

UNIT II ARRAYS AND STRINGS

Introduction to Arrays: Declaration, Initialization – One dimensional array – Example Program: Computing Mean, Median and Mode - Two dimensional arrays – Example Program: Matrix Operations (Addition, Scaling, Determinant and Transpose) - String operations: length, compare, concatenate, copy – Selection sort, linear and binary search

2.1 Arrays**Definition:**

An array is a data structure that is used to store data of the same type. The position of an element is specified with an integer value known as **index** or **subscript**.

E.g.

1	3	5	2
---	---	---	---

a(integer array)

1.2	3.5	5.4	2.1
-----	-----	-----	-----

b(float array)

[0] [1] [2] [3]

Characteristics:

- All the elements of an array share a common name called as array name
- The individual elements of an array are referred based on their position.
- The array index in c starts with 0.

In general arrays are classified as:

- Single dimensional array
- Multi-dimensional array

2.2 Single or one dimensional array

- It is also known as one-dimensional arrays or linear array or vectors
- It consists of fixed number of elements of same type
- Elements can be accessed by using a single subscript. eg) a[2]=9;

Eg)

a	1	3	5	2
	[0]	[1]	[2]	[3]

← subscripts or indices

Declaration of Single Dimensional Array**Syntax:**

datatype arrayname [array size];

E.g. int a[4]; // a is an array of 4 integers
 char b[6]; //b is an array of 6 characters

Initialization of single dimensional array

Elements of an array can also be initialized.

Rules

a) Elements of an array can be initialized by using an initialization list. An initialization list is a comma separated list of initializers enclosed within braces.

Eg) `int a[3]={1,3,4};`

b) If the number of initializers in the list is less than array size, the leading array locations gets initialized with the given values. The rest of the array locations gets initialized to

0 - for int array
0.0 - for float array
\0 - for character array

Eg) `int a[2]={1};`

a

1	0
---	---

`char b[5]={‘A’.’r’,’r’};`

b

‘A’	‘r’	‘r’	‘\0’	‘\0’
-----	-----	-----	------	------

Usage of single dimensional array

The elements of single dimensional array can be accessed by using a subscript operator([]) and a subscript.

Reading storing and accessing elements:

An iteration statement (i.e loop) is used for storing and reading elements.

Ex:1 Program to calculate the average marks of the class

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int m[5],i,sum=0,n;
```

```
    float avg;
```

```
    printf("enter number of students \n");
```

```
    scanf("%d",&n);
```

```
    printf("enter marks of students \n");
```

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

```
    {
```

```
        scanf("%d",&m[i]);
```

```
    }
```

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

```
    sum=sum+m[i];
```

```
    avg=(float)sum/n;
```

```
    printf("average=%f",avg);
```

```
}
```

Output:

Enter number of students

5

Enter marks of students

55

60

78

85

90

Average=73.6

2.3 EXAMPLE PROGRAMS:

- 1.Computing Mean
- 2.Computing Median
- 3.Computing Mode

Computing Mean:

* C Program to find the mean of n numbers using array *\

#include <stdio.h>

void main()

```
{
    int m[5],i,sum=0,n;
    float mean;
    printf("enter number of students \n");
    scanf("%d",&n);
    printf("enter marks of students \n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&m[i]);
    }
    for(i=0;i<n;i++)
    sum=sum+m[i];
    mean=(float)sum/n;
    printf("Mean=%f",mean);
}
```

Output:

Enter number of students

5

Enter marks of students

55

60

78

85

90

Mean=73.6

Computing Median:

#include<stdio.h>

#include<conio.h>

main()

```
{
    int i,j,temp,n,a[20],sum=0;
    float median;
    printf("enter n:");
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
{
    printf("\n enter %d number:",i+1);
    scanf("%d",&a[i]);
}
for(i=0;i<n-1;i++)
{
    for(j=i+1;j<n;j++)
    {
        if(a[j]<a[i])
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
        }
    }
}
if(n%2==0)
{
    median=((a[n/2]+a[n/2 -1])/2.0);
}
else
{
    median=a[n/2];
}
printf("\n the median value is %f",median);
getch();
}
```

Output:

```
Enter N:5
Enter 1 number:11
Enter 2 number:12
Enter 3 number:13
Enter 4 number:14
Enter 5 number:15
The median value is 13.000000
```

3.Computing Mode:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int maxvalue = 0, maxCount = 0, i, j,a[20],n,count=0;
    clrscr();
    printf("enter the N value:");
    scanf("%d",&n);
    for(i = 0; i < n;i++)
```

```

{
printf("\n Enter %d number:",i+1);
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
for (j = i+1; j < n;j++)
{
if(a[j] == a[i])
count++;
}
if (count > maxCount)
{
maxCount = count;
maxvalue = a[i];
printf("\nThe mode value is %d",maxvalue);
}
}
getch();
}

```

Output:

Enter the N value:5

Enter 1 Number:0

Enter 2 Number:6

Enter 3 Number:7

Enter 4 Number:2

Enter 5 Number:7

The mode value is 7

2.4 Two dimensional arrays

- A 2D array is an array of 1-D arrays and can be visualized as a plane that has rows and columns.
- The elements can be accessed by using two subscripts, row subscript (row no), column subscript(column no).
- It is also known as matrix.

E.g,

	1	2	3	6	7
	9	10	5	0	4
a[3][5]	3	1	2	1	6

Declaration

datatype arrayname [row size][column size]
--

e.g) int a [2][3];

//a is an integer array of 2 rows and 3 columns

number of elements=2*3=6

Initialization

1. By using an initialization list, 2D array can be initialized.

e.g. `int a[2][3] = {1,4,6,2}`

a

1	4	6
2	0	0

2. The initializers in the list can be braced row wise.

e.g. `int a[2][3] = { {1,4,6} , {2} };`

Example: Matrix Addition

```
#include<stdio.h>
#include<conio.h>
main()
{
int a[25][25],b[25][25], c[25][25], i, j m, n;
clrscr();
printf("\n Enter the rows and columns of two matrices... ");
scanf("%d %d", &m, &n)
printf{"\n Enter the elements of A matrix..."};
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
printf{"\n Enter the elements of B matrix..."};
for(i=0;i<m;i++)
for(j=0;j<n;j++)
scanf("%d", &b[i][j]);
for(i=0;i<m;i++)
for(j=0;j<n;j++)
c[i][j]=a[i][j] + b[i][j];
printf("\n The addition of two matrices");
for(i=0;i<m;i++)
{
printf("\n");
for(j=0;j<n;j++)
{
printf("\t %d",c[i][j]);
}
}
getch();
}
```

Output:

Enter the rows and columns of two matrices.... 3 3

Enter the elements of A matrix... 1 2 3 4 5 6 7 8 9

Enter the elements of B matrix... 1 2 3 4 5 6 7 8 9
The addition of two matrixes
2 4 6
8 10 12
14 16 18

Example: Matrix Multiplication

```
#include <stdio.h>
#include <conio.h>
main()
{
int a[10][10], b[10][10], c[10][10];
int r1, c1, r2, c2;
int i, j, k;
clrscr();
printf("Enter order of matrix A : ");
scanf("%d%d", &r1, &c1);
printf("Enter order of matrix B : ");
scanf("%d%d", &r2, &c2);
if (c1 != r2)
{
printf("Matrix multiplication not possible");
getch();
exit(0);
}
printf("Enter matrix A elements\n");
for(i=0; i<r1; i++)
for(j=0; j<c1; j++)
scanf("%d", &a[i][j]);
printf("Enter matrix B elements\n");
for(i=0; i<r2; i++)
for(j=0; j<c2; j++)
scanf("%d", &b[i][j]);
for(i=0; i<r1; i++)
{
for(j=0; j<c2; j++)
{
c[i][j] = 0;
for(k=0; k<c1; k++)
{
c[i][j] = c[i][j] + a[i][k] * b[k][j];
}
}
}
printf("Product matrix C\n");
for(i=0; i<r1; i++)
{
```

```
for(j=0; j<c2; j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
getch();
}
```

Output

```
Enter order of matrix A : 2 3
Enter order of matrix B : 3 2
Enter matrix A elements
1 1 1
1 1 1
Enter matrix B elements
2 2
2 2
2 2
Product matrix C
6 6
6 6
```

Example: Matrix Scaling

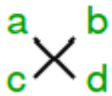
```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int graphdriver=DETECT,graphmode,errorcode;
int i;
int x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
line(x1,y1,x2,y2);
printf("Enter scaling co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=(x1*x);
y1=(y1*y);
x2=(x2*x);
y2=(y2*y);
printf("Line after scaling");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```


}

Example: Matrix Determinant**C code for Determinant of 2X2 matrix:****Determinant of 2 x 2 Matrix:**

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$
$$|A| = ad - bc$$

Remember by this -


$$\begin{matrix} a & b \\ c & d \end{matrix}$$

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[2][2],i,j;
    long determinant;
    clrscr();
    printf("Enter the 4 elements of matrix: ");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
            scanf("%d",&a[i][j]);
    }
    printf("\nThe matrix is\n");
    {
        for(i=0;i<2;i++)
        {
            printf("\n");
            for(j=0;j<2;j++)
                printf("%d\t",a[i][j]);
        }
    }
    determinant = a[0][0]*a[1][1] - a[1][0]*a[0][1];
    printf("\nDeterminant of 2X2 matrix: %ld",determinant);
```

```
getch();
}
```

Output:

Enter the 4 elements of matrix 4 8 3 9

4 8

3 9

Determinant of 2x 2 matrix : 12

C code for Determinant of 3X3 matrix:

Determinant of 3 x 3 Matrix:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$|A| = a(ei - fh) - b(di - gf) + c(dh - eg)$$

In terms of Cofactor:

$$\begin{bmatrix} a & b & c \\ \left| \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right| & - & \left| \begin{smallmatrix} d & f \\ g & i \end{smallmatrix} \right| & + & \left| \begin{smallmatrix} d & e \\ g & h \end{smallmatrix} \right| \end{bmatrix}$$

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],i,j;
long determinant;
clrscr();
printf("Enter the 9 elements of matrix: ");
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
scanf("%d",&a[i][j]);
}
printf("\nThe matrix is\n");
for(i=0;i<3;i++)
{
printf("\n");
for(j=0;j<3;j++)
printf("%d\t",a[i][j]);
}
```

```
determinant = a[0][0]*((a[1][1]*a[2][2]) - (a[2][1]*a[1][2])) -a[0][1]*(a[1][0]*a[2][2] -  
a[2][0]*a[1][2]) + a[0][2]*(a[1][0]*a[2][1] - a[2][0]*a[1][1]);  
printf("\n Determinant of 3X3 matrix: %ld", determinant);  
getch();  
}
```

Output:

Enter the 9 elements of matrix: 1 2 3 4 5 6 7 8 9

1 2 3

4 5 6

7 8 9

Determinant of 3X3 matrix:0

Example: Matrix Transpose

```
#include<stdio.h>  
#include<conio.h>  
void main()  
{  
int a[5][5],i,j,m,n;  
clrscr();  
printf("How many rows");  
scanf("%d",&n);  
printf("How many columns");  
scanf("%d",&m);  
printf("\nEnter the matrix:\n");  
for(i=0;i<m;i++)  
for(j=0;j<n;j++)  
scanf("%d",&a[i][j]);  
for(i=0;i<m;i++)  
{  
for(j=0;j<n;j++)  
printf("%d",a[i][j]);  
printf("\n");  
}  
printf("\nTranspose of given matrix:\n");  
for(i=0;i<m;++i)  
{  
for(j=0;j<n;++j)  
printf("%d ",a[j][i]);  
printf("\n");  
}  
getch();  
}
```

Output:

How many rows 2

How many columns 2

Enter the matrix: 1 2 3 4

1 2
3 4

Transpose of given matrix:

1 3
2 4

2.5 String Operations:(Length, Compare, Concatenate, Copy):

Definition:

The group of characters, digits, & symbols enclosed within double quotes is called as Strings. Every string is terminated with the NULL ('\0') character.

E.g. "INDIA" is a string. Each character of string occupies 1 byte of memory. The last character is always '\0'.

Declaration:

String is always declared as character arrays.

Syntax

```
char stringname[size];
```

E.g. char a[20];

Initialization:

We can use 2 ways for initializing.

1. By using string constant

E.g. char str[6]= "Hello";

2. By using initialisation list

E.g. char str[6]={ 'H', 'e', 'l', 'l', 'o', '\0' };

String Operations or String Functions

These functions are defined in **string.h** header file.

1. strlen() function

It is used to find the length of a string. The terminating character ('\0') is not counted.

Syntax

```
temp_variable = strlen(string_name);
```

E.g.

s= "hai";

strlen(s)-> returns the length of string s i.e. 3.

2. strcpy() function

It copies the source string to the destination string

Syntax

```
strcpy(destination,source);
```

E.g.
 s1="hai";
 s2= "welcome";
 strcpy(s1,s2); -> s2 is copied to s1. i.e. s1=welcome.

3. strcat() function

It concatenates a second string to the end of the first string.

Syntax

```
strcat(firststring, secondstring);
```

E.g.
 s1="hai ";
 s2= "welcome";
 strcat(s1,s2); -> s2 is joined with s1. Now s1 is hai welcome.

E.g. Program:

```
#include <stdio.h>
#include <string.h>
void main ()
{
    char str1[20] = "Hello";
    char str2[20] = "World";
    char str3[20];
    int len ;
    strcpy(str3, str1);
    printf("Copied String= %s\n", str3 );
    strcat( str1, str2);
    printf("Concatenated String is= %s\n", str1 );
    len = strlen(str1);
    printf("Length of string str1 is= %d\n", len );
    return 0;
}
```

Output:

```
Copied String=Hello
Concatenated String is=HelloWorld
Length of string str1is 10
```

4. strcmp() function

It is used to compare 2 strings.

Syntax

```
temp_variable=strcmp(string1,string2)
```

- If the first string is greater than the second string a positive number is returned.
- If the first string is less than the second string a negative number is returned.
- If the first and the second string are equal 0 is returned.

5. strlwr() function

It converts all the uppercase characters in that string to lowercase characters.

Syntax

```
strlwr(string_name);
```

E.g.

```
str[10]= "HELLO";
strlwr(str);
puts(str);
```

Output: hello

6. strupr() function

It converts all the lowercase characters in that string to uppercase characters.

Syntax

```
strupr(string_name);
```

E.g.

```
str[10]= "HEllo";
strupr(str);
puts(str);
```

Output: HELLO

7. strrev() function

It is used to reverse the string.

Syntax

```
strrev(string_name);
```

E.g.

```
str[10]= "HELLO";
strrev(str);
puts(str);
```

Output: OLLEH

String functions

Functions	Descriptions
strlen()	Determines the length of a String
strcpy()	Copies a String from source to destination
strcmp()	Compares two strings
strlwr()	Converts uppercase characters to lowercase
strupr()	Converts lowercase characters to uppercase
strdup()	Duplicates a String
strstr()	Determines the first occurrence of a given String in another string
strcat()	Appends source string to destination string
strrev()	Reverses all characters of a string

Example: String Comparison

```
void main()
{
char s1[20],s2[20];
int val;
printf("Enter String 1\n");
gets(s1);
printf("Enter String 2\n");
gets (s2);
val=strcmp(s1,s2);
if (val==0)
printf("Two Strings are equal");
else
printf("Two Strings are not equal");
getch();
}
```

Output:

```
Enter String 1
Computer
Enter String 2
Programming
Two Strings are not equal
```

String Arrays

They are used to store multiple strings. 2-D char array is used for string arrays.

Declaration

<code>char arrayname[rowsize][colsize];</code>
--

E.g.

```
char s[2][30];
```

Here, s can store 2 strings of maximum 30 characters each.

Initialization

2 ways

1. Using string constants

```
char s[2][20]={ "Ram", "Sam"};
```
2. Using initialization list.

```
char s[2][20]={ {'R', 'a', 'm', '\0'},  
                {'S', 'a', 'm', '\0'}};
```

E.g. Program

```
#include<stdio.h>
void main()
{
int i;
char s[3][20];
printf("Enter Names\n");
for(i=0;i<3;i++)
```

```
scanf("%s", s[i]);  
printf("Student Names\n");  
for(i=0;i<3;i++)  
printf("%s", s[i]);  
}
```

2.6 Sorting

Sorting is the process of arranging elements either in ascending or in descending order.

Sorting Methods

1. Selection Sort
2. Bubble Sort
3. Merge sort
4. Quick sort

1. Selection sort

It finds the smallest element in the list & swaps it with the element present at the head of the list.

E.g.

25 20 15 10 5

5 20 15 10 25

5 10 15 20 25

2. Bubble Sort

In this method, each data item is compared with its neighbour element. If they are not in order, elements are exchanged.

With each pass, the largest of the list is "bubbled" to the end of the list.

E.g.

Pass 1:

25 20 15 10 5

20 25 15 10 5

20 15 25 10 5

20 15 10 25 5

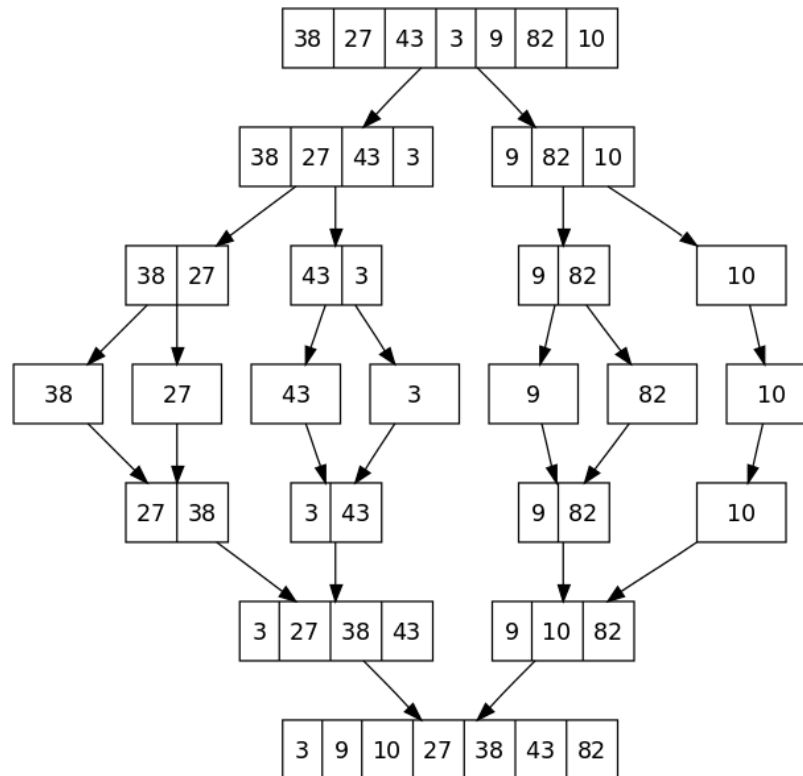
20 15 10 5 25

25 is the largest element

Repeat same steps until the list is sorted

3. Merge Sort:

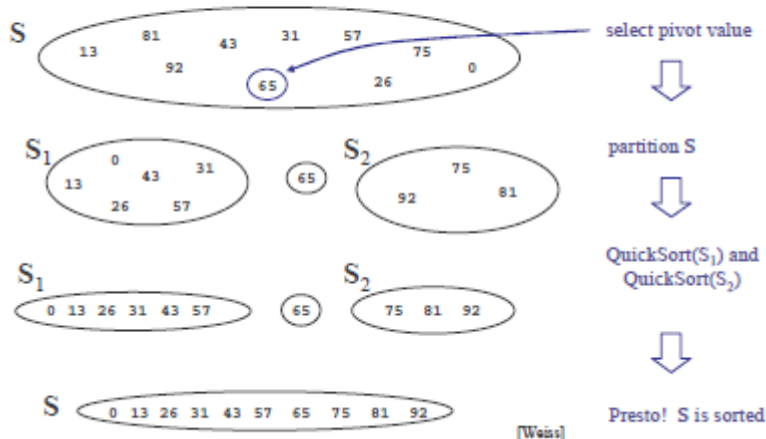
- Merge sort is based on Divide and conquer method.
- It takes the list to be sorted and divide it in half to create two unsorted lists.
- The two unsorted lists are then sorted and merged to get a sorted list.



4. Quick Sort

- This method also uses the technique of ‘divide and conquer’.
- Pivot element is selected from the list, it partitions the rest of the list into two parts – a sub-list that contains elements less than the pivot and other sub-list containing elements greater than the pivot.
- The pivot is inserted between the two sub-lists. The algorithm is recursively applied to sort the elements.

The steps of QuickSort



Program:

```
#include <stdio.h>
void main()
```

```
{
    int i, j, temp, n, a[10];
    printf("Enter the value of N \n");
    scanf("%d", &n);
    printf("Enter the numbers \n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (a[i] > a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    printf("The numbers arranged in ascending order are given below \n");
    for (i = 0; i < n; i++)
        printf("%d\n", a[i]);
    printf("The numbers arranged in descending order are given below \n");
    for(i=n-1;i>=0;i--)
        printf("%d\n",a[i]);
}
```

Output:

Enter the value of N

4

Enter the numbers

10 2 5 3

The numbers arranged in ascending order are given below

2

3

5

10

The numbers arranged in descending order are given below

10

5

3

2

2.7 Searching

Searching is an operation in which a given list is searched for a particular value. If the value is found its position is returned.

Types:

1. Linear Search
2. Binary Search

1. Linear Search

The search is linear. The search starts from the first element & continues in a sequential fashion till the end of the list is reached. It is slower method.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[10],i,n,m,c=0;
    clrscr();
    printf("Enter the size of an array: ");
    scanf("%d",&n);
    printf("Enter the elements of the array: ");
    for(i=0;i<=n-1;i++)
        scanf("%d",&a[i]);
    printf("Enter the number to be searched: ");
    scanf("%d",&m);
    for(i=0;i<=n-1;i++)
    {
        if(a[i]==m)
        {
            printf("Element is in the position %d\n",i+1);
            c=1;
            break;
        }
    }
    if(c==0)
        printf("The number is not in the list");
    getch();
}
```

Output:

```
Enter the size of an array: 4
Enter the elements of the array: 4 3 5 1
Enter the number to be search: 5
Element is in the position 3
```

2. Binary Search

- If a list is already sorted then we can easily find the element using binary search.
- It uses divide and conquer technique.

Steps:

1. The middle element is tested with searching element. If found, its position is returned.
2. Else, if searching element is less than middle element, search the left half else search the right half.
3. Repeat step 1 & 2.

Program:

```
#include<stdio.h>
```

```
void main()
{
    int a[10],i,n,m,c=0,l,u,mid;
    printf("Enter the size of an array: ");
    scanf("%d",&n);
    printf("Enter the elements in ascending order: ");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("Enter the number to be searched: ");
    scanf("%d",&m);
    l=0,u=n-1;
    while(l<=u)
    {
        mid=(l+u)/2;
        if(m==a[mid])
        {
            c=1;
            break;
        }
        else if(m<a[mid])
        {
            u=mid-1;
        }
        else
            l=mid+1;
    }
    if(c==0)
        printf("The number is not found.");
    else
        printf("The number is found.");
}
```

Sample output:

Enter the size of an array: 5

Enter the elements in ascending order: 4 7 8 11 21

Enter the number to be search: 11

The number is found.

Example:

3 5 7 9 11

Search key=7 middle element=7

Searching element=middle element. So the element is found.

Search key=11

Middle element=7

Searching element>middle

So go to right half: 9 11. Repeat steps until 11 is found or list ends.