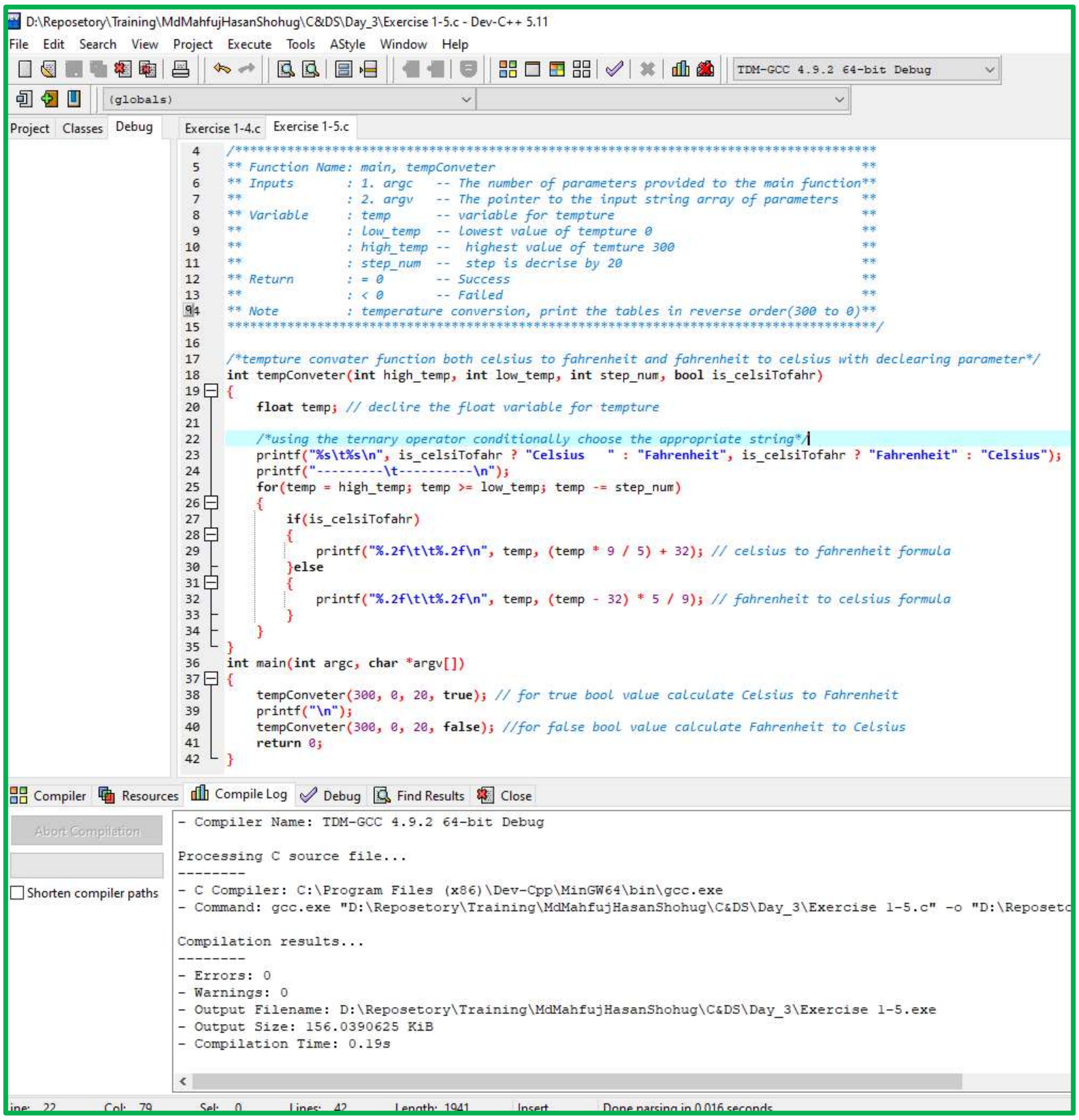
Documentation of Chapter 1

Md Mahfuj Hasan Shohug

BDCOM0019

1. Exercise 1-5:

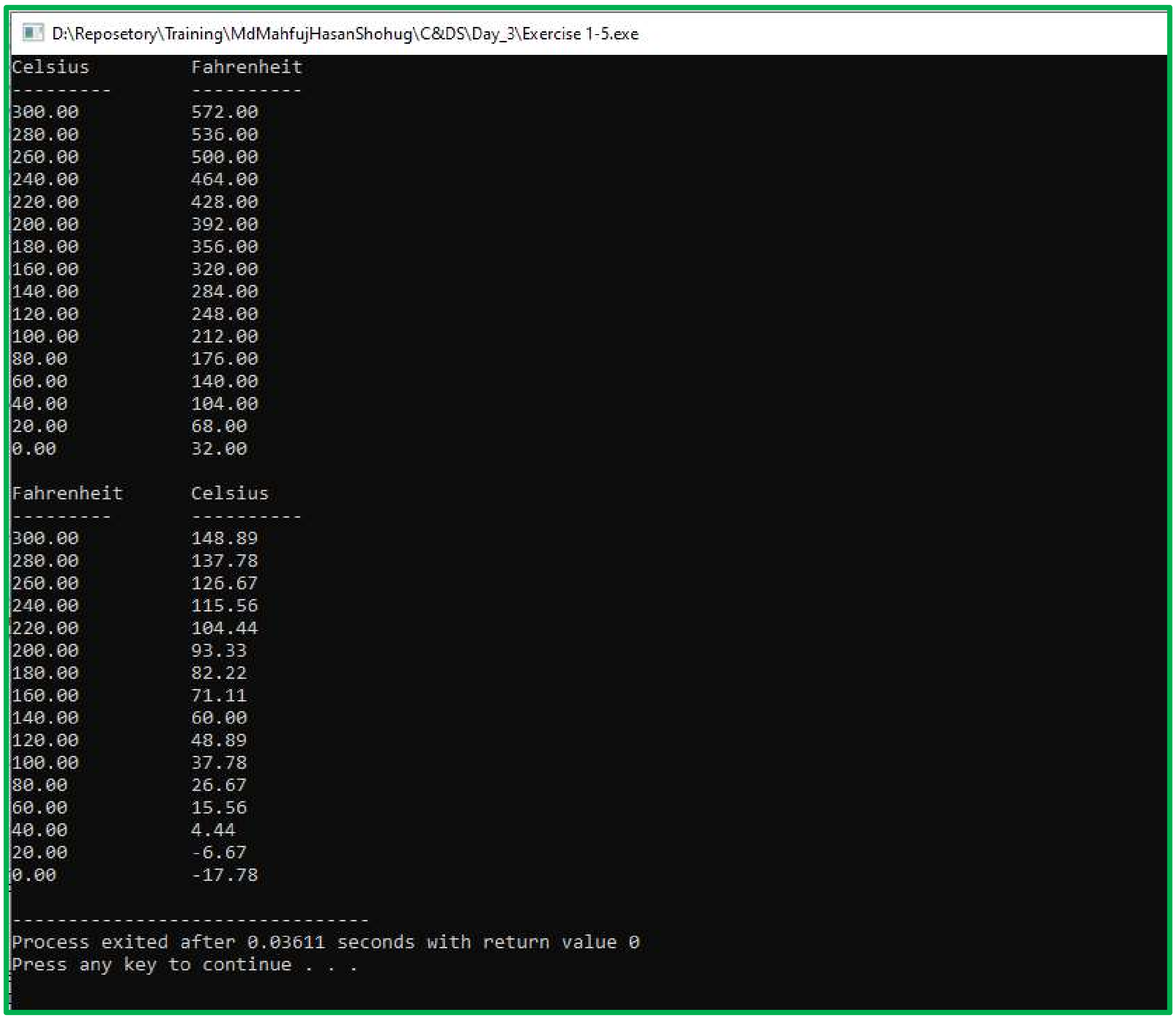
Modify the temperature conversion program to print the table in reverse order, that is, from 300 degrees to 0.
Code file:


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-5.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Debug
(globals)
Project Classes Debug
Exercise 1-4.c Exercise 1-5.c
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 22 Col: 79 Sel: 0 Lines: 42 Length: 1941 Insert Done parsing in 0.016 seconds

```c
/****************************************************************************
** Function Name: main, tempConveter                                       **
** Inputs        : 1. argc   -- The number of parameters provided to the main function**
**               : 2. argv   -- The pointer to the input string array of parameters   **
** Variable      : temp      -- variable for tempture                        **
**               : low_temp  -- lowest value of tempture 0                   **
**               : high_temp --  highest value of temture 300                **
**               : step_num  --  step is decrise by 20                       **
** Return        : = 0       -- Success                                      **
**               : < 0       -- Failed                                       **
** Note          : temperature conversion, print the tables in reverse order(300 to 0)**
*****************************************************************************/

/*tempture convater function both celsius to fahrenheit and fahrenheit to celsius with declearing parameter*/
int tempConveter(int high_temp, int low_temp, int step_num, bool is_celsiTofahr)
{
    float temp; // declire the float variable for tempture

    /*using the ternary operator conditionally choose the appropriate string*/
    printf("%s\t%s\n", is_celsiTofahr ? "Celsius   " : "Fahrenheit", is_celsiTofahr ? "Fahrenheit" : "Celsius");
    printf("---------\t----------\n");
    for(temp = high_temp; temp >= low_temp; temp -= step_num)
    {
        if(is_celsiTofahr)
        {
            printf("%.2f\t\t%.2f\n", temp, (temp * 9 / 5) + 32); // celsius to fahrenheit formula
        }else
        {
            printf("%.2f\t\t%.2f\n", temp, (temp - 32) * 5 / 9); // fahrenheit to celsius formula
        }
    }
}
int main(int argc, char *argv[])
{
    tempConveter(300, 0, 20, true); // for true bool value calculate Celsius to Fahrenheit
    printf("\n");
    tempConveter(300, 0, 20, false); //for false bool value calculate Fahrenheit to Celsius
    return 0;
}
```

```
- Compiler Name: TDM-GCC 4.9.2 64-bit Debug

Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-5.c" -o "D:\Reposeto

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-5.exe
- Output Size: 156.0390625 KiB
- Compilation Time: 0.19s
```

Program Output:



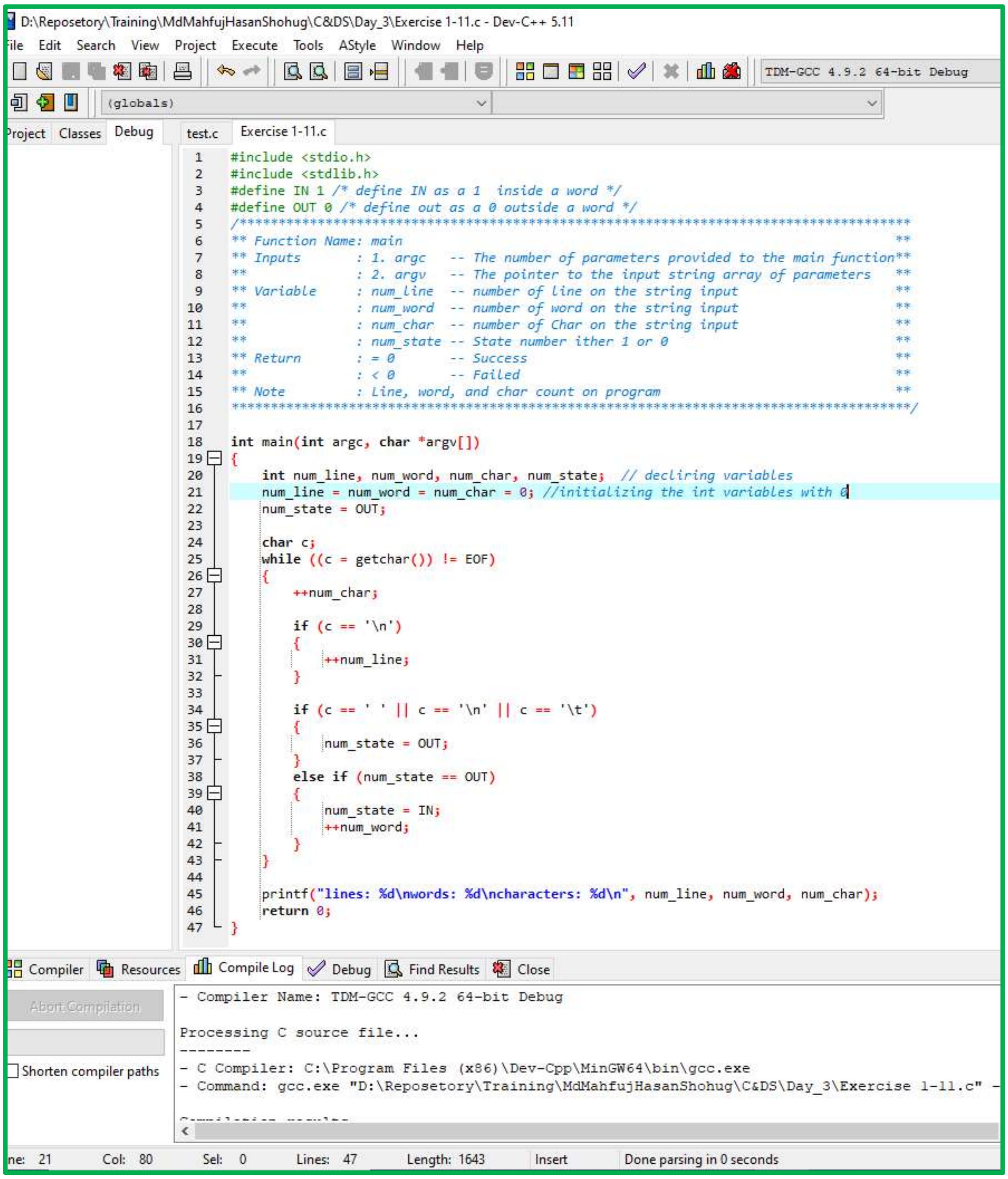
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-5.exe

```
Celsius         Fahrenheit
---------       ----------
300.00          572.00
280.00          536.00
260.00          500.00
240.00          464.00
220.00          428.00
200.00          392.00
180.00          356.00
160.00          320.00
140.00          284.00
120.00          248.00
100.00          212.00
80.00           176.00
60.00           140.00
40.00           104.00
20.00           68.00
0.00            32.00

Fahrenheit      Celsius
---------       ----------
300.00          148.89
280.00          137.78
260.00          126.67
240.00          115.56
220.00          104.44
200.00          93.33
180.00          82.22
160.00          71.11
140.00          60.00
120.00          48.89
100.00          37.78
80.00           26.67
60.00           15.56
40.00           4.44
20.00           -6.67
0.00            -17.78

--------------------------------
Process exited after 0.03611 seconds with return value 0
Press any key to continue . . .
```

Here in this problem I am printing the both Celsius to Fahrenheit and also Fahrenheit to Celsius in reverse order from 300 to 0 degree. This conversion will be calculate on my own defined function called tempConveter();
Here I am also used the Ternary operator for conditionally choose the operative string. Like (when the Boolean variable will be true then print the Celsius to Fahrenheit table and when its false then it will be print Fahrenheit to Celsius).

2. Exercise 1-11:

How would I test the word count program and what kinds of input are most likely to uncover bugs if. Here I am defining those 2 question with different types of example: First how would I test the word count on program. Here is the source code for this count on program:

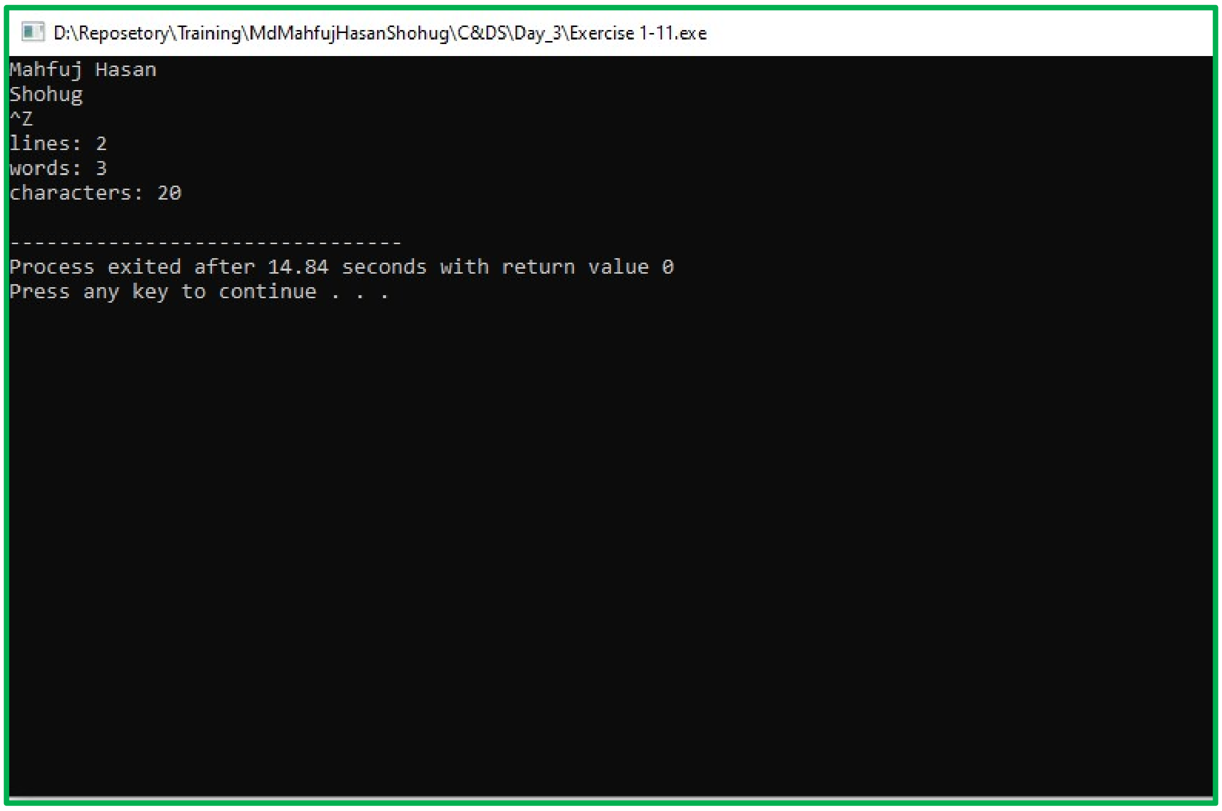

D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Debug
(globals)
Project Classes Debug
test.c Exercise 1-11.c
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

```c
#include <stdio.h>
#include <stdlib.h>
#define IN 1 /* define IN as a 1  inside a word */
#define OUT 0 /* define out as a 0 outside a word */
/*****************************************************************************
** Function Name: main                                                      **
** Inputs        : 1. argc   -- The number of parameters provided to the main function**
**               : 2. argv   -- The pointer to the input string array of parameters   **
** Variable      : num_line  -- number of line on the string input          **
**               : num_word  -- number of word on the string input          **
**               : num_char  -- number of Char on the string input          **
**               : num_state -- State number ither 1 or 0                   **
** Return        : = 0       -- Success                                     **
**               : < 0       -- Failed                                      **
** Note          : Line, word, and char count on program                    **
*****************************************************************************/

int main(int argc, char *argv[])
{
    int num_line, num_word, num_char, num_state;  // decliring variables
    num_line = num_word = num_char = 0; //initializing the int variables with 0
    num_state = OUT;

    char c;
    while ((c = getchar()) != EOF)
    {
        ++num_char;

        if (c == '\n')
        {
            ++num_line;
        }

        if (c == ' ' || c == '\n' || c == '\t')
        {
            num_state = OUT;
        }
        else if (num_state == OUT)
        {
            num_state = IN;
            ++num_word;
        }
    }

    printf("lines: %d\nwords: %d\ncharacters: %d\n", num_line, num_word, num_char);
    return 0;
}
```

```
- Compiler Name: TDM-GCC 4.9.2 64-bit Debug

Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.c" -
```

```
Line: 21    Col: 80    Sel: 0    Lines: 47    Length: 1643    Insert    Done parsing in 0 seconds
```

Write down some testing the word count program with outputs and also analyze those:

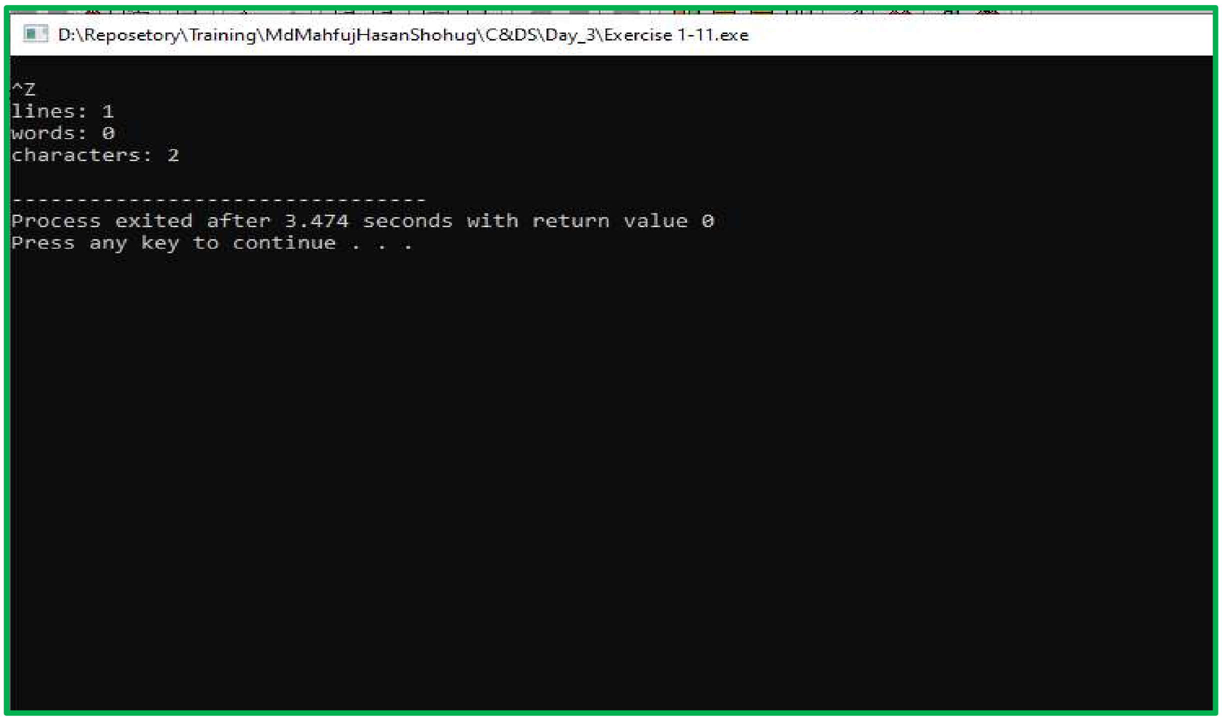


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe

```
Mahfuj Hasan
Shohug
^Z
lines: 2
words: 3
characters: 20

--------------------------------
Process exited after 14.84 seconds with return value 0
Press any key to continue . . .
```

Here I give the input a string “Mahfuj Hasan\nShohug” There are 2 lines and 3 words with 20 characters. That is properly work on Positive Test Cases

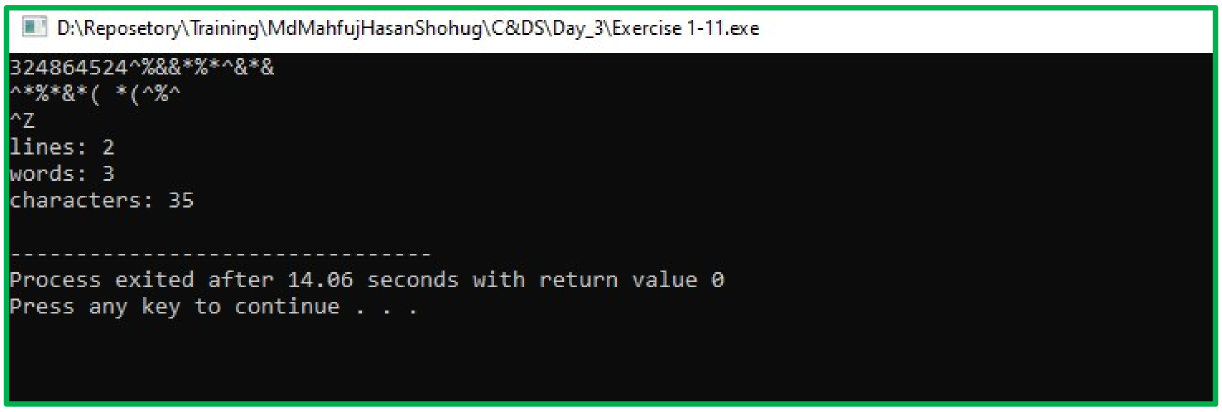
Now show some outputs on this code kinds of input are most likely to uncover bugs:



D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe

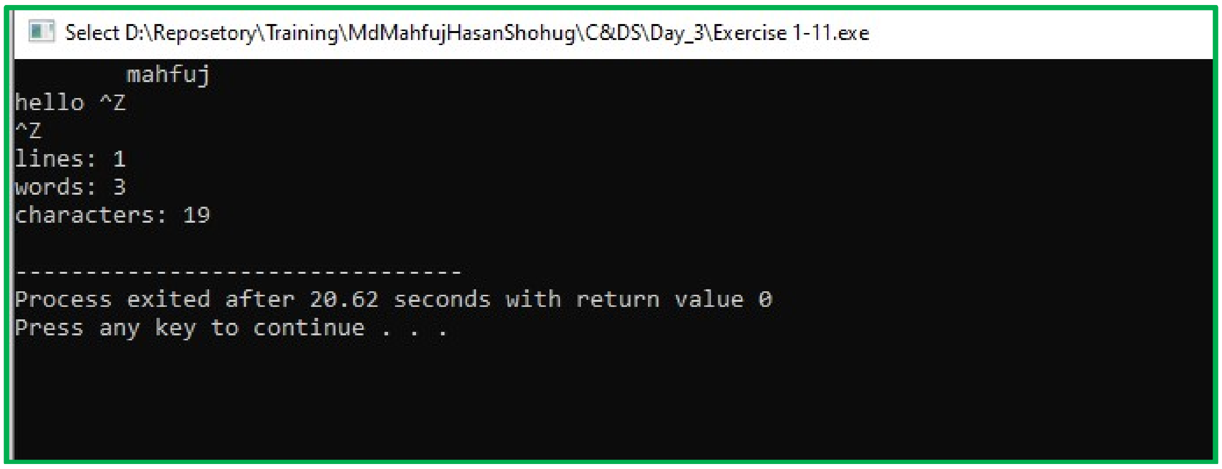
```

^Z
lines: 1
words: 0
characters: 2

--------------------------------
Process exited after 3.474 seconds with return value 0
Press any key to continue . . .
```

Here I am just input a spaces “ ” and enter “\n”, there are no any kind of word but the output shows that line number 1 and word is 0. This has some error to detect.


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe

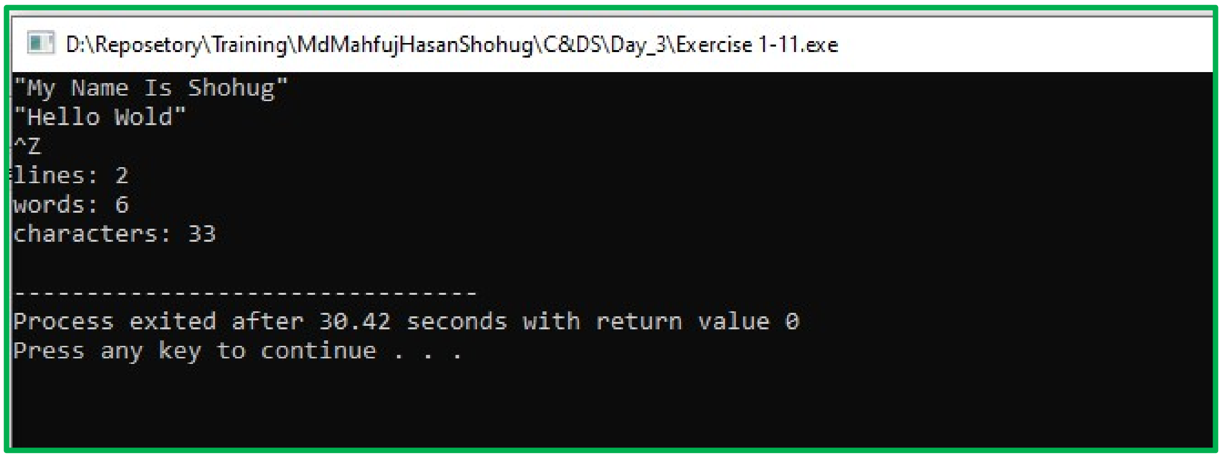
```
324864524^%&&*%*^&*&
^*%*&*( *(^%^
^Z
lines: 2
words: 3
characters: 35

--------------------------------
Process exited after 14.06 seconds with return value 0
Press any key to continue . . .
```

Some special characters also count a word and line this also most likely to uncover bugs


Select D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe

```
        mahfuj
hello ^Z
^Z
lines: 1
words: 3
characters: 19

--------------------------------
Process exited after 20.62 seconds with return value 0
Press any key to continue . . .
```

Starting with tab and entering the some extra tab and spaces the line count is one and the characters is 19 those are also in my opinion have most likely to uncover bugs.


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe

```
"My Name Is Shohug"
"Hello Wold"
^Z
lines: 2
words: 6
characters: 33

--------------------------------
Process exited after 30.42 seconds with return value 0
Press any key to continue . . .
```

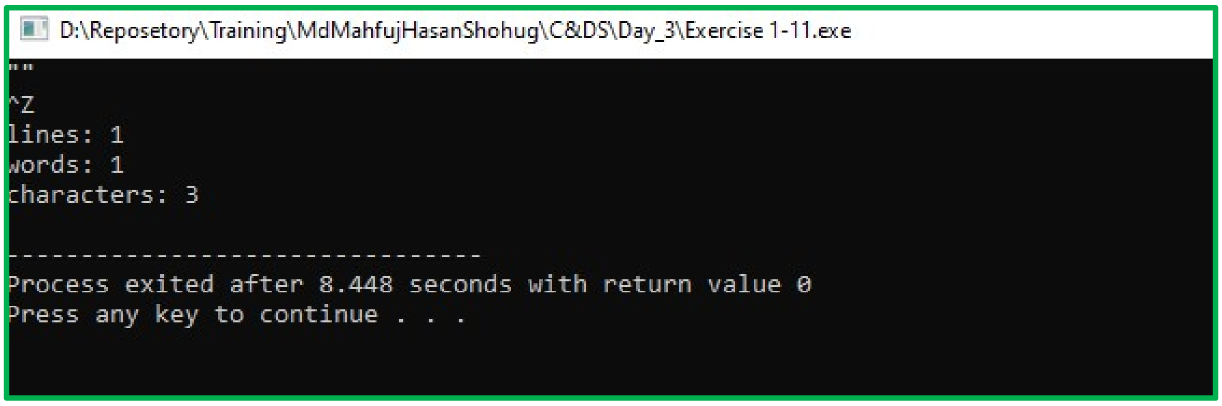
Here with " ", I input some string and the line count, word count and also the characters count is correct. And this is the Positive Test Cases

According my learning, In short description here I mention some test a word count program written in C that can follow these steps:

- Positive Test Cases: Test the program using common and anticipated inputs first. Different input strings with spaces or other whitespace characters between the

words are required.
Example: Input: "Mahfuj Hasan Shohug"
Expected Output: Word count = 3.

- Boundary Cases: Use inputs that are close to the borders or limits of the data types being utilized to test the application. For instance, extremely long input strings, a word limit, or severe edge cases particular to your program's specifications. Make sure the application handles these situations flawlessly and offers precise word counts.
  Example: Input: A very long string with thousands of words.
  Expected Output: Word count = Number of words in the input string.
- Empty Input: Test the program with an empty string as input to ensure it handles this case correctly and returns a word count of zero.
  Example:



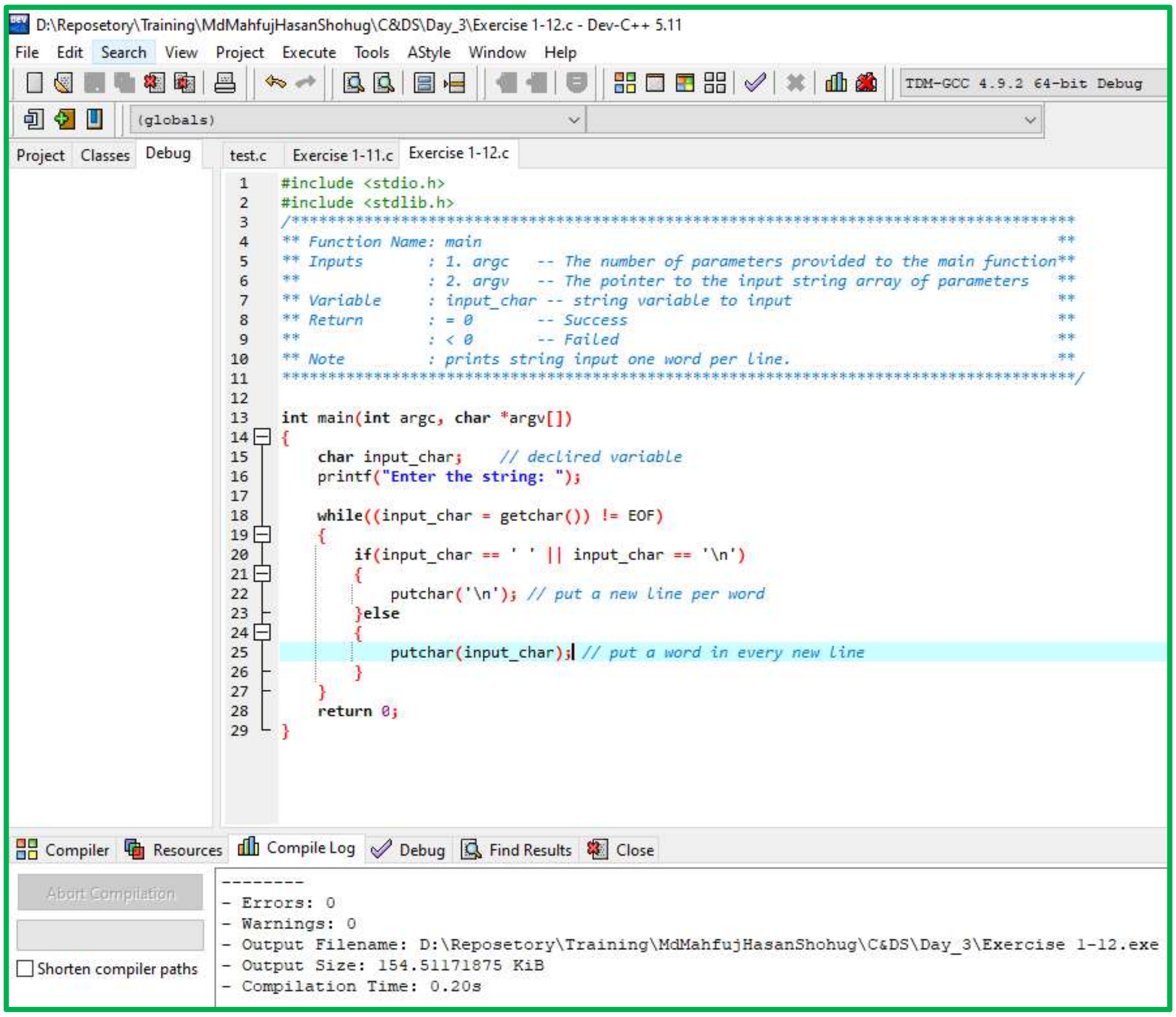
```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-11.exe
""
^Z
lines: 1
words: 1
characters: 3

--------------------------------
Process exited after 8.448 seconds with return value 0
Press any key to continue . . .
```

  Here counting line and word 1 for pressing the enter “\n” but the expected output should be 0;
- User Acceptance Testing (UAT): Participate stakeholders or end users in UAT. They can evaluate the app's usability, functionality, and general happiness by running it through real-world scenarios.
- Error Handling: Test the program's ability to handle different error situations, such as incorrect inputs or extreme circumstances. Make that the application displays the proper error messages and responds to these circumstances in a polite manner.

Also there are some other most likely to uncover bugs system have in this string word count program. It also keeps in mind to record test cases, anticipated outcomes, and any problems or defects that have been found. Evaluate and revise my tests frequently as the application changes or as new requirements appear.
I can make sure that my C word count software runs well, handles a variety of input conditions, and calculates precise word counts by using the testing strategies described above.

3. Exercise 1-12:

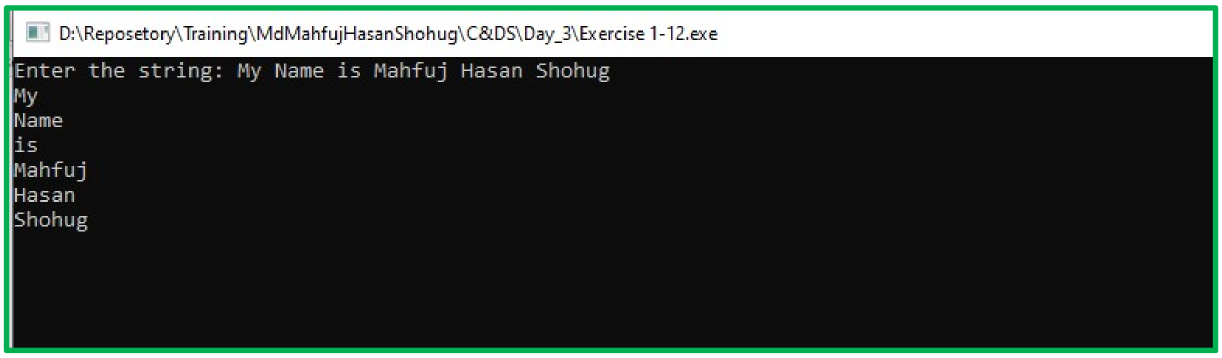
Write a program that prints its input one word per line.
Source Code file:


D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-12.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Debug
(globals)
Project Classes Debug
test.c Exercise 1-11.c Exercise 1-12.c
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

```c
#include <stdio.h>
#include <stdlib.h>
/******************************************************************************
** Function Name: main                                                       **
** Inputs        : 1. argc   -- The number of parameters provided to the main function**
**               : 2. argv   -- The pointer to the input string array of parameters   **
** Variable      : input_char -- string variable to input                    **
** Return        : = 0        -- Success                                     **
**               : < 0        -- Failed                                      **
** Note          : prints string input one word per line.                    **
*******************************************************************************/

int main(int argc, char *argv[])
{
	char input_char;	// declired variable
	printf("Enter the string: ");
	
	while((input_char = getchar()) != EOF)
	{
		if(input_char == ' ' || input_char == '\n')
		{
			putchar('\n'); // put a new line per word
		}else
		{
			putchar(input_char); // put a word in every new line
		}
	}
	return 0;
}
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_3\Exercise 1-12.exe
- Output Size: 154.51171875 KiB
- Compilation Time: 0.20s
```

Output:



```
Enter the string: My Name is Mahfuj Hasan Shohug
My
Name
is
Mahfuj
Hasan
Shohug
```

In this problem here I am entering “My name Mahfuj Hasan Shohug”. And the program make those all word in one individual line. That’s means prints its input one word per line.