Problem Solving Through programming in C
Course Code: ONL1001

ARRAYS(string handling function and array examples)

Ms. SHUBHRA DWIVEDI
School - SCOPE
VIT-AP Amaravati

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 the above mentioned string −

```
#include <stdio.h>

int main () {

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

Output:
Greeting message: Hello

The C language does not provide an inbuilt data type for strings but it has an access specifier “%s” which can be used to directly print and read strings.

Below is a sample program to read a string from user:

```
// C program to read strings

#include<stdio.h>

int main()
{
    // declaring string
    char str[50];

    // reading string
    scanf("%s",str);
    //gets(str);

    // print string
    printf("%s",str); return 0;
}
```

Output: Welcome India
Welcome

You can see in the above program that string can also be read using a single scanf statement. Also you might be thinking that why we have not used the ‘&’ sign with string name ‘str’ in scanf statement! To understand this you will have to recall your knowledge of scanf. We know that the ‘&’ sign is used to provide the address of the variable to the scanf() function to store the value read in memory. As str[] is a character array so using str without braces ‘[‘ and ‘]’ will give the base address of this string. That’s why we have not used ‘&’ in this case as we are already providing the base address of the string to scanf.

String Handling Functions

C language supports a large number of string handling functions that can be used to carry out many of the string manipulations. These functions are packaged in string.h library. Hence, you must include string.h header file in your programs to use these functions.

The following are the most commonly used string handling functions.

Sr.No. Function & Purpose

1. strcpy(s1, s2);Copies string s2 into string s1.

2. strcat(s1, s2);Concatenates string s2 onto the end of string s1.

3. strlen(s1);Returns the length of string s1

4. strcmp(s1, s2);Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.

. 5. strrev(); It is used to show reverse of a string

strcmp() function:

- strcmp() function will return the ASCII difference between first unmatching character of two strings.
- strcmp() compares the two strings lexicographically means it starts comparison character by character starting from the first character until the characters in both strings are equal or a NULL character is encountered.
- If first character in both strings are equal, then this function will check the second character, if this is also equal then it will check the third and so on
- This process will be continued until a character in either string is NULL or the characters are unequal.

1. Zero (0): A value equal to zero when both strings are found to be identical. That is, That is, All of the characters in both strings are same.

2. Less than Zero (<0): A value less than zero is returned when the first not matching character in leftStr have lesser ASCII value than the corresponding character in rightStr.

If character in leftStr is lexicographically before the character of rightStr.

3. Greater than zero (>0): A value greater than zero is returned when the first not matching character in leftStr have the greater ASCII value than the corresponding character in rightStr or we can also say

If character in leftStr is lexicographically after the character of rightStr

```
// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>

int main()
{

    char leftStr[] = "g f g";
    char rightStr[] = "g f g";

    // Using strcmp()
    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue returned by strcmp() is:  %d" , res);
    return 0;
}
```

Output:

```
Strings are equal
Value returned by strcmp() is:  0
```

```
// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>
int main()
{
    // z has greater ASCII value than g
    char leftStr[] = "zfz";
    char rightStr[] = "gfg";

    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue of result: %d" , res);

    return 0;
}
```

Output:

```
Strings are unequal
Value returned by strcmp() is:  19
```

```
// C program to illustrate
// strcmp() function
#include<stdio.h>
#include<string.h>
int main()
{
    // b has less ASCII value than g
    char leftStr[] = "bfb";
    char rightStr[] = "gfg";

    int res = strcmp(leftStr, rightStr);

    if (res==0)
        printf("Strings are equal");
    else
        printf("Strings are unequal");

    printf("\nValue returned by strcmp() is:  %d" , res);


    return 0;
}
```

Output:

```
Strings are unequal
Value returned by strcmp() is:  -5
```

strrev() function:

It is used to reverse the given string expression.

```
#include<stdio.h>

int main()
{
    char s1[50];

    printf("Enter your string: ");
    gets(s1);
    printf("\nYour reverse string is: %s",strrev(s1));
    return(0);
}
```

Output:

```
Enter your string: studytonight
Your reverse string is: thginotyduts
```

```
#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 lenghth 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
```

Write a program for entering data into an array & Reading data from an array.

```
#include<stdio.h>
void main()
{
int arr[10],i,n;
printf("\n Enter N Elements");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter arr[%d]:",i);
scanf("%d",&arr[i]);
}
for(i=0;i<n;i++)
{
printf("%d\n",arr[i]);
}
}
```

Output:

```
Enter N Elements  3
Enter arr[0]  :  2
Enter arr[1]  :  5
Enter arr[2]  :  3
      2
      5
      3
```

PROGRAM-ARRAY INITIALIZATION

Array Initialization

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5]={10,20,30,40,50};
int i;
clrscr();
for(i=0;i<5;i++)
{
printf("%d\n",a[i]);
}
getch();
}
```

Output:

10
20
30
40
50

/*C program to sort an one dimensional array in ascending order.*/

```
#include <stdio.h>
#define MAX 100
int main()
{
   int arr[MAX],n,i,j;
   int temp;

   printf("Enter total number of elements: ");
   scanf("%d",&n);

   //read array elements
   printf("Enter array elements:\n");
   for(i=0;i< n;i++)
   {
      printf("Enter element %d: ",i+1);
      scanf("%d",&arr[i]);
   }
   //sort array
   for(i=0;i< n;i++)
   {
      for(j=i+1;j< n;j++)
      {
         if(arr[i]>arr[j])
         {
            temp    =arr[i];
            arr[i]  =arr[j];
            arr[j]  =temp;
         }
      }
   }
   printf("\nArray elements after
   sorting:\n");
      for(i=0;i< n;i++)
      {
         printf("%d\n",arr[i]);
      }
      return 0;
   }
```

Output

```
Enter total number of elements: 5
Enter array elements:
Enter element 1: 100
Enter element 2: 999
Enter element 3: 200
Enter element 4: 800
Enter element 5: 300

Array elements after sorting:
100
200
300
800
999
```

Example of Array In C programming to find out the average of 4 integers

```
#include <stdio.h>
int main()
{
    int avg = 0;
    int sum =0;
    int x=0;

    /* Array- declaration – length 4*/
    int num[4];

    /* We are using a for loop to traverse through the array
     * while storing the entered values in the array
     */
    for (x=0; x<4;x++)
    {
        printf("Enter number %d \n", (x+1));
        scanf("%d", &num[x]);
    }
    for (x=0; x<4;x++)
    {
        sum = sum+num[x];
    }

    avg = sum/4;
    printf("Average of entered number is: %d", avg);
    return 0;
}
```

Output:

```
Enter number 1
10
Enter number 2
10
Enter number 3
20
Enter number 4
40
Average of entered number is: 20
```