Day 15 Documentation(I spend 5 hour for read book and solved problem 3 hour)

Md. Mahfuj Hasan Shohug

BDCOM0019

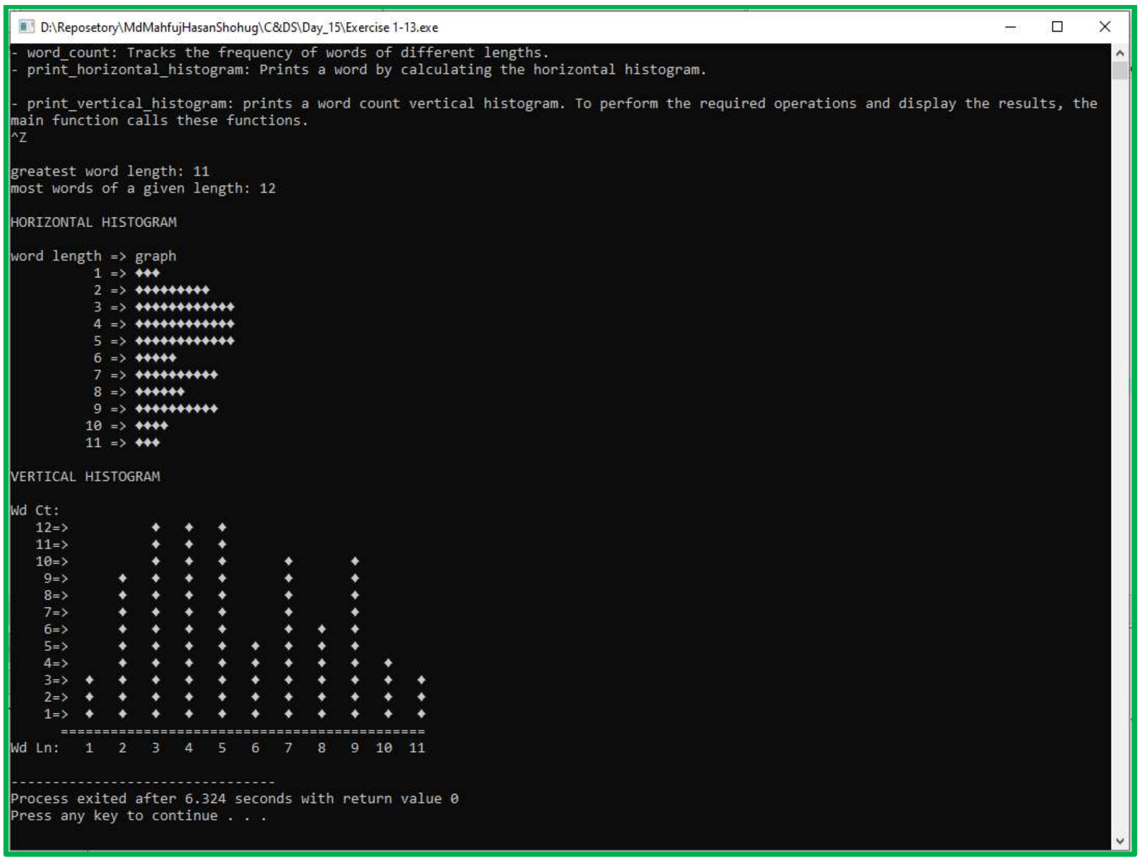
1. **Exercise 1-13:**

   Problem: Write a program to print a histogram of the lengths of words in its input. It is easy to draw the histogram with the bars horizontal; a vertical orientation is more challenging.

   Analysis:

   - This program counts occurrences of words of various lengths while reading user-supplied input text.
   - The word count for each word length is then represented by a horizontal and vertical histogram, which is generated and printed.
   - - word_count: Tracks the frequency of words of different lengths.
   - - print_horizontal_histogram: Prints a word by calculating the horizontal histogram.
   - - print_vertical_histogram: prints a word count vertical histogram. The required operations and display the results "\4", the main function calls these functions.

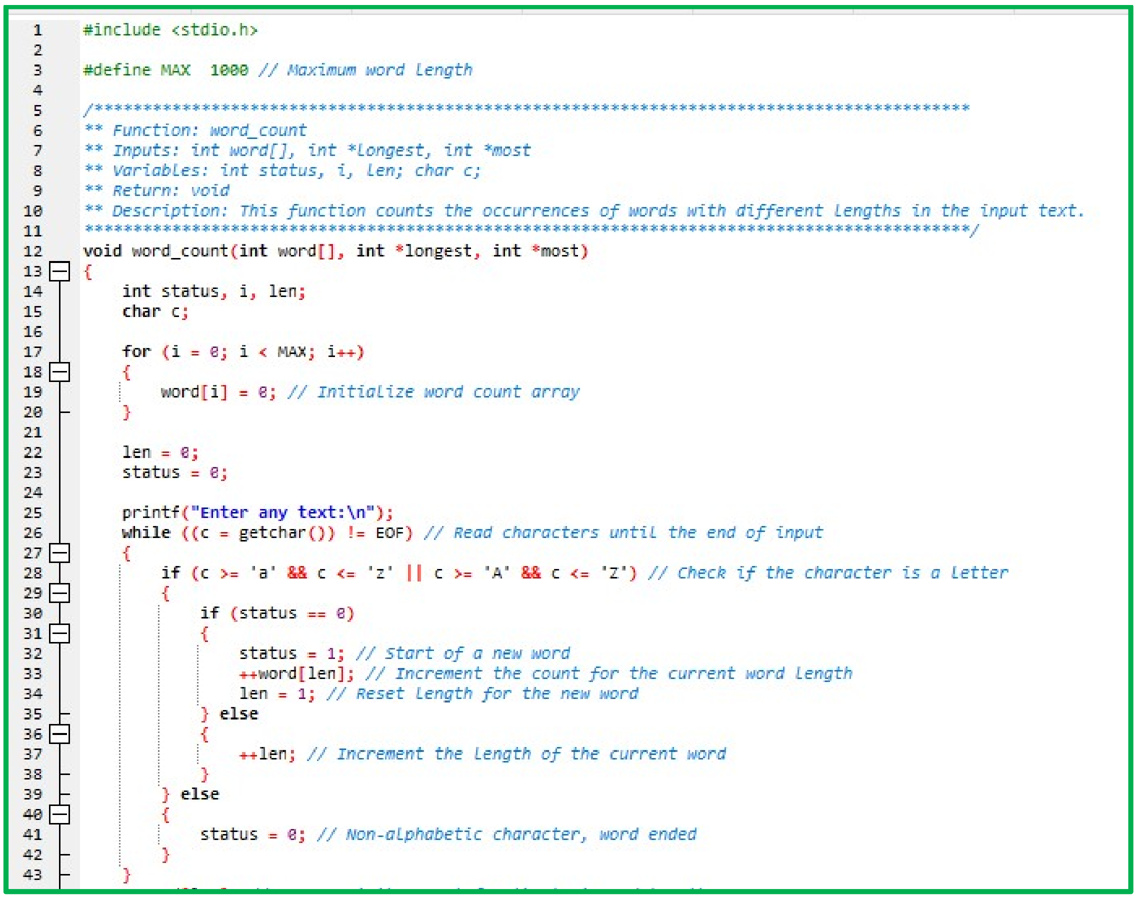
   Outputs:



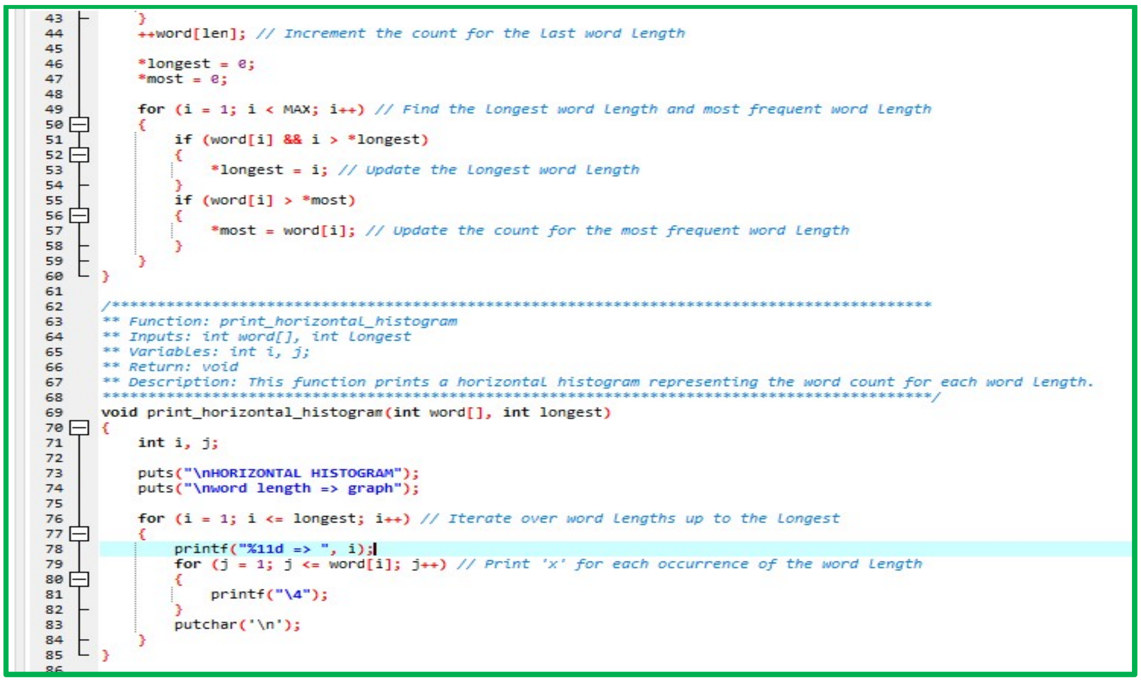
D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_15\Exercise 1-13.exe

```
- word_count: Tracks the frequency of words of different lengths.
- print_horizontal_histogram: Prints a word by calculating the horizontal histogram.

- print_vertical_histogram: prints a word count vertical histogram. To perform the required operations and display the results, the
main function calls these functions.
^Z

greatest word length: 11
most words of a given length: 12

HORIZONTAL HISTOGRAM

word length => graph
          1 => ◆◆◆
          2 => ◆◆◆◆◆◆◆◆◆
          3 => ◆◆◆◆◆◆◆◆◆◆◆◆
          4 => ◆◆◆◆◆◆◆◆◆◆◆◆
          5 => ◆◆◆◆◆◆◆◆◆◆◆◆
          6 => ◆◆◆◆◆
          7 => ◆◆◆◆◆◆◆◆◆◆
          8 => ◆◆◆◆◆◆
          9 => ◆◆◆◆◆◆◆◆◆◆
         10 => ◆◆◆◆
         11 => ◆◆◆

VERTICAL HISTOGRAM

Wd Ct:
   12=>          ◆   ◆   ◆
   11=>          ◆   ◆   ◆
   10=>          ◆   ◆   ◆       ◆       ◆
    9=>      ◆   ◆   ◆   ◆       ◆       ◆
    8=>      ◆   ◆   ◆   ◆       ◆       ◆
    7=>      ◆   ◆   ◆   ◆       ◆       ◆
    6=>      ◆   ◆   ◆   ◆       ◆   ◆   ◆
    5=>      ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆
    4=>      ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆
    3=>  ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆
    2=>  ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆
    1=>  ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆   ◆
      ============================================
Wd Ln:   1   2   3   4   5   6   7   8   9  10  11

--------------------------------
Process exited after 6.324 seconds with return value 0
Press any key to continue . . .
```

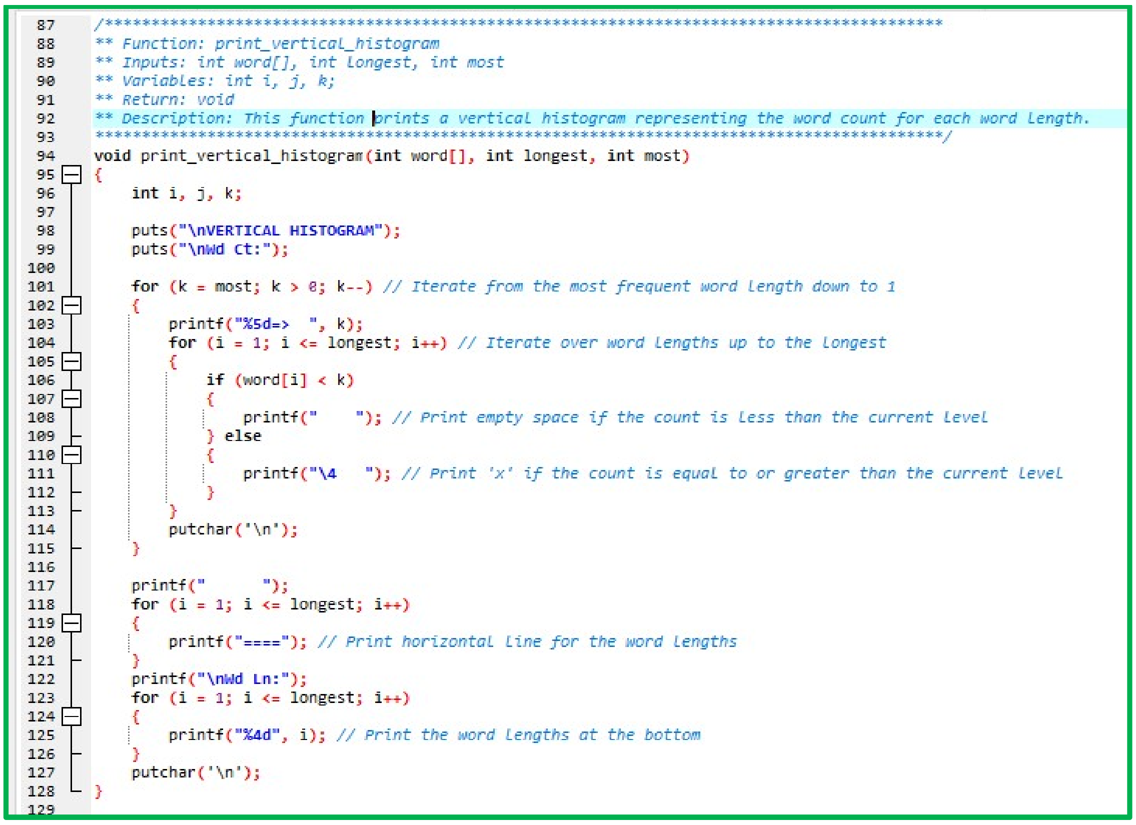
Source Code:



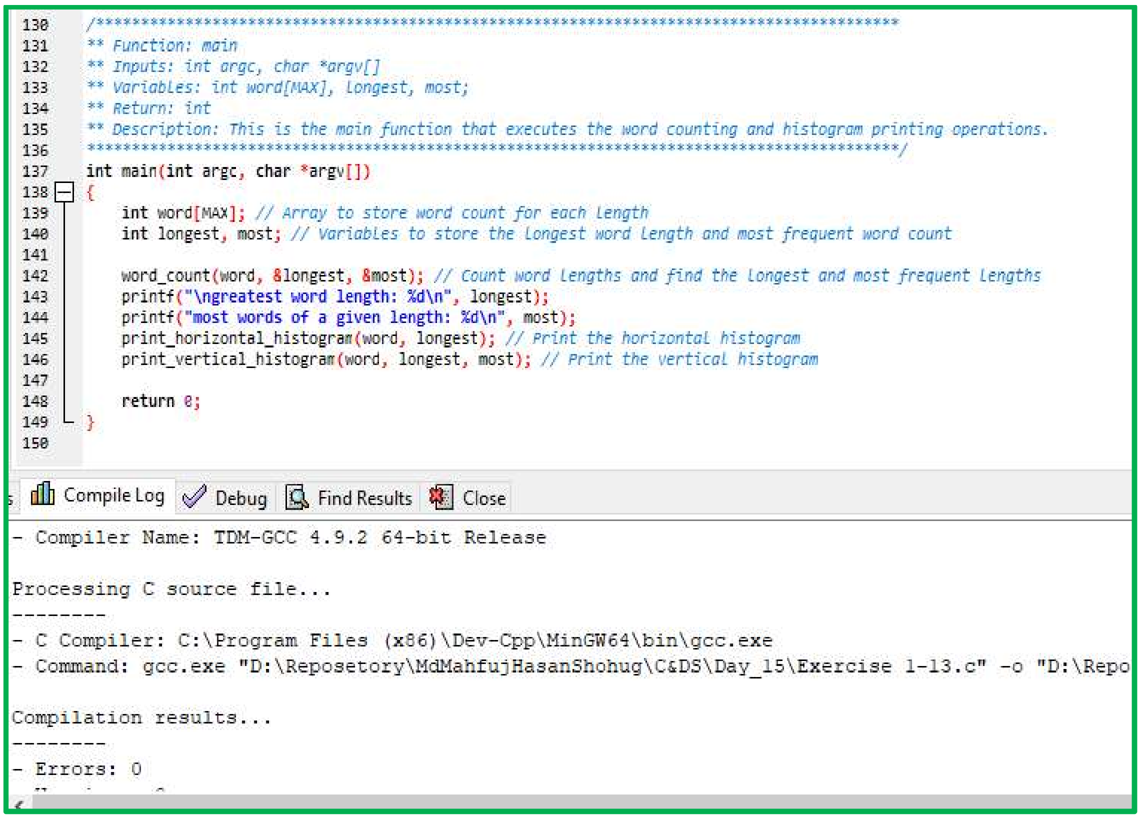
```c
#include <stdio.h>

#define MAX  1000 // Maximum word Length

/****************************************************************************************
** Function: word_count
** Inputs: int word[], int *Longest, int *most
** Variables: int status, i, Len; char c;
** Return: void
** Description: This function counts the occurrences of words with different Lengths in the input text.
****************************************************************************************/
void word_count(int word[], int *longest, int *most)
{
    int status, i, len;
    char c;

    for (i = 0; i < MAX; i++)
    {
        word[i] = 0; // Initialize word count array
    }

    len = 0;
    status = 0;

    printf("Enter any text:\n");
    while ((c = getchar()) != EOF) // Read characters until the end of input
    {
        if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') // Check if the character is a Letter
        {
            if (status == 0)
            {
                status = 1; // Start of a new word
                ++word[len]; // Increment the count for the current word Length
                len = 1; // Reset Length for the new word
            } else
            {
                ++len; // Increment the Length of the current word
            }
        } else
        {
            status = 0; // Non-alphabetic character, word ended
        }
    }
```



```c
    ++word[len]; // Increment the count for the Last word Length

    *longest = 0;
    *most = 0;

    for (i = 1; i < MAX; i++) // Find the Longest word Length and most frequent word Length
    {
        if (word[i] && i > *longest)
        {
            *longest = i; // Update the Longest word Length
        }
        if (word[i] > *most)
        {
            *most = word[i]; // Update the count for the most frequent word Length
        }
    }
}

/****************************************************************************************
** Function: print_horizontal_histogram
** Inputs: int word[], int Longest
** Variables: int i, j;
** Return: void
** Description: This function prints a horizontal histogram representing the word count for each word Length.
****************************************************************************************/
void print_horizontal_histogram(int word[], int longest)
{
    int i, j;

    puts("\nHORIZONTAL HISTOGRAM");
    puts("\nword length => graph");

    for (i = 1; i <= longest; i++) // Iterate over word Lengths up to the Longest
    {
        printf("%11d => ", i);
        for (j = 1; j <= word[i]; j++) // Print 'x' for each occurrence of the word Length
        {
            printf("\4");
        }
        putchar('\n');
    }
}
```



```c
/*************************************************************************************
** Function: print_vertical_histogram
** Inputs: int word[], int longest, int most
** Variables: int i, j, k;
** Return: void
** Description: This function prints a vertical histogram representing the word count for each word length.
*************************************************************************************/
void print_vertical_histogram(int word[], int longest, int most)
{
    int i, j, k;

    puts("\nVERTICAL HISTOGRAM");
    puts("\nWd Ct:");

    for (k = most; k > 0; k--) // Iterate from the most frequent word length down to 1
    {
        printf("%5d=>  ", k);
        for (i = 1; i <= longest; i++) // Iterate over word lengths up to the longest
        {
            if (word[i] < k)
            {
                printf("    "); // Print empty space if the count is less than the current level
            } else
            {
                printf("\4   "); // Print 'x' if the count is equal to or greater than the current level
            }
        }
        putchar('\n');
    }

    printf("       ");
    for (i = 1; i <= longest; i++)
    {
        printf("===="); // Print horizontal line for the word lengths
    }
    printf("\nWd Ln:");
    for (i = 1; i <= longest; i++)
    {
        printf("%4d", i); // Print the word lengths at the bottom
    }
    putchar('\n');
}
```



```c
/*************************************************************************************
** Function: main
** Inputs: int argc, char *argv[]
** Variables: int word[MAX], longest, most;
** Return: int
** Description: This is the main function that executes the word counting and histogram printing operations.
*************************************************************************************/
int main(int argc, char *argv[])
{
    int word[MAX]; // Array to store word count for each length
    int longest, most; // Variables to store the longest word length and most frequent word count

    word_count(word, &longest, &most); // Count word lengths and find the longest and most frequent lengths
    printf("\ngreatest word length: %d\n", longest);
    printf("most words of a given length: %d\n", most);
    print_horizontal_histogram(word, longest); // Print the horizontal histogram
    print_vertical_histogram(word, longest, most); // Print the vertical histogram

    return 0;
}
```

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

Processing C source file...
--------
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_15\Exercise 1-13.c" -o "D:\Repo

Compilation results...
--------
- Errors: 0
```

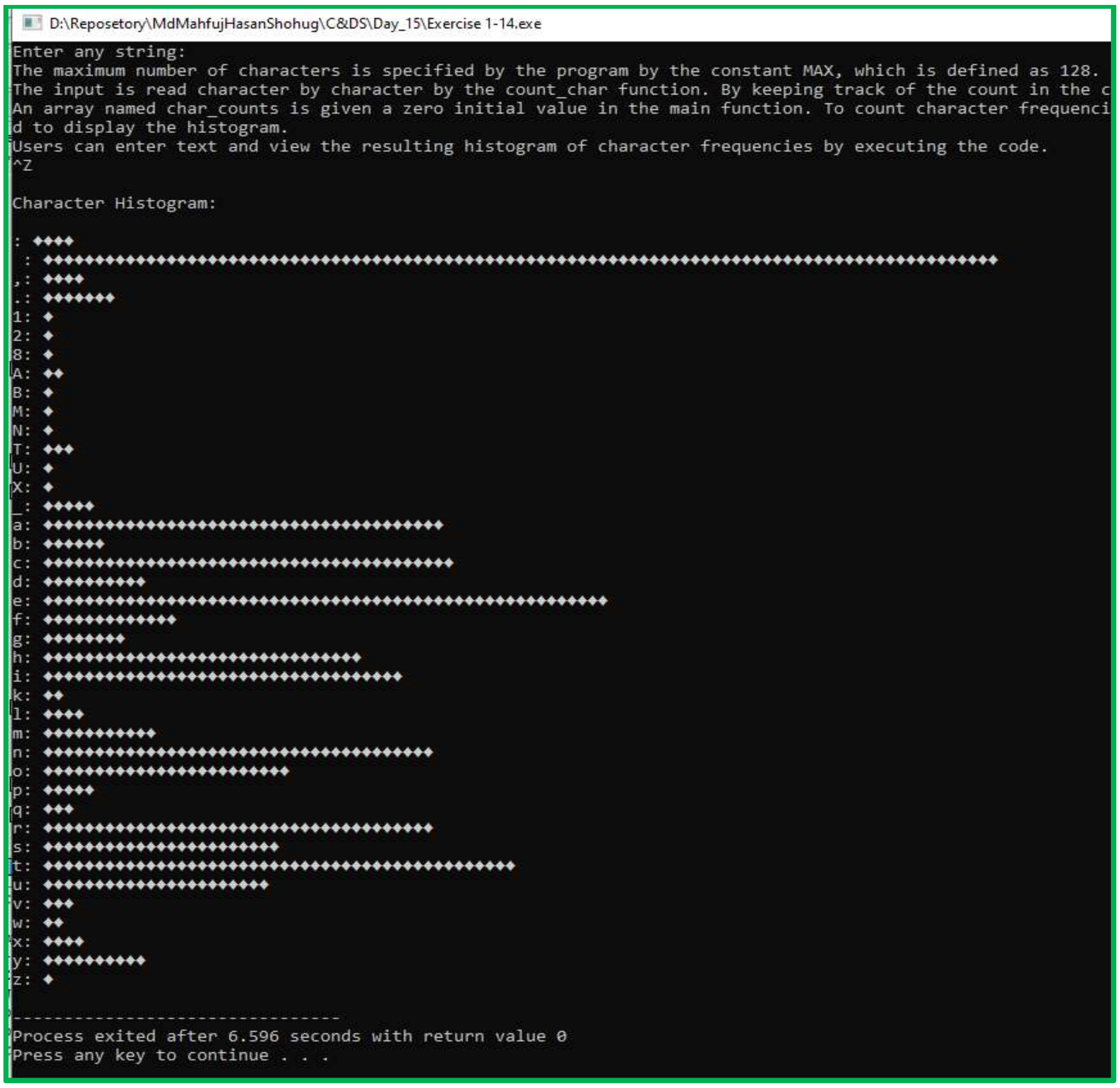
2. Exercise 1-14:

Problem: Write a program to print a histogram of the frequencies of different characters In its input.

Analysis:

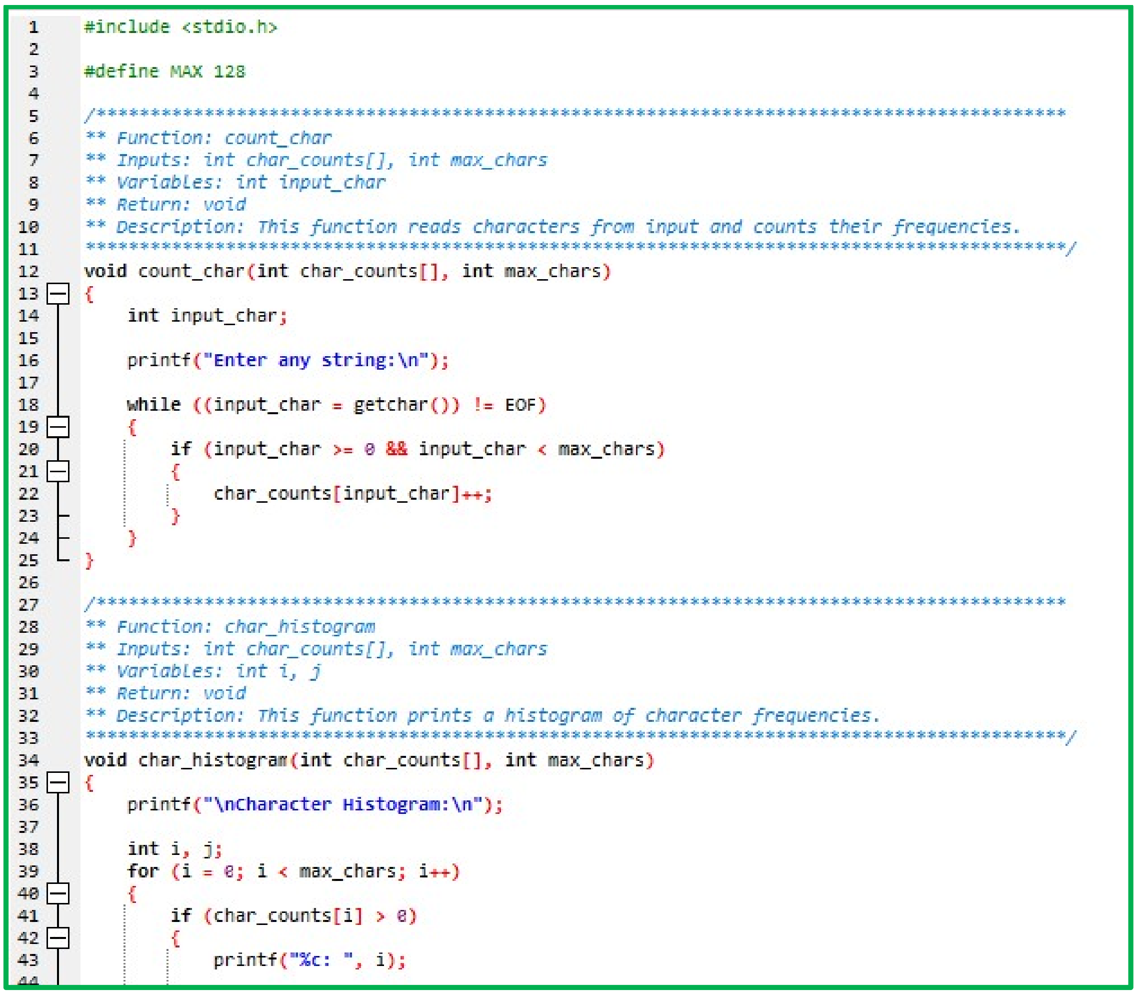
- The maximum number of characters is specified by the program by the constant MAX, which is defined as 128.
- The input is read character by character by the count_char function. By keeping track of the count in the char_count array, it increments the frequency count for each character.
- An array named char_counts is given a zero initial value in the main function. To count character frequencies, the count_char function is used. Next, the char_histogram function is used to display the histogram.

Output:

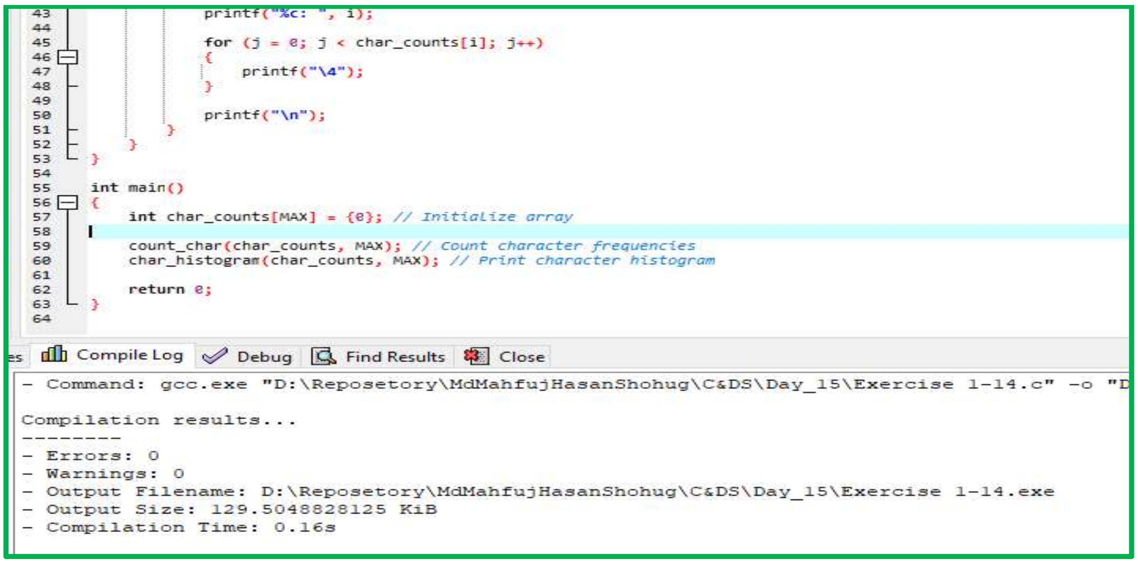


```
D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_15\Exercise 1-14.exe
Enter any string:
The maximum number of characters is specified by the program by the constant MAX, which is defined as 128.
The input is read character by character by the count_char function. By keeping track of the count in the c
An array named char_counts is given a zero initial value in the main function. To count character frequenci
d to display the histogram.
Users can enter text and view the resulting histogram of character frequencies by executing the code.
^Z

Character Histogram:

: ♦♦♦♦
 : ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
,: ♦♦♦♦
.: ♦♦♦♦♦♦♦
1: ♦
2: ♦
8: ♦
A: ♦♦
B: ♦
M: ♦
N: ♦
T: ♦♦♦
U: ♦
X: ♦
_: ♦♦♦♦♦
a: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
b: ♦♦♦♦♦♦
c: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
d: ♦♦♦♦♦♦♦♦♦♦
e: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
f: ♦♦♦♦♦♦♦♦♦♦♦♦♦
g: ♦♦♦♦♦♦♦♦
h: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
i: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
k: ♦♦
l: ♦♦♦♦
m: ♦♦♦♦♦♦♦♦♦♦♦
n: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
o: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
p: ♦♦♦♦♦
q: ♦♦♦
r: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
s: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
t: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
u: ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
v: ♦♦♦
w: ♦♦
x: ♦♦♦♦
y: ♦♦♦♦♦♦♦♦♦♦
z: ♦


--------------------------------
Process exited after 6.596 seconds with return value 0
Press any key to continue . . .
```

Source Code:



```c
#include <stdio.h>

#define MAX 128

/*******************************************************************************************
** Function: count_char
** Inputs: int char_counts[], int max_chars
** Variables: int input_char
** Return: void
** Description: This function reads characters from input and counts their frequencies.
*******************************************************************************************/
void count_char(int char_counts[], int max_chars)
{
    int input_char;

    printf("Enter any string:\n");

    while ((input_char = getchar()) != EOF)
    {
        if (input_char >= 0 && input_char < max_chars)
        {
            char_counts[input_char]++;
        }
    }
}

/*******************************************************************************************
** Function: char_histogram
** Inputs: int char_counts[], int max_chars
** Variables: int i, j
** Return: void
** Description: This function prints a histogram of character frequencies.
*******************************************************************************************/
void char_histogram(int char_counts[], int max_chars)
{
    printf("\nCharacter Histogram:\n");

    int i, j;
    for (i = 0; i < max_chars; i++)
    {
        if (char_counts[i] > 0)
        {
            printf("%c: ", i);
```



```c
            printf("%c: ", i);

            for (j = 0; j < char_counts[i]; j++)
            {
                printf("\4");
            }

            printf("\n");
        }
    }
}

int main()
{
    int char_counts[MAX] = {0}; // Initialize array

    count_char(char_counts, MAX); // Count character frequencies
    char_histogram(char_counts, MAX); // Print character histogram

    return 0;
}
```

es | Compile Log | Debug | Find Results | Close

```
- Command: gcc.exe "D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_15\Exercise 1-14.c" -o "D

Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\MdMahfujHasanShohug\C&DS\Day_15\Exercise 1-14.exe
- Output Size: 129.5048828125 KiB
- Compilation Time: 0.16s
```