Programming in C
Strings

STRINGS

The string in C programming language is actually a one-dimensional array of characters which is terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

If you follow the rule of array initialization then you can write the above statement as follows:

```
char greeting[] = "Hello";
```

Following is the memory presentation of above defined string in C:

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Actually, you do not place the null character at the end of a string constant. The C compiler automatically places the '\0' at the end of the string when it initializes the array. Let us try to print above mentioned string:

Use of String:

String is a long flexible structure made from fibers twisted together into a single strand, or from multiple such strands which are in turn twisted together. String is used to tie, bind, or hang other objects. It is also used as a material to make things, such as textiles, and in arts and crafts.

```
#include int main ()
{
        char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
        printf("Greeting message: %s\n", greeting );
        return 0;
}

Output:   Greeting message: Hello
```

C supports a wide range of functions that manipulate null-terminated strings:

1. strcpys1, s2;

    Copies string s2 into string s1.

2. strcats1, s2;

    Concatenates string s2 onto the end of string s1.

3. strlen s1;

    Returns the length of string s1

4. strcmps1, s2;

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

5. strchrs1, ch;

    Returns a pointer to the first occurrence of character ch in string s1.

6. strstrs1, s2;

    Returns a pointer to the first occurrence of string s2 in string s1.

Following example makes use of few of the above-mentioned functions:

```
#include <stdio.h>
#include <string.h>
int main ()
{
        char str1[12] = "Hello";
       char str2[12] = "World";
       char str3[12];
        int len ;

        /* copy str1 into str3 */
        strcpy(str3, str1);
        printf("strcpy( str3, str1) : %s\n", str3 );

        /* concatenates str1 and str2 */
        strcat( str1, str2);
        printf("strcat( str1, str2): %s\n", str1 );

        /* total length of str1 after concatenation */
        len = strlen(str1);
        printf("strlen(str1) : %d\n", len );

        return 0;
}
```

Output:
strcpy(str3, str1) : Hello
strcat(str1, str2): HelloWorld
strlen(str1) : 10

Programs:

1. Frequency of a character in the given string

```
#include <stdio.h>
int main() {
   char str[1000], ch;
   int count = 0;

   printf("Enter a string: ");
   fgets(str, sizeof(str), stdin);

   printf("Enter a character to find its frequency: ");
   scanf("%c", &ch);

   for (int i = 0; str[i] != '\0'; ++i) {
     if (ch == str[i])
        ++count;
   }

   printf("Frequency of %c = %d", ch, count);
   return 0;
}
```

Output:

Enter a string: Stay home stay safe.
Enter a character to find its frequency: a
Frequency of a = 3

2.   Reverse of the string

```
#include <stdio.h>
void reverseSentence();
int main() {
   printf("Enter a sentence: ");
   reverseSentence();
   return 0;
}

void reverseSentence() {
   char c;
   scanf("%c", &c);
   if (c != '\n') {
      reverseSentence();
```

3

```c
    printf("%c", c);
  }
}
```

Output:

Enter a sentence: Welcome to apssdc
cdsspa ot emocleW

3.   Remove characters in a string except Alphabets
4.   #include <stdio.h>

```c
int main() {
  char line[150];

  printf("Enter a string: ");
  fgets(line, sizeof(line), stdin); // take input


  for (int i = 0, j; line[i] != '\0'; ++i) {

    // enter the loop if the character is not an alphabet
    // and not the null character
    while (!(line[i] >= 'a' && line[i] <= 'z') && !(line[i] >= 'A' && line[i] <= 'Z') && !(line[i] ==
'\0')) {
      for (j = i; line[j] != '\0'; ++j) {

        // if jth element of line is not an alphabet,
        // assign the value of (j+1)th element to the jth element
        line[j] = line[j + 1];
      }
      line[j] = '\0';
    }
  }
  printf("Output String: ");
  puts(line);
  return 0;
}
```

Output:
Enter a string: p2'r-o@gram84iz./
Output String: programiz