Day 9: Count Characters in a String

Gati Goyal

"The art of programming is the art of organizing complexity, mastering simplicity, and creating efficiency."
— Edsger W. Dijkstra

---

1 Introduction

Counting characters in a string is a fundamental problem that involves analyzing and categorizing the content of a given string. This task strengthens the understanding of loops, conditionals, and character classification functions in C.

2 Problem Statement

Problem: Count the number of vowels, consonants, digits, and special characters in a given string. **Hint:** Use character classification functions like `isalpha()` and `isdigit()` for efficient categorization. **Edge Case:** Handle empty strings and strings with only special characters.

3 Algorithm

3.1 Steps to Solve the Problem

1. Initialize counters for vowels, consonants, digits, and special characters to 0.
2. Traverse each character of the string:
   - Use `isalpha()` to check if the character is an alphabet.
   - Further classify alphabets as vowels or consonants.
   - Use `isdigit()` to check if the character is a digit.
   - Count any other character as a special character.
3. Display the counts at the end of traversal.

4 Code

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void countCharacters(char str[]) {
    int vowels = 0, consonants = 0, digits = 0, specialChars = 0;
    for (int i = 0; str[i] != '\0'; i++) {
        char ch = str[i];
        if (isalpha(ch)) { // Check if it's an alphabet
            ch = tolower(ch); // Convert to lowercase for uniformity
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
                vowels++;
            } else {
                consonants++;
            }
        } else if (isdigit(ch)) { // Check if it's a digit
            digits++;
        } else { // Anything else is a special character
            specialChars++;
        }
    }
    printf("Vowels: %d\n", vowels);
    printf("Consonants: %d\n", consonants);
    printf("Digits: %d\n", digits);
    printf("Special Characters: %d\n", specialChars);
}
int main() {
    char input[100];

    printf("Enter a string: ");
    scanf("%99[^\n]", input); // Read the input string, avoiding overflow

    if (strlen(input) == 0) {
        printf("The input string is empty.\n");
    } else {
        countCharacters(input);
    }
    return 0;
}
```

5 Step-by-Step Explanation

1. **Initialize Counters:** Set counters for each category (vowels, consonants, digits, special characters) to zero.

2. **Iterate Through the String:**

- Use `tolower()` to simplify character classification.
- Check for alphabets using `isalpha()` and classify them as vowels or consonants.
- Use `isdigit()` to identify numerical digits.
- Any character not meeting the above criteria is counted as a special character.

3. **Output Results:** Print the counts of each category after processing the entire string.

6 Complexity Analysis

- **Time Complexity:** $O(n)$ Traversal of the string takes linear time with respect to its length.
- **Space Complexity:** $O(1)$ Only a fixed amount of memory is used for counters, regardless of string size.

7 Examples and Edge Cases

Input String	Vowels	Consonants	Digits	Special Characters
"Hello, World! 123"	3	7	3	4
"AEIOUaeiou"	10	0	0	0
"12345"	0	0	5	0

8 Output

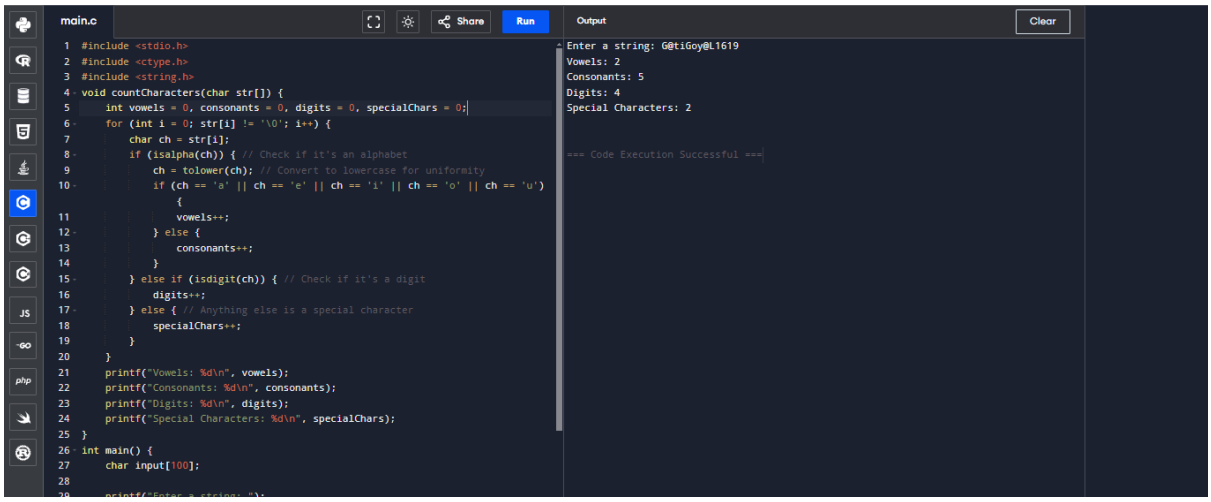

Figure 1: Output in an online compiler

9 Conclusion

This program efficiently counts vowels, consonants, digits, and special characters in a given string. It uses character classification functions, ensuring readability and maintainability. The time complexity of $O(n)$ makes it suitable for large input strings.