# Chapter 8

# STRINGS

***LEARNING OBJECTIVES***
*After going this chapter the reader will be able to*
- *Use different input/output functions for string*
- *Learn the usage of important string manipulation functions available in the header file string.h*
- *Write his own functions for string manipulation*
- *Learn the storage and manipulation of multiple strings.*

## 8.1. INTRODUCTION:

- **String** is a sequence of characters treated as a single data item.
- Any group of characters(except double quotes) enclosed between double quotation marks is a **string constant.**
  "Vijayawada" ,"green fields","1234" , "5" ,"A" are string constants

- 5 is integer constant and is assigned to an integer variable
- '5' is character constant and is assigned to a character variable.
- "5" is string constant and is assigned to a string variable

If the double quote is also a character of the string we may include it preceded by black slash sign.

"welcome\"" is a string constant

C does not support string data type . Strings in C are represented by arrays of characters.

A string is stored in an array of characters as shown below

The end of the string is marked with a special character , the null character '\0' whose ASCII value is zero.

char city [10] = "HYDERABAD";

The above string is stored as

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| H | Y | D | E | R | A | B | A | D | \0 |

The name of the character array 'city' is pointer to the beginning of the string. The compiler automatically append null character (\0) at the end of the string. The ASCII value of the null character is zero.(null character and the null pointer are different . In the ASCII character set , the null character is named as NUL)

Null string is represented as "" and it is stored as

| |
|---|
| \0 |

## 8.2. DECLARING AND INITIALIZING STRINGS

A string variable is any valid C variable name and it is always declared as an array. Hence to declare a string variable we use character array. The general format is

```
char str[size];
```

Where str is a string variable  and the size determines the number of characters in the string. Consider the following array declarations

```
char  city[10],name[20],address[25];
```

Where city, name and address are character arrays which can hold strings of length 9,19 and 24 characters respectively

**Initializing strings**:

We can initialize a string the same way that we initialize any other variable by assigning a value to it when it is defined. In this case the value is a string. For example

```
char city[10] = “HYDERABAD”;
```

If we declare city as

```
char city[9] = “HYDERABAD”;
```

then the null character will not be appended automatically to the character array . because , city can hold 9 characters and the string HYDERABAD has already 9 characters filling the 9 spaces.

Since a string is stored in a character array , while initializing  through type declaration, size of the array can be omitted as shown below.

```
char city[ ] = “HYDERABAD”;
```

In this case compiler will create an array of 10 bytes and initialize it with HYDERABAD followed by null character

- A string may be initialized as an array of characters.
  ```
  char city[ ] = {‘H’,’Y’,’D’,’E’,’R’,’A’,’B’,’A’,’D’,’\0’};
  ```
- A character  pointer  may be initialized with a string.
  ```
  char *city = “HYDERABAD”;
  ```
- Since a string variable is a character array, one string variable cannot be assigned to another string variable.
  ```
  char str1[6 ] = “green”,str2[6];

  str2 = str1;
  ```

The above assignment leads to compilation error.

- When pointers are initialized with strings we can assign one string to another string.
  ```
  char *str1=”green”,*str2;

  str2=str1;
  ```
  The above assignment is valid.

- A character array cannot be initialized with a string by an assignment statement.
  char city[10];

  city = "HYDERABAD";

  The above assignment is invalid.

## 8.3. READING STRINGS FROM KEYBOARD

There are different methods to read a string

**Using %s control with scanf()**

The familiar input function scanf ( ) can be used with %s conversion specification to read string of characters.

**Example 8.1:**

```
char city[15];

scanf("%s",city);
```

If the input is

```
VIJAYAWADA
```

then it is stored in the memory as given below.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| V | I | J | A | Y | A | W | A | D | A | \0 | ? | ? | ? | ? |

The '\0' (null character) is appended automatically after the last character to designate the end of the string.

Note: Thus the unused locations are filled with garbage

The problem with the scanf function is that it terminates its input on the occurrence of first white space .

**Example8.2:**

```
char city[20];

scanf("%s",city);
```

If the input is

```
GREEN FIELDS
```

Only first word GREEN is read and is stored in the memory as given below

| | | | | | |
|---|---|---|---|---|---|
| G | R | E | E | N | \0 |

**Using gets() function:**

gets() function is useful to read strings from the keyboard. The general syntax is

```
gets(str);
```

str is any valid string variable name and it will read a string of characters until enter key is pressed(i.e until new line character is entered).

**Example8.3:**

```
char city[20];

gets(str);
```

If the input is

GREEN FIELDS

then the string GREEN  FIELDS    is stored in the memory as

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | R | E | E | N | | F | I | E | L | D | S | \0 |

The '\0' (null character) is appended automatically at the end of the string to designate the end of the string.

**Using scan set:**

The function scanf() also allows for the input of string through the scan set , which is a sequence of characters enclosed in square brackets[] and proceeded by a % sign . The scan set causes the function scanf() to search for the characters in the input string that matches those in the scan set. Only those matching characters are stored in the corresponding array. The scan set operation terminates once a character not contained in the scan set is encountered.

**Example 8.4:**

```
char str[10];

scanf("%[pqrst]",str);
```

If the input is  pqrsx

Then the string pqrs is stored in the memory as

| | | | | |
|---|---|---|---|---|
| p | q | r | s | \0 |

**Using inverted scan set**:

The inverted scan set can also be used for character input. The inverted scan set does the opposite of scan set. This means that only characters not appearing with in the square brackets are stored inside the character array. An inverted scan set is identified by a caret ^ with in the square brackets and preceding the character sets

**Example 8.5:**

```
char str[20];
```

```
   scanf(”%[^\n]”,str);
```

The string entered through key board until new line character is taken and is stored in the array str.

If the input string is GREEN FIELDS

Then it is stored in the memory as

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | R | E | E | N | | F | I | E | L | D | S | \0 |

**Using getchar() function**:

A string may also be read character by character using getchar() function.

**Example8.6:**

```
   char str[50];

   int i=0;

   while((str[i]=getchar()) !=’\n’)

   {

      ++i;

   }

   str[i] =’\0’;
```

Note that null character(‘\0’) must be appended explicitly in this case. We can also use any other character in place of ‘\n’ and the string will be read until that character is entered.

**8.4.WRITING STRINGS TO THE SCREEN:**

**Using %s control with printf():**

We can use printf ( )function with %s conversion to print string. The string is taken from the memory until the occurrence of null character and is displayed. For example,

```
  char str[ 20];

printf(“%s”,str);
```

Unlike scanf() even if the string contains white space characters entire string is displayed

**Example8.7:**

```
      char str[20]=”GREEN  FIELDS”;

      printf(“%s”,str);
```

would output

```
         GREEN  FIELDS
```

Note that the stored string has a white space characters.

**Using puts() function :**
puts() function is useful to print/display string on the screen. The general syntax is

```
puts(str);
```

Where str is any valid string variable name.

**Program 8.1** program to read and write string using gets and puts functions.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   char name[20];
   clrscr();
   printf("enter string  from keyboard\n");
   gets(name);
   printf("the string is\n ");
   puts(name);
   getch();
}
```

The output for above program is

```
Enter string from keyboard
Good morning
The string  is
Good morning
```

**Program 8.2** Program to copy one string into another string

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char source[20],destination[20];
    int i=0;
    clrscr();
    printf("enter any string  from keyboard\n");
    gets(source);
    while((source[i] = destination[i] ) !='\0')
         i++;
    printf("the destination string is \n",);
    puts(destination);
    getch();
}
```

The character of the string destination are copied into the string source one by one until null character is copied. After the null character is copied, while loop condition becomes false and control is exited from the loop.

## 8.5. STRING MANIPULATION FUNCTIONS

C language supports several functions useful for string manipulation. Whenever these functions are used the header file string.h must be included. Some of the functions are:

**1. strlen():** This function returns length of a string, that is counts no of characters in the string excluding null character. If the string is empty it returns zero. The general format of this function is

```
n=strlen(str);
```

**Where n is an integer variable which receives the length of the string str.**

**Program 8.3** Program to find length of string**.**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char str[20];
   int n;
   clrscr();
   printf("enter any string \n");
   scanf("%s",str);
   n=strlen(str);
   printf("the length of string %s is %d",str,n);
   getch();
}
```

The output for above program is

```
Enter any string

Welcome

The length of the string welcome  is 7
```

**2.strcat():** This function is useful to join two strings together. The general format of strcat() function is

```
strcat(str1,str2);
```

str1, str2 are two string variables. When strcat() function is executed str2 is appended to strg1 and str2 remains unchanged.

**Program 8.4** Program to join two strings "hello" and "world"

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
     char str1[20] ="hello",str2[20] ="world";
     clrscr();
     printf("the two strings before concatenation\n");
     printf("the first string is %s",str1);
     printf("\n second string is %s",str2);
```

```
        strcat(str1,str2);
        printf("the two strings after concatenation\n");
        printf("\nthe first string is %s",str1);
        printf("\n second string is %s",str2);
        getch();
    }
```

**3.strcmp():** This function is useful to compare to strings. If the two strings are similar then it returns 0 , otherwise , it returns the difference between the ASCII values of first pair of non-matching characters. The general syntax is

```
strcmp(str1,str2);
```

str1 and str2 may be string variables or string constants.

The returned value is

Zero if the strings are identical.

Positive if the strings are not in alphabetical order.

Negative if the strings are in alphabetical order.

**Program8.5.** Program to check the given strings are same or not.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   int n;
   char str1[20],str2[20];
   clrscr();
   printf("enter any two strings\n");
   scanf("%s%s",str1,str2);
   n=strcmp(str1,str2);
   if (n==0)
      printf("two strings are same");
   else
      printf("strings are not same");
   getch();
}
```

**4.strcpy():** This function is useful to copy one string into another string. The general syntax is

```
strcpy(str1,str2);
```

The content of str2 will be copied into str1.

**Program8.6:** Program to copy one string into another string**.**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
       char str1[20],str2[20];
       clrscr();
       printf("Enter the first string\n");
      gets(str1);
      printf("Enter the second string\n");
      gets(str2);
      printf("strings before copying\n")
      printf("%s\t%s",str1,str2);
      strcpy(str1,str2);
      printf("two strings after copying\n");
      printf(("%s\t%s",str1,str2);
     getch();
}
```

**Program8.7 :** Program to check the given string is palindrome.

```
#include<stdio.h>
#include<string.h>
void main()
{
     char str[100];
     int i,n,j,flag=0;
     printf("enter the string to be checked\n");
     gets(str);
     n=strlen(str);
     for(i=0, j=n-1; i<j; ++i, --j)
     {
            if(str[i] != str[j])
            {
                flag=1;
                 break;
            }
      }
      if (flag ==1)
      printf("given string is not a  palindrome");
      else
      print("given string is  palindrome");
}
```

Using the function strlen() length of the string is obtained and is assigned to n. Initial values of i and j are 0 and n-1 respectively, represents the first and last positions of the string. If the given string is not palindrome, a pair of corresponding characters from the beginning and end are unequal and the loop is terminated after 1 is assigned to flag, otherwise, the initial value of flag 0 is not changed.

The following are some other string functions which we commonly use in c programming language**.**

| S.No | Function name | Description |
|---|---|---|
| 1 | strrev() | reverses a string |
| 2 | strstr() | locates a substring in a string |
| 3 | strchr() | locates the first occurrence of given character in a string |
| 4 | strlwr() | converts a string into lower case |
| 5 | strupr() | converts a string into upper case |

**1.strrev():** This function reverses the string. The general syntax is

```
strrev(string);
```

**Example 8.8**:

```
char str[10] = ”GREEN”;
strrev(str);
printf(“the reverse string is %s\n”,str);
```

Output:

```
the reverse string is NEERG.
```

**2.strstr():** This function is used to locate a sub string in a string. The general syntax is

```
strstr(str1,str2);
```

This function searches str1 to see whether str2 is contained in str1.If yes, then the function returns the position of the first occurrence of the sub string str2 otherwise, it returns a NULL pointer.

**3.strchr()** This function is used to locate the first occurrence of a character in a string. The general syntax is

```
strchr(str1,character);
```

**Example 8.9:**

```
strchr(“this is an example”,’a’);
```

**4.strlwr()**: This function is useful to convert upper case string into lower case. The general syntax is

String is any valid string variable which holds upper case data.

```
strlwr(str);
```

**5.strupr():** This function is useful to convert lower-case data into upper-case.The general format for this function is

```
strupr(str);
```

"str" is any valid string variable , which contains lower-case data.

**8.6. ARRAY OF STRINGS:**

A list of names can be treated as a table of strings and two dimensional character arrays can be used to store the entire list.

The character array str[20][15] may be used to store a list of 20 names.

Since a two dimensional array is an array of one dimensional arrays, str[0],str[1],str[2] etc represent the pointers to first, second third etc rows of the two dimensional array str.Thus the following code can be used to store the list of names in the two dimensional array.

```
for(i=0;i<n;++i)

    gets(str[i]);
```

Similarly , the following code can be used to display the list of names.

```
for(i=0;i<n;++i)

    puts(str[i]);
```

**Program 8.9** Program to sort a list of names**.**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
   char str[10][20],temp[20];
   int n,p,i,j;
   clrscr();
   printf("Enter the strings one by one ,at the end of the list type END:\n");
   i=0;
   while(i)
   {
      gets(str[i]);
      if(strcmp(str[i],"END")==0)
         break;
      ++i;
   }
   n=i;
   /* sorting process begins*/
   for(p= 1; p < n;pi++)
   {
      for(j = 0; j <= n-p; j++)
      {
         if (strcmp(str[j],str[j+1]>0)
         {
            strcpy(temp,str[j]);
            strcpy(str[j],str[j+1]);
            strcpy(str[j+1],temp);
         }
      }
   }
   printf(“\n The  sorted list of names  is :\n”);
   for(i=0; i<n; i++)
printf(“%s\n”,str[i]);
   getch();
}
```

This program allows the user to enter unspecified number of strings until the string END is entered. The program will count the strings as they are entered, ignoring the last string END. Here str is a two-dimensional character array.str[0],str[1], str[2] etc are pointers to 0th, 1st, 2nd etc rows of it, i.e they are pointers to 1st, 2nd, 3rd etc strings entered through keyboard. strcmp( ) function used at two places: while testing for a stopping condition, and in rearrange, to test the need for interchange of strings. The actual interchange of strings is done through strcpy( )function.

## 8.7.POINTER TO STRINGS

In section 8.1 we have seen that strings are one-dimensional arrays of type char and therefore they are declared and initialized as follows.

```
char str[10]=”GREEN”;
```

The compiler automatically append null character at the end of the string.

The following examples specify that how the pointers are used in string manipulation functions.

**Example 8.10:** Function to find length of a string using pointers

```
int str_len(char *p)
{
    char *q=p;
    while(*p != ’\0’)
        ++p;
    return(p-q);
}
```

The local pointer variable q is initialized with the address of the first character of the string. Since a string is terminated by the null character, the while loop

```
while(*p!=’\0’)

    ++p;
```

is executed until the end of the string is reached . When the while loop is terminated, the pointer p holds the address of the null character. Hence the expression (p-q) gives the length of the string.

**Example 8.11:** Function to copy one string to another string.

```
void str_cpy(char *ptr1, char *ptr2)
{
   while((*ptr1 = *ptr2) !='\0')
{
   ++ptr1;
   ++ptr2;
}
return;
}
```

The content of the second string represented by pointer ptr2 is copied in to the string represented by ptr1 until the null character is copied.

**Example 8.12:** Function to concatenate one string at the end of another string.

```
void  str_cat(char *ptr1, char *ptr2)
{
   while(*ptr1 !='\0')
      ptr1++;
   while((*ptr1 = *ptr2) != '\0')
  {
      ++ptr1;
      ++ptr2;
  }
return;
}
When the loop while(*ptr1!='\0')

      prt1++;
```

is executed, ptr1 points to the null character of the first string. When the second while loop is executed, the content of second string represented by the pointer ptr2 is appended to the first represented by ptr1 character by character until the null character is copied.

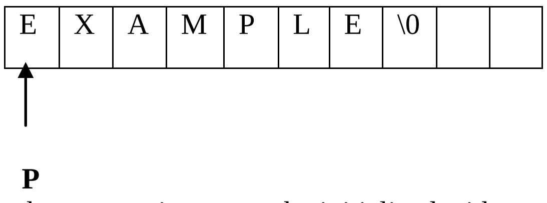
The pointer to charter is can be taken as a pointer to string

**Example 8.13:**

```
char *str[10]="EXAMPLE";

char *p=&str[0];
```

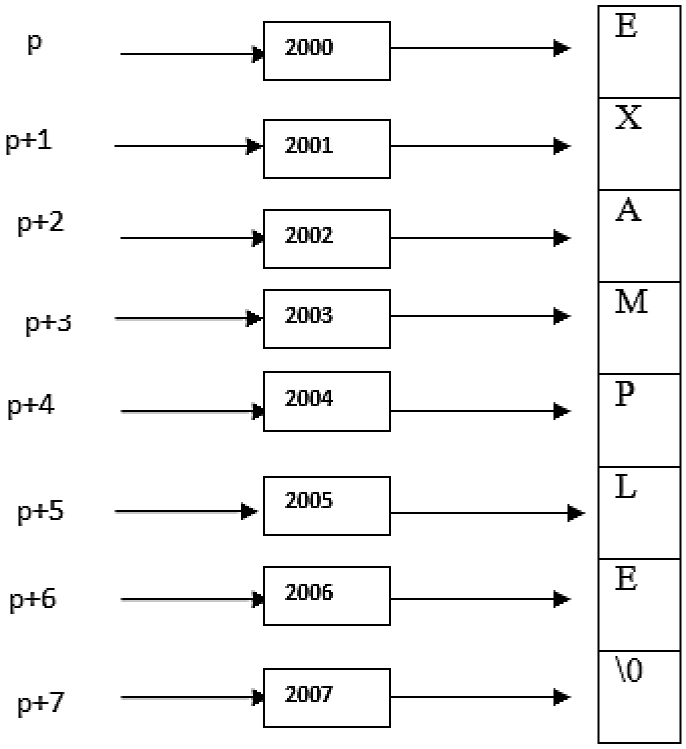
then 'p' is a pointer to the string.



A character pointer may be initialized with a string constant

```
Char *str = "EXAMPLE";
```

The address of the constant is stored in the pointer variable str . The pointer str now points to the first character of the string “EXAMPLE” as:



Where p represent the address 2000 of E and p+1 ,P+2,p+3, etc.. represent the addresses (2001,2002,2003 etc..) of the characters X,A,M etc.. We can also give values to the string pointer by assigning strings during runtime using scanf() with %s format or gets() function.

**Example 8.14:**

```
char *str;

scanf(“%s”,str);   (or)

gets(str);
```

p+1

The contents of the string str can be printed using either printf() or puts() functions.

```
printf(“%s”,str);  (or)

puts(str);
```

Here no need to use indirection operator * as str is a pointer

A constant character string always represents a pointer to that string Hence the following statements are valid.

```
char *ptr;

ptr = “ GREEN FIELDS”;
```

so the character string “GREEN FIELDS”; is assigned to the pointer to character str.

This type of assignment is not allowed to character array.

**Example8.15:**

```
char place[20];

place = “GREEN FIELDS”;
```

is not allowed.

**Example 8.16:** Function to compare two strings.

```
int str_cmp(char *ptr1, char *ptr2)
{
   while(( *ptr1 == *ptr2) && (*ptr1 !=’\0’) && (*ptr2 != ‘\0’))
   {
      ++ptr1;
      ++ptr2;
   }
return(*ptr1-*ptr2);
}
```

The comparison of the strings is done until there is a mismatch or one of the strings terminated in to a null character, whichever occurs first When the control exits from the loop the difference between the ASCII value of the first pair of non-matching characters is returned.

**Example 8.17:** Function to reverse a string**.**

```
void str_rev( char *sptr)
{
   char *eptr = sptr,*temp;
   while( *eptr != ‘\0’)
      ++eptr;
   for(--eptr; sptr <eptr; ++sptr,--eptr)
   {
       temp = *sptr;
       *sptr=*eptr;
       *eptr=temp;
   }
   return;
}
```

/*sptr is the pointer to first character and eptr is the pointer to the last character*/
When the loop

```
while(*eptr!=’\0’)
```

```
        ++eptr;
```

is executed eptr points to the null character .When it is decremented in the initial expression of the for loop it points to the last character of the string.Initially first and last characters are interchanged and then second and last but one characters are interchanged and so on . To access the next characters starting pointer str is incremented by 1 and the end pointer eptr is decremented by 1 in each repetition.

**Array of pointers to strings** :

A list of strings can be stored in the memory in another way. For each string, a memory block is allocated through malloc() function and the pointer to that block is stored in an array of pointers to strings . The declaration

```
  char *str[20];
```

would represent an array of pointers to strings.

The following program code can be used to allocate the memory block for each string and store the string in that block

```
 for(i=0;i<n;++i)

{

   str[i]=(char * ) malloc(15*size of (char));

   gets(str[i]);

}
```

The following program will illustrate the sorting a list of names using array of pointers.

**Program 8.10** Program to arrange a list of names in alphabetical order using array of pointers

```
#include<stdio.h>
#include<conio.h>
void main()
{
    char *str[20],*temp;
    int i,n;
    printf("Enter the number of names:");
    scanf("%d",&n);
    printf("Enter the names:\n");
    for(i=0;i<n;++i)
    {
        str[i]=(char*)malloc(15*sizeof(char));
        flushall();
        gets(str[i]);
    }
    for(p=1;p<n;++p)
        for(i=0;i<n-p;++i)
            if(strcmp(str[i],str[i+1]>0)
   {
       temp=str[i];
        str[i]=str[i+1];
```

```
      str[i+1]=temp;
    }
    printf("The list of names  in alphabetical order :\n");
    for(i=0;i<n;++i)
         puts(str[i]);
    getch();
}
```
Note that if the strings are not in alphabetical order the pointers to the strings are interchanged rather than the strings. This would result in the saving of execution time.

**SUMMARY**

- A single character is assigned to 0 character variable , that is stored in the memory allocated to a character variable .
- A string is a group of characters and is stored in a character array.
- Although C does not have a string data type , it allows string constant
- scanf() function with %s conversion, gets[] function, scan set inverted scan set, using getchar() function repeatedly can be used to input strings
- When stored in the memory the string will be terminated by the null character '\0' and is used to identify the end of the string.
- printf() function with %s conversion , puts() functions can be used for string output.
- There are several library functions in the header file <string.h> for manipulations of strings
- List of strings can be stored in a two dimensional character array.

**Suggested Reading:**

1. Chapter-12 : C for Engineers and Scientists by Harry H.Cheng.
2. Chapters-11 : Computer Science- A Structured Programming approach using C by B.A.Forouzan & Ritchard F.Gilberg.

**EXERCISES**

**Review Questions:**

8.1 Describe the limitations of getchar(), gets() and scanf() functions in reading string

8.2 Explain the feature of null character in the strings?

8.3 Can we use strings in switch statement?

8.4 How can we compare two strings?

8.5 _______ conversion specification is used in scanf() function to read a string.

8.6 _______ function does not require any conversion specification to read a string from the key board.

8.7 When reading a string with scanf() , terminating null character is automatically inserted(T/F).

8.8 When reading a string with scanf() using the conversion specification %s , stop reading when white space character is encountered (T/F)

8.9 While concatenating two strings the function strcat() insert a space between them (T/F)

8.10 ______ string manipulation function is used to determine whether a character is present in a string.

**Multiple choice questions**:

1. Which library function is used to append one string at the end of another string

   1) strcpy 2) strstr 3) strcat 4) strcmp

2. What is the output when the following code is executed ?

```
  #include<stdio.h>
  main()
  {
     char str[]= "VIJAYAWADA";
     int i=0;
     while(str[i]!='\0')
     {
         if(i%2==0)
    printf("%c",str[i]);++i;
     }
  }
```

a) 02468 b)VJYWD c)IAAAA d)13579

3) Which of the following is true ?

a) String must have at least one character

b) String is terminated in the memory by the character '\n'

c) String is a group of characters enclosed between single quotation marks

d) A string is stored in a one dimensional character array.

4) Which of the following initialization is invalid ?

a) char str[10]="HELLO" ; b) char str[5]="HELLO" ;

c) char str[6]={'H','E','L','L','O','\0'} d) char str[]="HELLO"

5) Which of the following is invalid string.

a) "356" b) "3" c) "3\0" d) "HONR"BLE"

6) What is the output when the following code is executed ?

```
char str[][10]={“BLUE”,”RED”,”ORANGE”,”GREEN”,”YELLOW”}

puts(str[2]);
```

a) RED b) ORANGE c) puts(str[2]) is invalid d)Invalid initialization

7) Which of the following code is invalid ?

a) char str[10],str2[10]=”HELLO”; b) char str[6],str2[10]=”HELLO”;

strcpy(str1,str2); strcpy(str1,str2);

c) char str[10],str2[10]=”HELLO”; d) char *str;

str1=str2**;** str=”HELLO”;

8) What will be the output of the following statement?

```
printf(“%d”,strcmp(“Rahul”,”Rajiv”));
```

a)2 b)0 c)-2 d)Given statement is not correct

9) Which of the following statements will correctly store the concatenation of the strings str1 and str2 in the str3

a) strcpy(str3,strcpy(str1,str2)); b) strcpy(str1,str2,str3);

c) strcpy(str3,strcat(str1,str2)); d) strcpy(strcat(str1,str2),str3);

10) What will be the output of the following code ?

```
char str[10]=”GREEN”;
int i=0;
while(str[i]!=’\0’)
{
    putchar(str[i]+1);
    ++i;
}
```

a) HSFFO b)5 c)FQDDM d)Invalid code

11) What will be the output of the following code ?

```
char str[10]= ”GREEN”;
 while(str[i]!=’\0’)
  {
  printf(“%d”,str[i]);
 ++i;
  }
```

a)70,8,68,68,77 b)67,69,69,75,81

c)71,82,69,69,78 d)none of these

**Comprehensive Questions**:

8.1 What are the different ways in which a single character and a string can be accepted and displayed

8.2 Write a program to read two strings and compare them using the function strcmp() and print a message that the first string is equal , lesser or grater than the second one

8.3 Develop a program to check given name is present in the given list or not.

8.4 Write a program to accept a list of strings and display the largest string.

8.5 Write a program to accept a string and a character and find out whether this character is present in the string.

8.6 Write a program to find length of string including and excluding spaces.

8.7 Write a program to copy source string to destination string up to a specified length. Length is to be entered through the keyboard.

8.8 Write a program to extract a substring of given length from a string starting from a given position.

8.9 Write a program to delete all occurrence of vowels in a given text assume that the text length will be of one line.

8.10 Write a program to encrypt the text "INDIA" the output should be "KPFKC" ('A' should be replaced with 'C', 'B' with 'D' and 'C' with 'E' and so on).

8.11 Write a program which will read a text and count all occurrences of a particular word.

8.12 Write a program which will read a string and rewrite it in alphabetical order. for example the word HELLO should be written as OLLEH.

8.13 Write a program to enter a string containing combination of capital, small, symbols and numerical characters. Carry out separation of capitals, small, symbols and numerical by using ASCII values from 48 to 122.

8.14 Write a program to display alphabets as given below .

```
az by cx dw ev fu gt hs ir jq kp lo mn nm ol pk qj ri sh tg uf ve wd xc yb za.
```

8.15 Given a string
char str[]="123456789";
write a program that displays the following:

```
        1
      2 3 2
    3 4 5 4 3
  4 5 6 7 6 5 4
5 6 7 8 9 8 7 6 5
```

8.16 Write a program to display the following pattern

```
        K
      K L
    K L U
  K L U B
K L U B  S
  K L U B
    K L U
      K L
        K
```

8.17 Write a program to convert each character of a string into the next character and print the same.

8.18 Write a function that takes a string and number between 0 and 9 as parameters and display the string that many times and returns its length.

8.19 Write a program to read a sentence and count the number of times each of the vowels appears in it, count the consonants separately.

8.20 Write a program to copy its input to its output,replacing each string of one or more blanks by a single blank

8.21 Write a program to copy its input to its output,replacing each tab by **\t** and each backslash by \\ .

8.22 Write a program to print all input lines that are longer than 80 characters.

8.23 Write a program to remove trailing blanks and tabs from each line of input and to delete entairly blank lines.

8.24 Write a function **reverse(s)** that reverses the character string s,use it to write a program that reverse input a line at a time.

8.25 Write a program detab that replaces tabs in the input with the proper number of blanks to space to the next tab stop. Assume a fixed set of tab stops , say every n columns.

8.26 Write a program to “fold” long input lines in to two or more shorter lines after the last non-blank character that occurs before the nth column of input .

8.27 Write a function **htoi(s)** which converts a string of hexadecimal digits(including an optional 0x or 0X) into its equivalent integer value.The allowable digits 0 through 9 ,a through f,and A through F.

8.28 Write a function that deletes each character in string s1 that matches any character in the string s2.