MIT-WPU

MIT-WORLD PEACE UNIVERSITY

F. Y. B. Tech

Trimester: I/II/III Subject: Programming and Problem Solving

Name: Krishnaraj Thadesar **Division:** 9

Roll No.: 109054 **Batch:** I3

Experiment No.: 5

Name of the Experiment: Write a Menu driven C program to perform all String operations using User Defined functions.

Performed on: 10rd February 2022

Submitted on: 15th February 2022

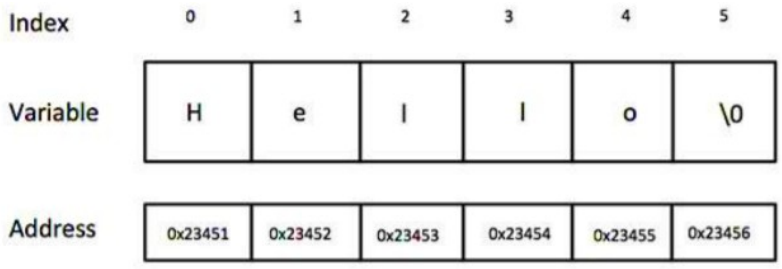
AIM: Write a Menu driven C program to perform all String operations using User Defined functions.

OBJECTIVE:

To learn string Operations in C.

THEORY:

Strings are actually one-dimensional array of characters terminated by a null character '\0'



String Library Functions:

strcpy(s1, s2);

Copies string s2 into string s1.

strcat(s1, s2);

Concatenates string s2 onto the end of string s1.

strlen(s1);

Returns the length of string s1.

strcmp(s1, s2);

Returns 0 if s1 and s2 are the same; less than 0 if s1s2.

PLATFORM: *64-Bit ArchLinux x86 with gdb/g++ compiler.*

ALGORITHM:

Step 1: Start

Step 1: Start

Step 2: Declare/create user defined string functions to compute the **length** of a string, strlength(), to **concatenate** two strings, strconcatenate(), to **compare** two strings, strcompare() and to **copy** a string, strcopy()

Step 3: Declare two arrays of type character, str1[ ], str2[ ] and variable c

Step 4: Read strings, str1[ ], str2[ ],c

Step 5: Declare switch case statement

switch(c)

case 1: Compute the length of a string with user defined function strlength()

break;

case 2: Concatenate a two strings with user defined function strconcatenate()

break;

case 3: Compare a two strings with user defined function strcompare()

break;

case 4: Copy a one string from another string with user defined function strcopy()

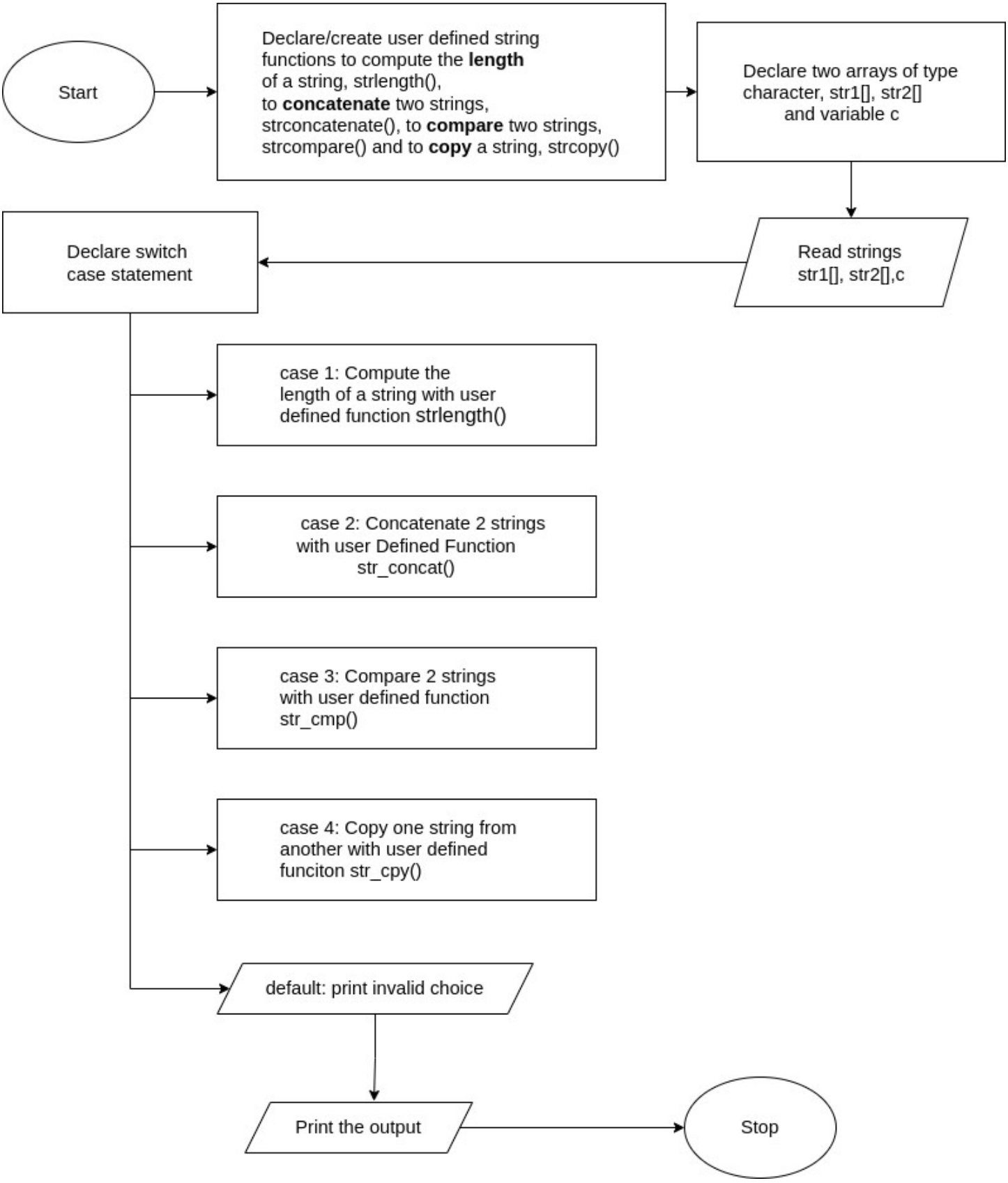
break;

default:

Invalid choice

Step 6: Stop

Flowchart:


Start
Declare/create user defined string functions to compute the **length** of a string, strlength(), to **concatenate** two strings, strconcatenate(), to **compare** two strings, strcompare() and to **copy** a string, strcopy()
Declare two arrays of type character, str1[], str2[] and variable c
Read strings str1[], str2[],c
Declare switch case statement
case 1: Compute the length of a string with user defined function strlength()
case 2: Concatenate 2 strings with user Defined Function str_concat()
case 3: Compare 2 strings with user defined function str_cmp()
case 4: Copy one string from another with user defined funciton str_cpy()
default: print invalid choice
Print the output
Stop

CODE:

```
//      strlen() computes string's length strcpy() copies a string to another
//      strcat() concatenates(joins) two strings
//      strcmp() compares two strings
//      strlwr() converts string to lowercase
//      strupr() converts string to uppercase

// Write a menu driven program to perform all string operations (user defined
functions)

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// User Defined Functions
int str_length(char *ptr)
{
    /*
        Function: Returns the number of characters in the given character.
        Input: char * pointing to the character array.
        Returns: Integer.
     */
    int count = 0;
    for (int i = 0; ptr[i] != '\0'; i++)
    {
        count++;
    }
    return count;
}
char *str_concat(char *user_string_1, char *user_string_2)
{
    /*
        Function: Returns a character pointer pointing to an array of characters that
is made by concatenating 2 strings.
        Input: char * pointing to the 2 strings.
        Returns: char *.
     */
```

```
    // allocating on the heap coz otherwise it would be a local variable
    // that you cant pass outside the scope of this function as a pointer, as memory
would be invalid.
    char *concat_string = malloc(1000);
    strcpy(concat_string, user_string_1);
    for (int i = 0; i ⩽ str_length(user_string_2); i++)
    {
        concat_string[str_length(user_string_1) + i] = user_string_2[i];
    }
    return concat_string;
}
int str_compare(char *user_string_1, char *user_string_2)
{
    /*
        Compares the C string str1 to the C string str2.
        This function starts comparing the first character of each string.
        If they are equal to each other, it continues with the following pairs until
the characters
        differ or until a terminating null-character is reached.
        Returns:
        <0  the first character that does not match has a lower value in ptr1 than in
ptr2
        0   the contents of both strings are equal
        >0  the first character that does not match has a greater value in ptr1 than
in ptr2
    */
    int result = 0;
    for (int i = 0; user_string_1[i] ≠ '\0' || user_string_2[i] ≠ '\0'; i++)
    {
        if (user_string_1[i] == user_string_2[i])
        {
            if (user_string_1[i + 1] == '\0' && user_string_2[i + 1] ≠ '\0')
            {
                result = 0;
                continue;
            }
            else if (user_string_1[i + 1] == '\0' && user_string_2[i + 1] ≠ '\0')
            {
                result = -1;
```

```
            }
            else if (user_string_1[i + 1] == '\0' && user_string_2[i + 1] == '\0')
            {
                result = 0;
            }
            if (user_string_1[i + 1] != '\0' && user_string_2[i + 1] == '\0')
            {
                result = 1;
            }
        }
        else if (user_string_1[i] < user_string_2[i] || user_string_1[i] >
user_string_2[i])
        {
            result = (user_string_1[i] - user_string_2[i]) / abs(user_string_1[i] -
user_string_2[i]);
            break;
        }
    }
}

void str_cpy(char *str_to_copy, char *user_string_2)
{
    /*
    Function: Copies string 2 to string 1.
    Input: char * pointing to the 2 strings.
    */

    for (int i = 0; i < str_length(user_string_2); i++)
    {
        str_to_copy[i] = user_string_2[i];
    }
}
char *str_lower(char *user_string)
{
    /*
    Returns a new char * to an array that contains the converted lowercase of the
user_string
    */
    char *lower_string = malloc(1000);
    strcpy(lower_string, user_string);
```

```
    for (int i = 0; i < str_length(lower_string); i++)
    {
        if (lower_string[i] ⩾ 'A' && lower_string[i] ⩽ 'Z')
        {
            int AASCI_val = lower_string[i] + 32;
            lower_string[i] = AASCI_val;
        }
    }
    return lower_string;
}
char *str_upper(char *user_string)
{
    /*
    Returns a new char * to an array that contains the converted uppercase of the
user_string
    */
    char *upper_string = malloc(1000);
    strcpy(upper_string, user_string);
    for (int i = 0; i < str_length(upper_string); i++)
    {
        if (upper_string[i] ⩾ 'A' && upper_string[i] ⩽ 'Z')
        {
            int AASCI_val = upper_string[i] - 32;
            upper_string[i] = AASCI_val;
        }
    }
    return upper_string;
}
char *str_reverse(char *user_string)
{
    /*
    Returns a new char * to an array that contains the reversed user_string
    */

    // allocating on the heap coz otherwise it would be a local variable
    // that you cant pass outside the scope of this function as a pointer, as memory
would be invalid.
    char *rev_string = malloc(1000);
    strcpy(rev_string, user_string);
    for (int i = 0; i < str_length(user_string); i++)
```

```
    {
        rev_string[i] = user_string[str_length(user_string) - i - 1];
    }
    rev_string[str_length(user_string)] = '\0';
    return rev_string;
}

int main()
{
    int choice = 0;
    char user_string[500];
    char user_string_1[500], user_string_2[500];

    printf("Enter What operation you want to perform [1, 2, 3, 4, 5]: \n\
        1. Find the length of the String\n\
        2. Concatenate 2 Strings\n\
        3. Compare 2 Strings\n\
        4. Convert a String to lowercase\n\
        5. Convert a String to Uppercase\n\
        6. Reverse a string\n\
        ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
        printf("Enter the String that you want to find the length of: ");
        scanf("%s", &user_string);
        printf("The Length is: %d", str_length(user_string));
        break;
    case 2:
        printf("Enter the First String: ");
        scanf("%s", &user_string_1);
        printf("Enter the First String: ");
        scanf("%s", &user_string_2);
        printf("The Concatenated is: %s", str_concat(user_string_1, user_string_2));
        break;
    case 3:
        printf("Enter the First String: ");
        scanf("%s", &user_string_1);
        printf("Enter the First String: ");
```

```
        scanf("%s", &user_string_2);
        printf("The Comparison of the Strings is: %d", str_compare(user_string_1,
user_string_2));
        break;
    case 4:
        printf("Enter the String that you want to convert to lowercase to: ");
        scanf("%s", &user_string);
        printf("The converted String is: %s", str_lower(user_string));
        break;
    case 5:
        printf("Enter the String that you want to convert to Uppercase to: ");
        scanf("%s", &user_string);
        printf("The converted String is: %s", str_upper(user_string));
        break;
    case 6:
        printf("Enter the String that you want to reverse: ");
        scanf("%s", &user_string);
        printf("The converted String is: %s", str_reverse(user_string));
        break;
    case 7:
        printf(" the String that you want to copy: ");
        scanf("%s", &user_string_2);
        str_cpy(user_string_1, user_string_2);
        printf("The copied strings are: %s and %s", user_string_1, user_string_1);
        break;
    default:
        printf("Incorrect Choice, Please try again.");
    }
    return 0;
}
```

OUTPUT

```
What operation you want to perform [1, 2, 3, 4, 5]:
        1. Find the length of the String
        2. Concatenate 2 Strings
        3. Compare 2 Strings
        4. Convert a String to lowercase
        5. Convert a String to Uppercase
```

```
        6. Reverse a string
        1
Enter the String that you want to find the length of: example
The Length is: 7

2
Enter the First String: example
Enter the First String: String
The Concatenated is: exampleString

3
Enter the First String: First
Enter the First String: Second
The Comparison of the Strings is: -1

3
Enter the First String: First
Enter the First String: First
The Comparison of the Strings is: 0

4
Enter the String that you want to convert to lowercase to: LOWERcase
The converted String is: lowercase

5
Enter the String that you want to convert to Uppercase to: upperCASE
The converted String is: UPPERCASE

6
Enter the String that you want to reverse: reverse
The converted String is: esrever

7
the String that you want to copy: hello_world
The copied strings are: hello_world and hello_world
```

CONCLUSION:

Thus we have learned various string operations in C

FAQs:

Q1. How to find the maximum occurring character in given String?

```c
int max_occuring_char(char *user_string)
{
    int max_value = 0;
    int characters[str_length(user_string)];
    int occured_before = 0;
    for (int i = 0; i < str_length(user_string); i++)
    {
        // Check if the character has occured before so that it doesnt get evaluated twice.
        for (int k = 0; k < i; k++)
        {
            occured_before = 0;
            if (user_string[k] == user_string[i])
            {
                occured_before = 1;
                break;
            }
        }

        // count the number of times each character has occured in the string.
        characters[i] = 1;
        for (int j = 0; j < str_length(user_string) && (occured_before == 0); j++)
        {

            if (j != i)
            {
                if (user_string[i] == user_string[j])
                {
                    characters[i]++;
                }
            }
        }

        // finding the maximum value
        for (int l = 0; l < str_length(user_string); l++)
        {
            if (characters[i] > max_value)
            {
                max_value = characters[i];
            }
        }
    }
    // Printing the character that has been used the most times.
    for (int i = 0; i < str_length(user_string); i++)
    {
        if (max_value == characters[i])
        {
            printf("The Character %c has been used %d times in the string\n", user_string[i], characters[i]);
        }
    }
}
```

Q2. How to remove all duplicates from a given string?

```c
void remove_duplicates(char *user_string)
{
    int new_string_length = str_length(user_string);
    for (int i = 0; i < str_length(user_string); i++) {
        for (int j = 0; j < new_string_length; j++) {
            // Shifting the rest of the string backwards if there is a duplicate
            if ((user_string[i] == user_string[j]) && (i != j)) {
                for (int k = j; k <= user_string[k] != '\0'; k++) {
                    user_string[k] = user_string[k + 1];
                    new_string_length--;
                }
            }
        }
    }
    printf("The String without the duplicates is: %s", user_string);
}
```

Q3. How do you check if a given String is Palindrome or not?

```c
int if_palindrome(char *user_string)
{
    int palindrome = 0;
    for (int i = 0; i < str_length(user_string) / 2; i++)
    {
        if (user_string[i] == user_string[str_length(user_string) - 1 - i])
        {
            palindrome = 1;
            continue;
        }
        else
        {
            palindrome = 0;
            break;
        }
    }
    return palindrome;
}
```