UNIT-5

Array

Array: Array is a user defined datatype. Array is group of elements, these elements are homogeneous. Each element shared common name elements are variable with their index value.

Syntax: datatype arrayname[size];

Use of arrays:

1. Storing more than one value at a time under a single name.
2. Reading, processing and displaying the array elements is easy.
3. Some logics can be implemented only through arrays.

Declaration of arrays:

Syntax: datatype arrayname[size];

Example: int a[10];

Declaration of arrays with datatype, arrayname and size of the array. Size denotes the maximum number of elements that can be stored in the array. Accessing elements (initialization of array elements)

Access the array elements in two ways:

1. Design time initialization
2. Run time initialization

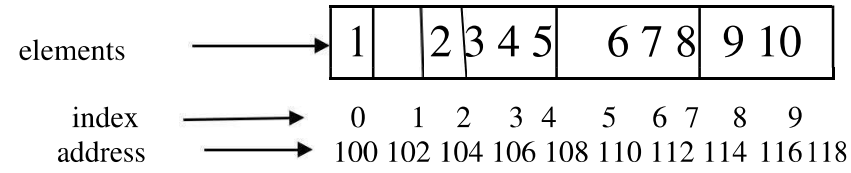
Design time accessing:

Declaration and initialization at a time.

Syntax: datatype arrayname[size]={list of values};

int a[10]={1,2,3,4,5,6,7,8,9,10};

Organization of the elements in memory



The index begin with 0(zero) and ends with one less than the size of the array.

Runtime accessing:

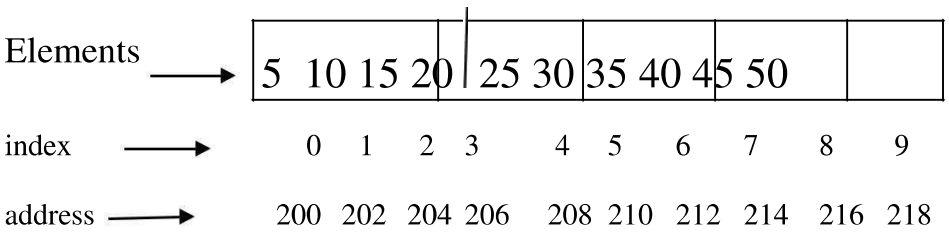
Read the array elements at time of execution from the keyboard using input statements.

Example:

```
int a[10];
for(i=0;i<=9;i++)
{
scanf("%d",&a[i]);
}
```

The above loop execute 10times and input the different array elements. For example entered elements are 5 10 15 20 25 30 35 40 45 50.

Organization of the elements in memory.



How to read and write the array elements.

```
#include<stdio.h>
main()
{
int a[10],i;
clrscr();
printf("Enter array elements \n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("Display the array elements \n");
for(i=0;i<10;i++)
{
printf("%d",a[i]);
}
getch();
}
```

Storing elements:

Declaration and definition only reserve space for the elements in the array. No values are stored. If we want to store values in the array, we must initialize the elements, read values from the keyboard or assign values to each individual element. int sample[5]={22,55,33,77,88};

Sample array have five space for store the five different elements. The elements are stored in array based on the index. For example 22 stored at sample[0], 55 stored at sample[1], 33 stored at sample[2], 77 stored at sample[3] and 88 stored at sample[4].

Types of arrays: There are three types‟ arrays.

1. One-dimensional array(1-D)
2. Two-dimensional array(2-D)
3. Multi dimensional array

One-dimensional array:

In one dimensional array, the organization of data is only one direction, because 1-D has only one dimension.

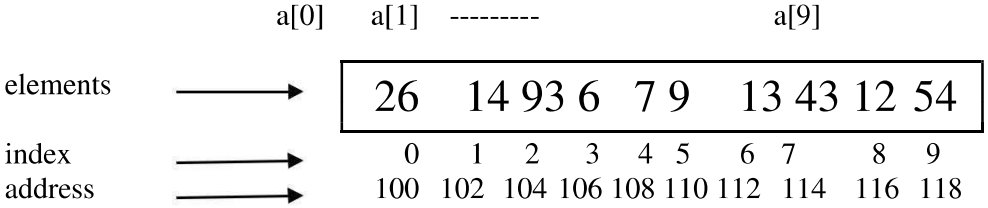
Example: datatype arrayname[size];

1-D integer array:

int a[10];

In the above statement array name „a‟ has 10 integer locations, „a‟ occupied memory is 20 bytes (2*10).

Organization of the elements in memory.



1-D character array(string):

A group of characters can be stored in a character array. 1-D character arrays are also called as strings

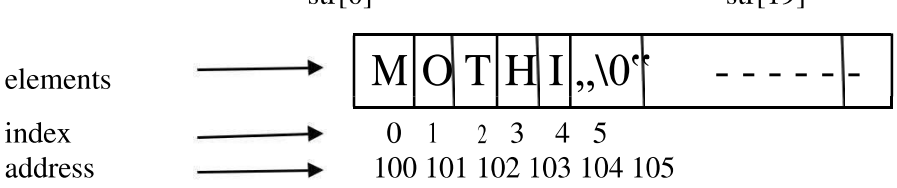
char str[20];

Above statement, arrayname str has 20 character elements. str occupied memory is 20 bytes(1*20).

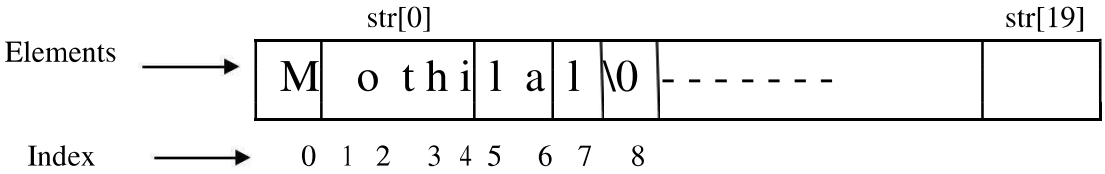
Example1: char str[20]={„v‟,‟i‟,‟j‟,‟a‟,‟y‟,‟a‟};

Example2: char str1[20]=”vijaya Prasad”;

Organization of elements in memory for str



Organization of elements in memory for str1



Declaration and initialization of 1-D:

Declaration:

Syntax: datatype arrayname[size];

Example: int a[10];

Initialization:

Designtime: datatype arrayname[size]={list of values}; int a[10]={1,2,3,4,5,6,7,8,9,20};

Runtime:

```
int a[5],i;
printf("Enter array elements");
for(i=0;i<5;i++)
{
scanf("%d",&a[i]);
```

```
}
```

Examples for 1-D:

Write a c-program to read and print array elements.

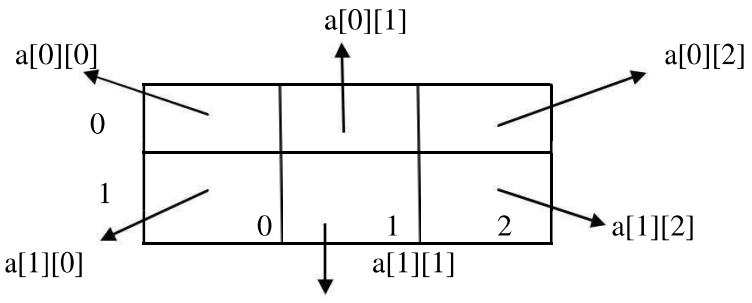
```
#include<stdio.h>
main()
{
int a[10],i;
clrscr();
printf("Enter array elements \n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
printf("Entered array elements are\n");
for(i=0;i<10;i++)
{
printf("4%d",a[i]);
}
getch();
}
```

Write c-program to read and print your name.

```
main()
{
char name[20]="vijaya";
clrscr();
printf("My name is : %s", name);
getch();
}
```

Two dimensional array(2-D):

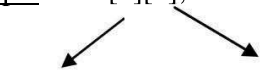
Many application require that data be stored in more than one dimension. One common example is a table. Which is an array that consists of rows and columns. Syntax: datatype arrayname[row-size][column-size]; Example: int a[2][3];



Declaration of 2-D:

Syntax: datatype arrayname[row-size][column-size];

Example: int a[2][3];



rows columns

Have 12 bytes(6*2 integers) gets allocated to „a" initializing a 2-D array.

At design time:

```
datatype arrayname[r-size][c-size]={list values};

int a[2][3]={5,10,15,20,25,30};
            Or
int a[2][3]={{5,10,15},{20,25,30}};
```

At runtime:

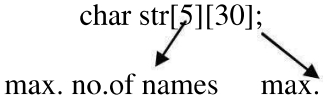
```
datatype arrayname[r-size][c-size];
int a[2][2],i,j;
for(i=o;i<2;i++)
{
for(j=0;j<3;j++)
{
scanf("%d",&a[i][j]);
}
}
```

2-D character arrays:

2-D chaarater arrays are nothing but an array(collection) of strings.

Declaration:

```
char str[5][30];
```

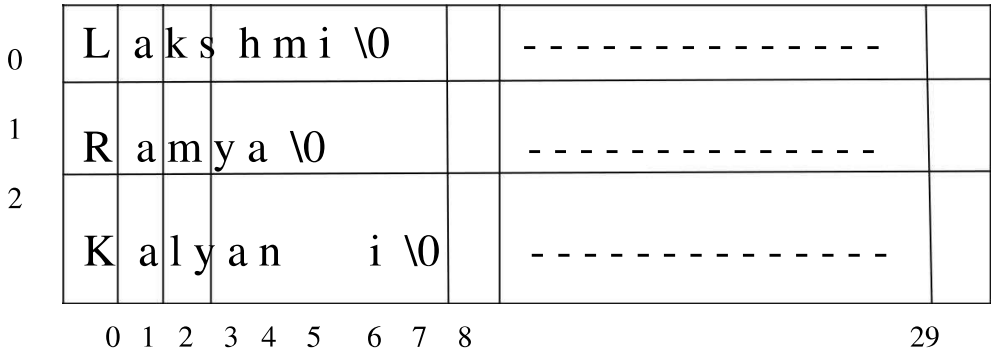


max. no.of names max. size of each name

Initialization:

```
char str[3][30]={"Lakshmi","Ramya","Kalyani"};
```

Stored on:



Examples on Two-dimensional array:

1. Read and print array elements using 2-D

```
main()
{
        int a[2][2],i,j;
        printf("Enter array elements
        \n"); for(i=0;i<2;i++)
        {
          for(j=0;j<2;j++)
          {
                scanf("%d",&a[i][j]);
          }
        }
        printf("Entered elements are \n");
        for(i=0;i<2;i++)
        {
          for(j=0;j<2;j++)
          {
            printf("4%d",a[i][j]);
          }
        }
getch();
}
```

→ Multidimensional arrays:

Multidimensional arrays can have three, four or more dimensions below diagram shows an array of three dimensions. Note the terminology used to describe the array. The first dimension is called a plane which consists of rows and columns. Arrays of four or more dimensions can be created and used but they are difficult to draw c language takes the three dimensional array to be an array of two dimensional arrays. It considers the two dimensional array to be an array of one dimensional arrays. In other words a three dimensional arrays in c is an array of arrays of arrays.

→ Declaring multidimensional arrays:

Declaration tells the compiler the name of the arrays, the type of each element and size of each dimension. The size of the fixed length array is a constant and must have a value at compilation time.

```
int table[planes][rows][cols];
```

→ Initialization:

Declaration and definition only reserve space for the elements in the array. No values will be stored in the array. If we want to store values, we must either initialize the elements, read values from the keyboard or assign values to each individual element.

```
int table[3][5][4]=
 {
   {{0,1,2,3},{10,11,12,13},{20,21,22,23},{30,31,32,33},{40,41,42,43}}
   {{100,101,102,103},{200,201,202,203},{300,301,302,303},
{400,401,402,403},{500,501,502,503} }
{{110,111,112,113},{210,211,212,213},{310,311,312,313},{410,411,412,413},
                        {510,511,512,513}}
                       };
```

1. Finding number of words in a sentence.

```
void main()
{
char str[128];
int i,count=0;
clrscr();
puts("Enter a sentence")
gets(str);
for(i=0;str[i]!=0;i++)
{
if(str[i]==" ")
count++;
}
printf("Number of words=%d",count+1);
}
```

2. sample

```
program main()
{
int a[10],i;
clrscr();
printf("Enter array elements
        \n"); for(i=0;i<2;i++)
{
scanf("%d",&a[i]);
}
printf("Elements\t Index \t Address \n");
for(i=0;i<2;i++)
{
printf("%d\t %d\t %d\t\n",a[i],I,&a[i]);
}
getch();
}
```

3.<u>Asending order:</u>

```
 #include<stdio.h>
main()
{
int i,j,k;
float n[10],t=0;
clrscr();
printf("Enter any 10 number:  \n");
for(i=0;i<10;i++)
{
scanf("%f",&n[i]);
```

```
}
for(i=0;i<10;i++)
{
for(j=i+1;j<10;j++)
{
if(n[i]>=n[j])
{
t=n[i];
n[i]=n[j];
n[j]=t;
}
}
}
printf("The asending order of the given number is : \n");
for(i=0;i<=9;i++)
printf("%7.5f",n[i]);
getch();
}
```

4. Write c-program to generate first 10 fibonacci numbers #include<stdio.h>

```
#include<conio.h>
     main()
{
int i,fib[10];;
clrscr();
fib[0]=0;
fib[1]=1;
          for(i=2;i<10;i++)
{
     fib[i]=fib[i-1]+fib[i-2];
     }
     for(i=0;i<2;i++)
     {
     printf("%d\n",fib[i]);
     }
     getch();
     }
```

5. Read array elements and print them into reverse order.

```
#include<stdio.h>
     main()
     {
     int a[10],i;
     clrscr();
     printf("Enter array elements
     \n"); for(i=0;i<10;i++)
     {
```

```
scanf("%d",&a[i]);
}
printf("The elements in reverse order\n");
for(i=9;i>=0;i--)
{
printf("%d",a[i]);
}
getch();
}
```

6. Largest and smallest.

```
main()
{
int number[10],i,large=0,small=32767;
clrscr();
printf("Enter array elements \n");
for(i=0;i<10;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<10;i++)
{
if(number[i]>large)
large=number[i];
}
for(i=0;I,=10;i++)
{
if(number[i]<small)
small=number[i];
}
printf("Largest value is %d",large);
printf("Smallest value is %d",small);
getch();
}
```

7.

```
#include<stdio.h>
#include< string.h>
    main()
    {
    char str[20];
    int i;
    clrscr();
    printf("Enter a string in uppercase");
    gets(str);
    for(i=0;str[i]!="\0")
    {
    if(str[i]>="A"&&str[i]<="2")
    str[i]=str[i]+32;
```

```
}
printf("The converted string is %s",str);
getch();
}
```

C – Strings:

A c-string is a variable length array of characters that is delimited by the null character.

Storing strings:

In c, a string is stored in an array of characters. It is terminated by the null characters („\O")

H E L L O \0

String delimiter:

Why do we need a null character at the end of a string?

Answer is that a string is not a data type but a data structure. This means that its implementation is logical, not physical.

Difference between the strings and character arrays

H E L L O \0 H E L L O

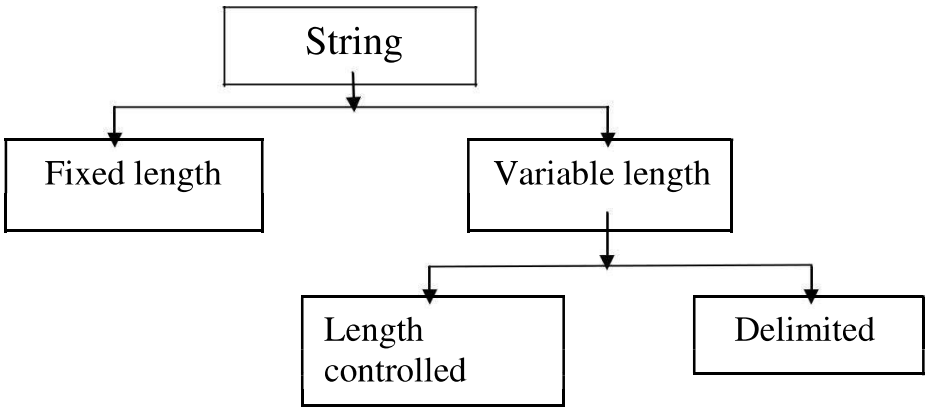
String literals: A string literal is a sequence of characters enclosed in double quotes.

Example:

"Have a nice day"
"Hello"
"Hai"
"Prasad"

Strings:

Strings concepts :



Fixed length strings: When implementing a fixed length string format the first decision is the size of the variable. If we make it too small, we can‟t store all the data. If we make it too big, we waste memory.

Example: 1. char str[15]=”Have a nice day”;
2. char str[100]=”Hai”;

Variable length strings: In variable length strings the user can use the required memory only. So reduce the memory wastage and data losses.

Example: char str[]=”good day”;

1. Length controlled strings: Length controlled strings add a count that specifies the number of characters in the string.
2. Delimited strings: Another techniques used to identify the end of the string is the delimiter at the ends of delimited strings.

Strings and characters: Data can be stored in two ways.
1. Data as character literal.
2. Data as string literal.

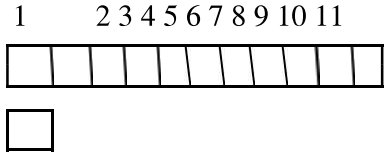
Data stored as character literal, we can use single quote marks.
Data stored as character literal, we can use double quote marks.

Example:

„a“	a as character
“a”	a as string
” ”	empty character
“ “	empty string

Declare strings: String declaration defines memory for a string when it is declared as an array in local memory.

Syntax: 1. char str[9];
Syntax: 2. string pointer declaration.
char *pstr;



Initializing strings: We can initialize a string the same way that we initialize any storage structure by assigning a value to it when it is defined.

```
char str[9]=”good day”;
char name[ ]=”Prasad”;
```
String initialized as a character pointer.
```
char *pstr=”good day”;
```
String initialized as an array of characters.
```
char str[9]={„g“,“o“,“o“,“d“,“ „„“d“,“a“,“y“};
char str[10]={„n“,“I“,“c“,“e“,“
```
„„“d“,“a“,“y“,“\O“}; Strings are end with null character.

String input and output functions:

C provide two basic ways to read and write strings. First, we can read and write strings with the formatted input/output functions.

scanf/fscanf and printf/fprintf

Second, we can use a special set of string only functions.

getstring → (gets/fgets)

putstring → (puts/fputs)

Formatted string input:

scanf(): scanf is the general input statement. It is used to read character data or numeric data from the keyboard.

Syntax: scanf("format specifier", variablename);

Example: char str[9];
scanf("%s",str);

Formatted string output:

printf(): printf is the general output statement. It is used to write character data or numeric data on the output units.

Syntax: printf("Format specifier",variablename);

Example: char str[9];
str="Hai";
printf("%s",str);

Example program:

```
main()
{
char str[9];
clrscr();
printf("Enter your string");
scanf("%s",str);
printf("Your string is %s",str);
getch();
}
```

String functions (input and output):

String input: gets()

The function gets accepts the name of string as a parameter, and fills the string with characters that are input from the keyboard, till a newline character is encountered, at the end, the function gets appends a null terminator as must be done for any string and returns. The new line character is not added to the string.

Syntax: gets(variable);

String output: puts()

This function is used to print the string message on to the output screen and also display the stored string in variable.

Syntax: puts("string message");
puts(variable);

Example on gets() and puts():

```
#include<string.h>
```

```
main()
{
char str[10];
clrscr();
puts("Enter your name");
gets(str);
puts("Entered name is ");
puts(str);
getch();
}
```

Output: Enter your name
Prasad
Entered name is
Prasad.

Character input and output functions:

Character input functions:

1. getchar()
2. getche()
3. getch()

Above three statements used for read a single character from keyboard at a time.

getchar(): getchar() function is used read a character from the keyboard and assigned the character to variable, the variable stored the character.

Syntax: variable=getchar ();

Example program:

```
#include<stdio.h>
main()
{
char c;
printf("Enter your character\n");
c=getchar();
printf("Entered character is \n");
putchar(c);
getch();
}
```

getch():

It is also read the characters from the keyboard and passes it immediately to the program with echoing on the screen.

Example program:

```
#include<stdio.h>
main()
{
char ch;
clrscr();
printf("enter a character\n");
ch=getch();
printf("enter character is %c",ch);
getch();
```

```
}
```

getche():

Accepts a character and passes it to the program and echos(displays) the same character on the screen also.

Example program:

```
#include<stdio.h>
main()
{
char ch;
printf("Enter your character \n");
ch=getche();
printf("Entered character is \n");
putchar(ch);
getch();
}
```

Character output function:

putchar(): putchar() is the character output function it is used to print the stored character in the variabale on to the screen.

Syntax: putchar(variable);

Example program:

```
#include<stdio.h>
main()
{
char ch;
clrscr();
printf("Enter your character  \n");
ch=getchar();
printf("Your character is \n");
putchar(ch);
getch();
}
```

Standard C string library functions:

Function	Description
strlen()	Determines length of a string
strcpy()	Copy the string from one to another
strncpy()	Copies characters of a string to another string upto specified length
strcmp()	Compares two strings
stricmp()	Compare two strings(function does not discriminates between small and capital letter)
strlwr()	Converts uppercase letters to lowercase letters
strupr()	Converts lowercase letters to uppercase letters
strdup()	Duplicate a string

strchr()	First character of the given string
strrchr()	Last character of the given string
strstr()	Determines first occurrence of the given string in another string
strcat()	String concatenation, means second string appends to the first string.
strncat()	Appends source string to destination string up to specified length.
strrev()	String reverse, reverse the all character of a string
strset()	Sets all characters of string with a given argument or symbol
strnset()	Sets specified numbers of characters of string with a given number or symbol
strspn()	Finds up to what length two strings are identical
strpbrk()	Searches the first occurrence of a character in a given string and then it displays the string starting from that character.

strlen(): This functions returns the number of character of a string data as an integer.

Syntax: strlen(variable);

Example program:

```
#include< string.h>
main()
{
char str[20];
int i;
clrscr();
puts("Enter your string");
gets(str);
i=strlen(str);
printf("Length of the given string is %d",i);
getch();
}
```

strcpy(): This function is used to copy the data of one string to another string variable.

Syntax: strcpy(stringvariable,stringvariable1);

The string in the second position is the source string that contains the data to be copied. The string in the first position is destination string that receives the data.

Example program:

```
#include< string.h>
main()
{
char str[20],str1[20];
clrscr();
puts("Enter ur string");
gets(str);
strcpy(str1,str);
puts(str);
puts(str1);
getch();
}
```

strcat(): This function is used concatenate two strings.

Syntax: strcat(stringvariable,stringvariable1);

The string in the first position is called destination string and the string in the second position is called source string the source is joined with the destination and the combined string is stored the destination itself.

Example program:

```
#include<string.h>
main()
{
char str1[20],str2[20];
clrscr();
puts("Enter both string1 and string2");
gets(str1);
gets(str2);
strcat(str1,str2);
puts(str1);
puts(str2);
getch();
}
```

strcmp(): This function is used to compare two strings.

Syntax: strcmp(stringvariable1,stringvariable2);

The strcmp() functionreturn an integer value as the result.

1.result +ve : String variable1 is greater than string variable2.
2.result –ve : String1is lessthan string2.
3.result is zero : Both strings are equal.

Example program:

```
#include< string.h>
main()
{
int i;
char str1[20],str2[20];
clrscr();
puts("Enter both strings");
gets(str1);
gets(str2);
i=strcmp(str1,str2);
printf("Difference is %d",i);
getch();
}
```

strrev(): This function is used to get the reverse string of a given string.

Syntax: strrev(string);

Example program:

```
#include< string.h>
```

```
main()
{
char str[20];
clrscr();
puts("Enter a string");
gets(str);
strrev(str);
puts("Reverse string of the given string is");
puts(str);
getch();
}
```

strlwr(): This function is used to convert the uppercase string to lower case string.

Syntax: strlwr(string);

Example program:

```
#include< string.h>
main()
{
char str[20];
clrscr();
puts("Enter the string in uppercase");
gets(str);
strlwr(str);
puts(str);
getch();
}
```

strupr(): This function is used to convert the lowercase string to uppercase string.

Syntax: strupr(string);

Example program:

```
#include< string.h>
main()
{
char str[20];
clrscr();
puts("Enter the string in lowercase");
gets(str);
strlwr(str);
puts(str);
getch();
}
```

Sample programs on Strings:

1. To check the given string is palindrome or not.

```
#include< string.h>
main()
```

```
{
char str[20]str1[20];
clrscr();
puts("Enter a string ");
gets(str);
strcpy(str1,str);
strrev(str);
i=strcmp(str,str1);
if(i==0)
puts("Given string is palindrome");
else
puts("Given string is not palindrome");
getch();
}
```

2. Find a word in a string using string function.

```
#include< string.h>
main()
{
char str[20]str1[20];
clrscr();
Ppts("Enter a string ");
gets(str);
puts("Enter find string");
gets(str1);
if(strstr(str,str1))
puts("Yes it is in the string ");
else
puts("Not found");
getch();
}
```