Activity 8:Counting Vowels

Name:Krushna Bhausaheb Kokane
Class:CE(2)
Roll no:65

Research:

Strings are one of the most important data types in computer programming, especially in languages like C where a string is stored as an array of characters. In text-based applications such as word processing, search engines, and language translation, analyzing characters in a string is a common and essential operation.

Vowels (a, e, i, o, u) play a major role in constructing meaningful words in the English language. They affect pronunciation, phonetics, and linguistic structure. Therefore, detecting vowels in text is useful in various real-world applications including speech recognition, text-to-speech systems, and language learning tools.

In C programming, characters are stored in memory using their ASCII codes. Every string ends with a null terminator `'\0'`, which helps the program identify where the string finishes. By iterating through each character of the string until the null character is reached, we can analyze and categorize characters such as vowels, consonants, digits, and special symbols.

Ideate:

To solve the problem of identifying vowels in a given string, different approaches can be considered. First, we explore how characters are stored and processed in C. Since each string contains a sequence of characters, we can scan each one and classify whether it is a vowel or not. Various logical methods can be used such as `if-else` conditions, `switch` statements, or ASCII conversions.

Using a separate function to perform vowel checking makes the program modular and easier to maintain. The idea is to allow the user to enter any

name or sentence and efficiently count vowels, including uppercase and lowercase characters

Analysis:

When a user enters a string, each character is stored in a continuous block of memory using a character array. The program then processes the string one character at a time until it reaches the null terminator `'\0'`. Each character is compared with a set of predefined vowels. If a match is found, the vowel count is increased.

By using a loop and a function, the program achieves efficient execution with low time complexity (O(n), where n is the number of characters in the string). Memory usage is minimal since only a few variables are used apart from the string input.

Build:

```
#include <stdio.h>

int main() {
    char name[100];
    int i, count = 0;

    printf("Enter a string: ");
    scanf(" %[^\n]", name);

    for(i = 0; name[i] != '\0'; i++) {
        if(name[i]=='a' || name[i]=='e' || name[i]=='i' || name[i]=='o' || name[i]=='u'
||
           name[i]=='A' || name[i]=='E' || name[i]=='I' || name[i]=='O' ||
name[i]=='U') {
            count++;
        }
```

```
    }

    printf("Number of vowels in string = %d\n", count);

    return 0;
}
```

Testing:

```
Enter a string: my college name is mmcoe
Number of vowels in string = 8
```

Implementation: