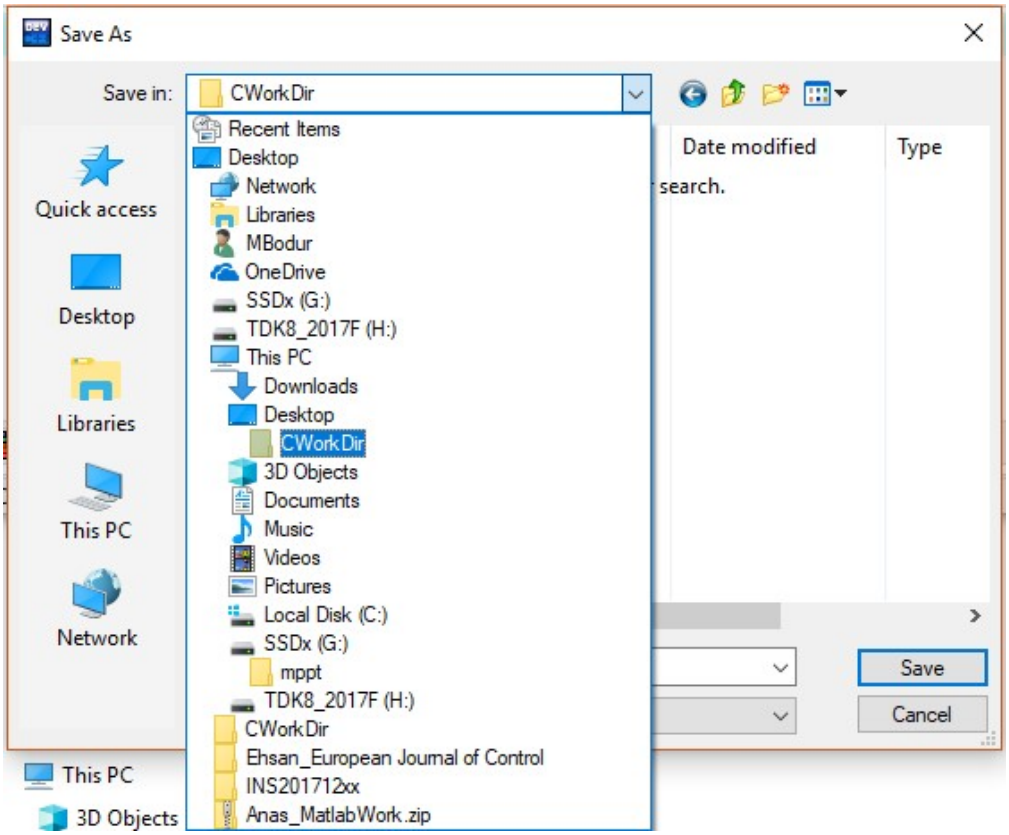
Starting C file

DevCpp can compile a single C file without a project. To create a new C source file, you shall click on **file**, and select **new**, then select **source file** from the pop-up menu lists.

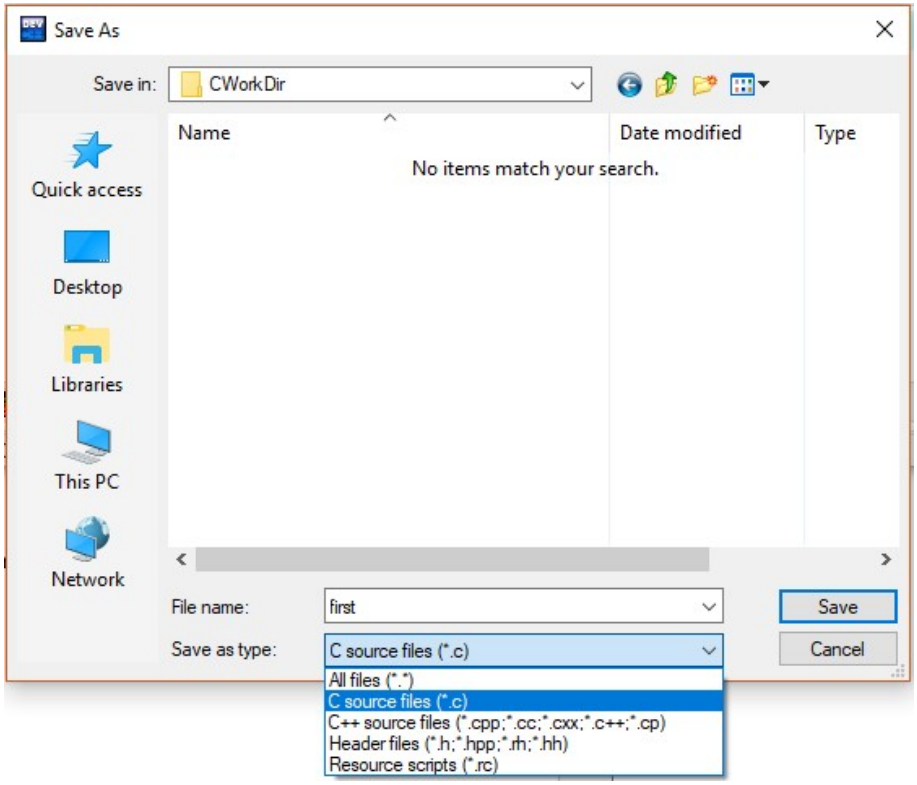


[1] Updated to DevCpp5.11 by Dr. Mehmet Bodur. Original prepared by Cem Kalyoncu and Yıltan Bitirim using BLGM101 laboratory sheets

You may start a new file also by the keys Ctrl+N. The file starts with the name **Untitled1**. You shall save the file to your working directory by menu items **File, Save as**. On Save As window you shall select your working directory using the Save in: box by clicking its pop-up button, and selecting a suitable directory on the available locations.



Next, you shall give a name and a type to your new file. You may write the name directly into **File name** box, and select the type by **Save as type** menu



After selecting **C source files**, click on **Save** button to save your new file in your working directory.

Once you have your file in your working directory, now you can write and save your program code into that file.

First C Program

Now, you will write the program source code into **DevCpp editor window.** Performing these operations as you read them will help you to understand it better.

The first part that should exist in a C program is its entry function. This function should be named as "main", so that C compiler will know which method it has to call. This function should return an integer (int) after its operation. For now we will only use the value 0, which denotes that the application completed its task as expected. The following is the shortest C program that can be executed.

```
int main()
{
   return 0;
}
```

As a simple program we can use "Hello World!" program which will print "Hello World!" to the screen and exits. To perform this operation we will require standard input/output library. We can perform include operation with the following preprocessor directive. This command should be written in global scope, which resides outside the function and other structure definitions. Moreover, the

preprocessor directives, which starts with # symbol, does not need semicolon (";") afterward. Generally, preprocessor directives are place at the top of your file.

```
#include <stdio.h>
```

After adding this library file to our project, we can use **printf** (print formatted) function to print text to the screen. This function is defined in "stdio.h" file. The following code fragment will print "Hello World!" to console screen. This code should be inside our main function. However, since our program ends at `return 0;` line, printing code should be before that line. The "\n" added to the end of the text denotes that the text coming afterward will be printed on the next line.

```
printf("Hello World!\n");
```

In Visual Studio 2008, the console application that we write automatically closes after they are finished. Because of this, the result will not be available for us to read. We will use "pause" command from MS-DOS to prevent this. We can use "system" command to run programs from the operating system. However, we will need to include related library, which is standard library (stdlib.h). Therefore, following line should be added to our program. As before, it should be on top.

```
#include <stdlib.h>
```

The following command runs the pause program which will wait for user to press a key.

```
system("pause");
```

This program stops our program execution until a key is pressed. Therefore, it should be after the operation we perform. However, it should be before the end of our program, otherwise it will not be effective. You can use this command to pause your program temporarily on other occasions too.

The following is the whole program.

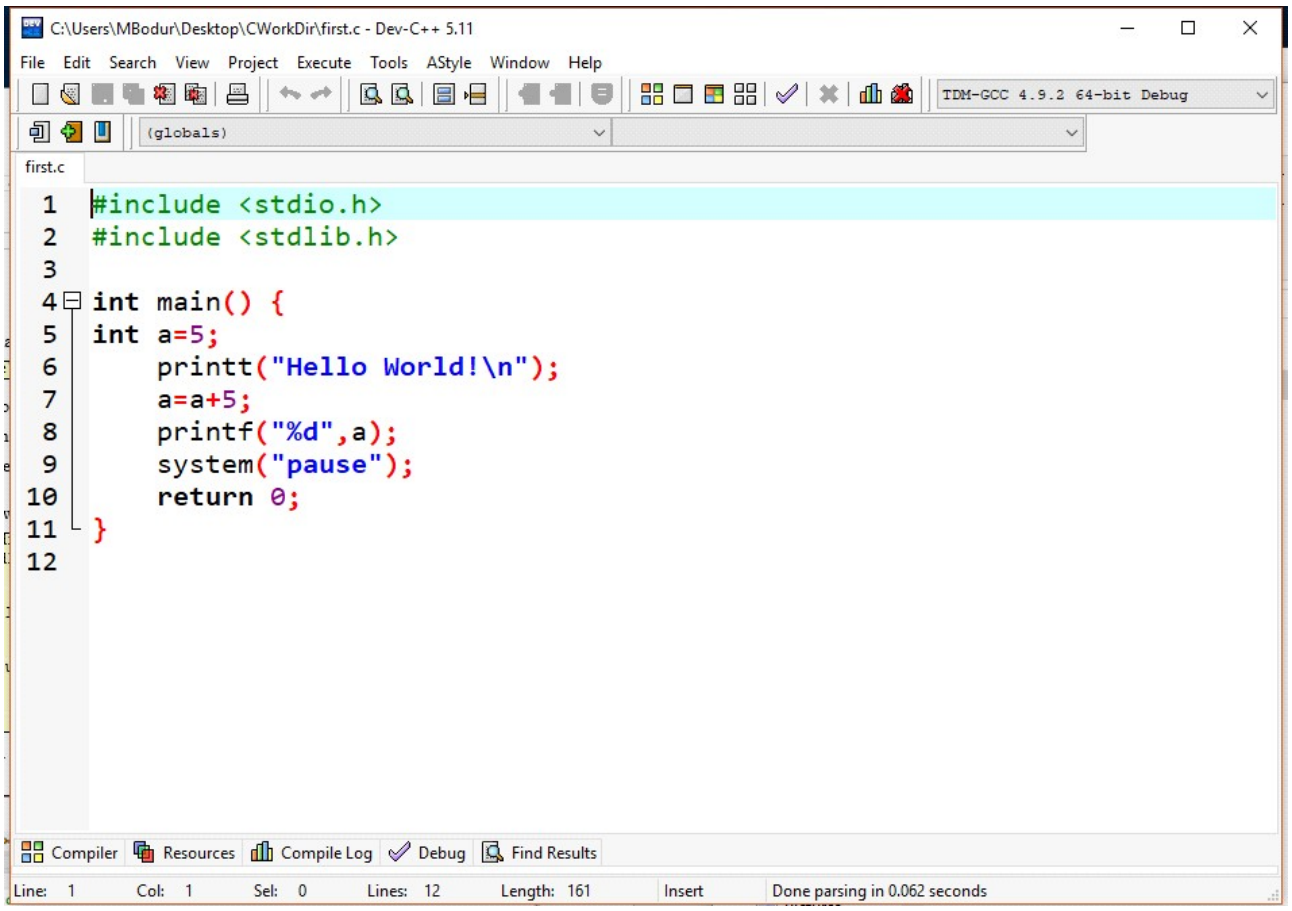
```
#include <stdio.h>
#include <stdlib.h>

int main() {
   printf("Hello World!\n");


   system("pause");

   return 0;
}
```

In the DevCpp editor window, it shall appear like

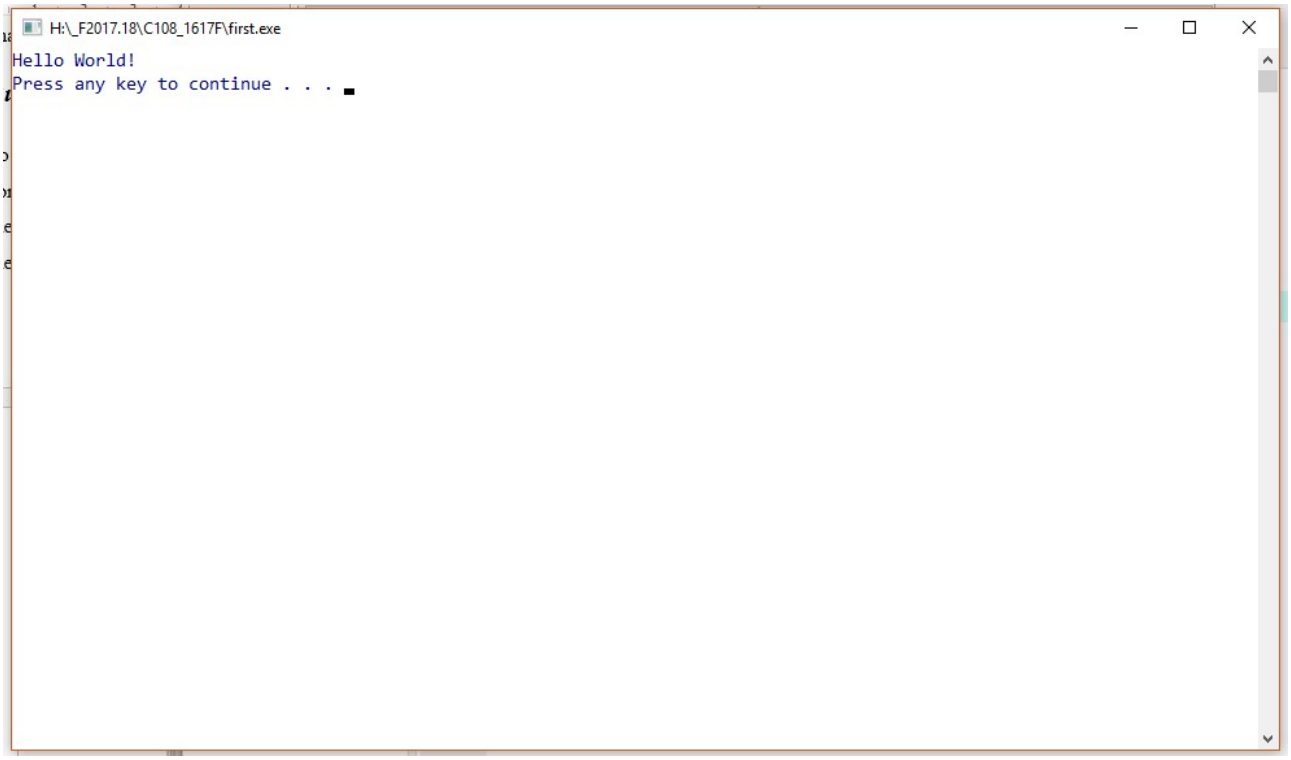


Use **file - save** from the menu, or button from the toolbar to save your file on disk whenever you change its contents

Running C Programs

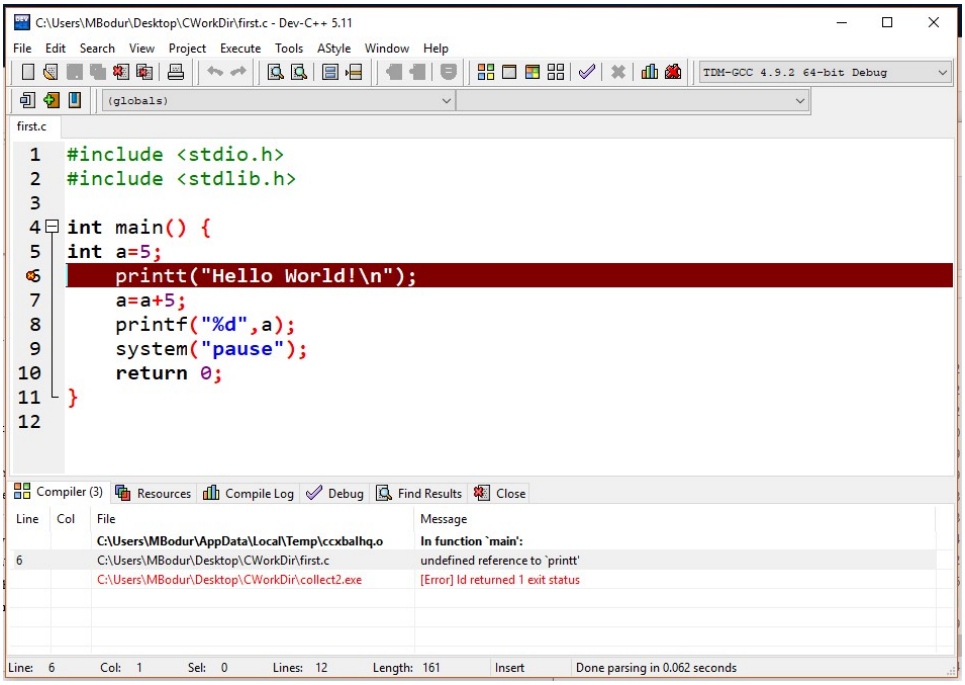
To run your program you can click on at the toolbar, or press the "F11" key, or use menu commands **Execute – Compile&run**, which provides both compile, and run functions together. If there are problems in compilation, it will mark the buggy lines, and wait for correction. Execution of

the corrected program shall start a command window, and display the output in that window.



Fixing Problems

If there is an error in your program while compiling, DevCpp will give indicate the first line which has an error, together with the GCC compiler message related to the error. For example, if printf is written mistakenly as printt, it will mark the line where the errory exists.



It is important to know that an error might be caused by one of the previous lines. If you are getting too many errors or if you are sure that the line displayed in the error list does not contain any problems,

remember to check quotes, parentheses or semicolons (“;”) in the previous lines. Most of the time missing semicolon error is given to the next line.

There are errors that cause your program to generate unexpected results. These are called logic errors and will not be handled by compiler. If you have such an error, examine your program carefully. These types of errors will be explained later.

The following is the list of most common errors. There might be errors that cannot be detected while your program is being compiled. Also common errors falling to this category is listed here.

1. Invalid symbol at include command

After “include” preprocessor command, the file name should be written in angled brackets (< >). If other characters placed outside the angled brackets, compiler continues compilation with a warning message

[Warning] extra tokens at end of #include directive

2. Quotation errors

[Error] missing terminating " character

In C programming language string literals should be surrounded by straight double quotes and each string should be written in a single line. Word processors replaces straight double quotes with opening and closing quotes, so be careful while copying from text documents.

3. main function is missing or defined multiple times

Error: undefined symbol __main or symbol __main is already defined

[Error] redefinition of 'main'

All C programs must have a main function. Check the name of it. In C language a function can be defined only once (there are exceptions to this rule) per project, therefore, you cannot add a second file and write another main function in there.

5. Semicolon, coma or parenthesis missing

Error: Unexpected ... expecting ...

[Error] expected ';' before 'return'

This error states that you have forgotten the specified symbol. Do not forget that this error is actually exist in somewhere before the error. Therefore, check the previous statements and even previous lines.

6. Unassigned variable

DevCpp clears uninitialized integers at the declaration statements (makes them zero), and it does not detect using an unassigned variable in expressions.

7. Address Error

DevCpp does not detect and point out addressing errors at scanf and printf functions. The user shall

be very careful not place “&” (address of) operator correctly.

Experiments

Edit and execute the following codes and perform the given tasks:

Task 1:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
      printf("Hello\n");
      printf("My name is Ali\n");
      system("pause");
      return 0;
}
```

How can you modify the code produce the following output on the computer monitor:

```
      Hello ”My name is Ali”
```

Task 2: The following code reads two integer numbers (x, y) from the keyboard and finds their sum, i.e., x+y.

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
      int x,y;
      printf("Enter two Numbers:");
      scanf("%d%d", &x,&y);
      printf("%d + %d =%d \n",x + y);
      system("pause");
      return 0;
}
```

a) Execute this code using the following input entered from the keyboard: 5 12

b) How can you modify this code to read three integer numbers (x, y, z) and compute their product, i.e., x*y*z. Use the following input for execution: 5 12 2