**String Programs C**

Program 1:

In the following C program, user would be asked to enter a String (it can be in complete uppercase or partial uppercase) and then the program would convert it into a complete(all characters in lower case) lower case string. The logic we have used in the following program is: All the upper case characters (A-Z) have ASCII value ranging from 65 to 90 and their corresponding lower case characters (a-z) have ASCII value 32 greater than them. For example ‘A‘ has a ASCII value 65 and ‘a‘ has a ASCII value 97 (65+32). Same applies for other characters.

/\* C program to convert uppercase string to

\* lower case

\*/

#include<stdio.h>

#include<string.h>

int main(){

/\* This array can hold a string of upto 25

\* chars, if you are going to enter larger string

\* then increase the array size accordingly

\*/

char str[25];

int i;

printf("Enter the string: ");

scanf("%s",str);

for(i=0;i<=strlen(str);i++){

if(str[i]>=65&&str[i]<=90)

str[i]=str[i]+32;

}

printf("\nLower Case String is: %s",str);

return 0;

}

Program 2:

In the following program, user would be asked to enter a lower case String and the program would convert it into a Upper case String. Logic followed in the program: All lower case characters (a to z) have ASCII values ranging from 97 to 122 and their corresponding upper case characters (A to Z) have ASCII values 32 less than them. For example ‘a’ has a ASCII value 97 and ‘A’ has a ASCII value 65 (97-32). Same applies for other alphabets. Based on this logic we have written the below C program for conversion.

#### C program – Conversion of a String from lowercase to uppercase

/\* C Program to convert Lower case

\* String to Upper case.

\*/

#include<stdio.h>

#include<string.h>

int main(){

char str[25];

int i;

printf("Enter the string:");

scanf("%s",str);

for(i=0;i<=strlen(str);i++){

if(str[i]>=97&&str[i]<=122)

str[i]=str[i]-32;

}

printf("\nUpper Case String is: %s",str);

return 0;

}

Program 3:

In the following program user would be asked to enter a set of Strings and the program would sort and display them in ascending alphabetical order.

#### C Program – Sorting of a Set of Strings in Ascending alphabetical order

/\* This program would sort the input strings in

\* an ascending order and would display the same

\*/

#include<stdio.h>

#include<string.h>

int main(){

int i,j,count;

char str[25][25],temp[25];

puts("How many strings u are going to enter?: ");

scanf("%d",&count);

puts("Enter Strings one by one: ");

for(i=0;i<=count;i++)

gets(str[i]);

for(i=0;i<=count;i++)

for(j=i+1;j<=count;j++){

if(strcmp(str[i],str[j])>0){

strcpy(temp,str[i]);

strcpy(str[i],str[j]);

strcpy(str[j],temp);

}

}

printf("Order of Sorted Strings:");

for(i=0;i<=count;i++)

puts(str[i]);

return 0;

}

Program 4:

In the following C program we are counting the number of characters in a given String to find out and display its length on console. Upon execution of this program, the user would be asked to enter a string, then the program would count the chars and would display the length of input string as output.

#### C Program – finding length of a String without using standard library function strlen

/\* C Program to find the length of a String without

\* using any standard library function

\*/

#include <stdio.h>

int main()

{

/\* Here we are taking a char array of size

\* 100 which means this array can hold a string

\* of 100 chars. You can change this as per requirement

\*/

char str[100],i;

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

scanf("%s",str);

// '\0' represents end of String

for(i=0; str[i]!='\0'; ++i);

printf("\nLength of input string: %d",i);

return 0;

}

Program 5:

In the following program user would be asked to enter two strings and then the program would concatenate them. For concatenation we have not used the standard library function strcat(), instead we have written a logic to append the second string at the end of first string.

#### C program for string concatenation

/\* C program to concatenate two strings without

\* using standard library function strcat()

\*/

#include <stdio.h>

int main()

{

char str1[50], str2[50], i, j;

printf("\nEnter first string: ");

scanf("%s",str1);

printf("\nEnter second string: ");

scanf("%s",str2);

/\* This loop is to store the length of str1 in i

\* It just counts the number of characters in str1

\* You can also use strlen instead of this.

\*/

for(i=0; str1[i]!='\0'; ++i);

/\* This loop would concatenate the string str2 at

\* the end of str1

\*/

for(j=0; str2[j]!='\0'; ++j, ++i)

{

str1[i]=str2[j];

}

// \0 represents end of string

str1[i]='\0';

printf("\nOutput: %s",str1);

return 0;

}

Program 6:

Here we have defined a function reverse\_string, this function calls itself recursively.

#include <stdio.h>

#include <string.h>

void reverse\_string(char\*, int, int);

int main()

{

//This array would hold the string upto 150 char

char string\_array[150];

printf("Enter any string:");

scanf("%s", &string\_array);

//Calling our user defined function

reverse\_string(string\_array, 0, strlen(string\_array)-1);

printf("\nReversed String is: %s",string\_array);

return 0;

}

void reverse\_string(char \*x, int start, int end)

{

char ch;

if (start >= end)

return;

ch = \*(x+start);

\*(x+start) = \*(x+end);

\*(x+end) = ch;

//Function calling itself: Recursion

reverse\_string(x, ++start, --end);

}