MODULE-3

ARRAYS AND STRINGS

Arrays

Contents

Arrays

- An array is a collection of elements of the same type that are referenced by a common name.

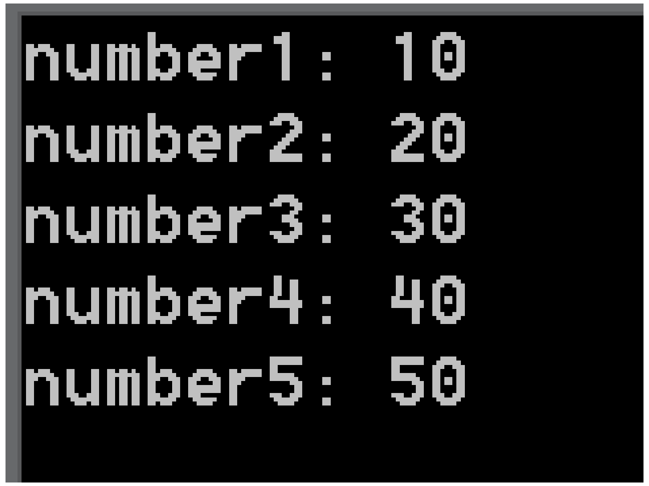
- Arrays are the data types in C which can store the primitive data type of data such as int, char, double, float.

- Consider a situation, where we need to store 5 integer numbers.

Arrays

```
 #include<stdio.h>
int main()
{
int number1=10;
int number2=20;
int number3=30;
int number4=40;
int number5=50;
 printf( "number1: %d \n", number1);
printf( "number2: %d \n", number2);
printf( "number3: %d \n", number3);
printf( "number4: %d \n", number4);
printf( "number5: %d ", number5);
Return 0;
}
```

```
int number1=10, number2=20, number3=30, number4=40,
number5=50;
```

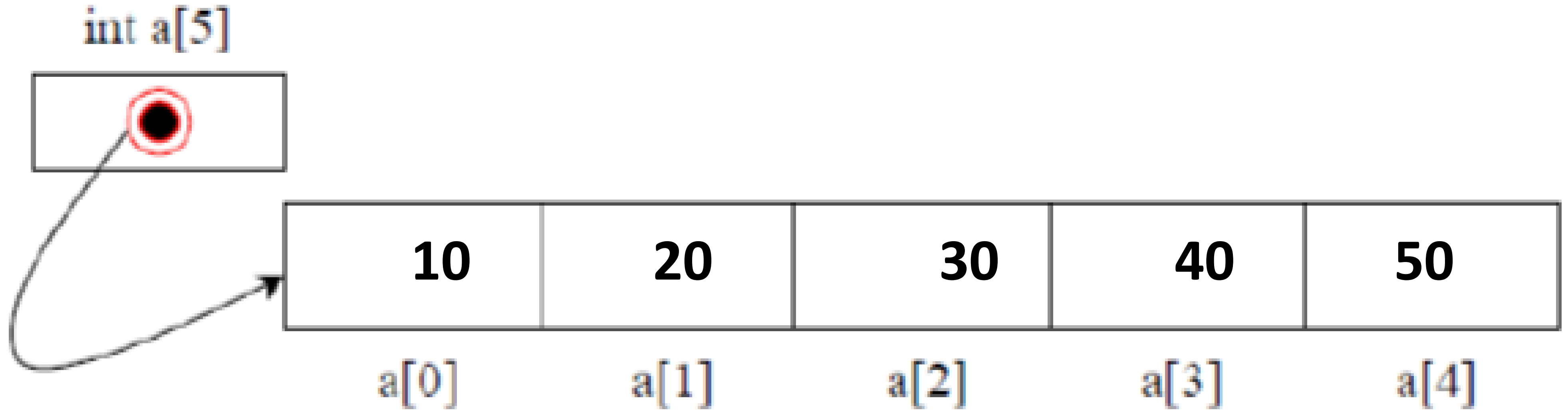


```
number1: 10
number2: 20
number3: 30
number4: 40
number5: 50
```

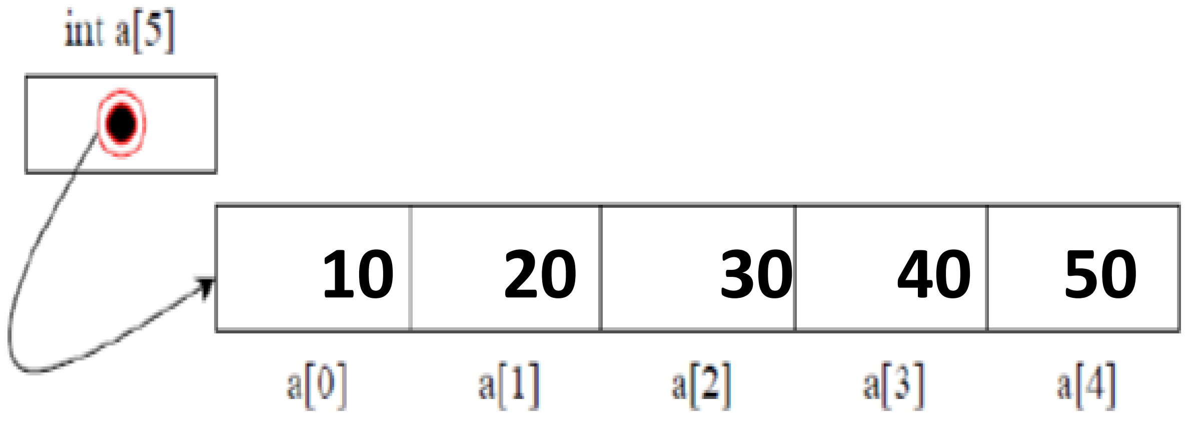
Arrays

- To handle such situation, C language provides a concept called the **ARRAY.**

```
int a[5]={10,20,30,40,50};
```



Arrays



- a[0] holds the first element in the array
- a[1] holds second element in the array
- a[2] holds third element in the array
- a[3] holds fourth element in the array
- a[4] holds fifth element in the array

Arrays

- The elements are stored sequentially one after the other in memory.
- Any element can be accessed by using

1. name of the array
2. position of element in the array (index)

- **DEFINITION OF ARRAYS**

 The array is a fixed-size sequenced collection of elements of same data type.

 The array is a collection of homogeneous elements of same data type.

 Array groups the data items of similar types sharing the common name.

Types of Arrays

- Single dimensional array or One dimensional array
- Two dimensional array

Single Dimensional Array :-

- An Array which has only one subscript is known as Single dimensional array or

One dimensional array.

- The individual array elements are processed by using a common array name with different index values that start with **Zero** and ends with **array_size-1**.

Syntax:

data_type array_name[array_size];

• Single Dimensional Array :-

Syntax:

data_type array_name[array_size];

Where,

data_type: can be int, float or char

array_name: is name of the array

array_size : an integer constant indicating the maximum number of data elements to be stored.

• Single Dimensional Array :-

• **Rules for declaring Single Dimensional Array :-**

1. An array variable must be declared before being used in a program
2. The declaration must have a data type(int, float, char), variable name and subscript.
3. The subscript represents the size of the array. If the size is declared as 10, programmers can store 10 elements.
4. An array index always starts from **0**.
5. **Example:** if an array variable is declared as **a[10]**, then it ranges from **0 to 9**.
6. Each array element stored in a separate location.

• Single Dimensional Array :-

Memory occupied by 1D array

Total memory=array size * size of datatype

For example : **int a[5];**

Total memory =5*sizeof (int)

= 5*4

=20 bytes.

- Single Dimensional Array :-

Example: Integer array example:

int age[5];

int age[5]={0, 1, 2, 3, 4};

age[0]; /*0 is accessed*/

age[1]; /*1 is accessed*/

age[2]; /*2 is accessed*/

Example: Character array example:

char str[10];

char str[10]={‘H’,‘a’,‘i’}; (or)

char str[0] = ‘H’;

char str[1] = ‘a’;

char str[2] = ‘i;

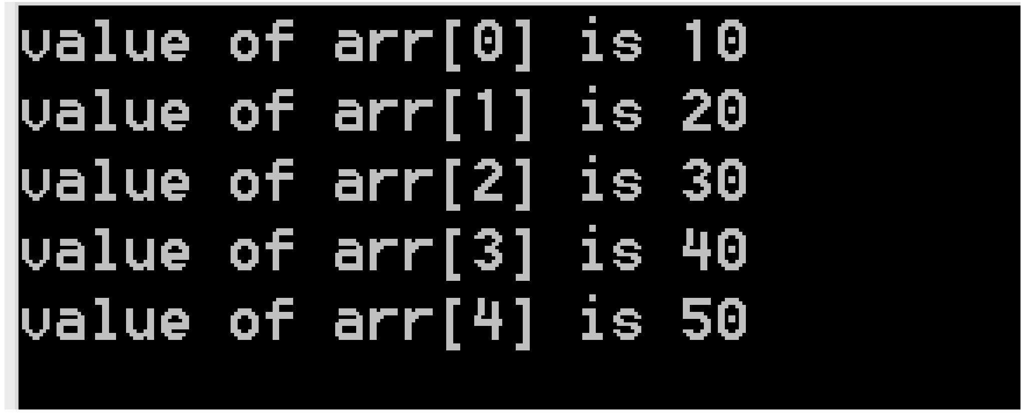
str[0]; /*H is accessed*/

str[1]; /*a is accessed*/

str[2]; /*i is accessed*/

- Single Dimensional Array :-

```
#include<stdio.h>
int main()
{
   int i;
   int arr[5] = {10,20,30,40,50};
for(i=0;i<5;i++)
   {
      // Accessing each variable
      printf("value of arr[%d] is %d \n", i, arr[i]);
   }
   return 0;
}
```



```
value of arr[0] is 10
value of arr[1] is 20
value of arr[2] is 30
value of arr[3] is 40
value of arr[4] is 50
```

Storing Values in Arrays

The values can be stored in array using following methods:

- Static initialization
- Initialization of array elements one by one.
- Partial initialization of array
- Array initialization without specifying the size
- Run Time array Initialization

Storing Values in Arrays

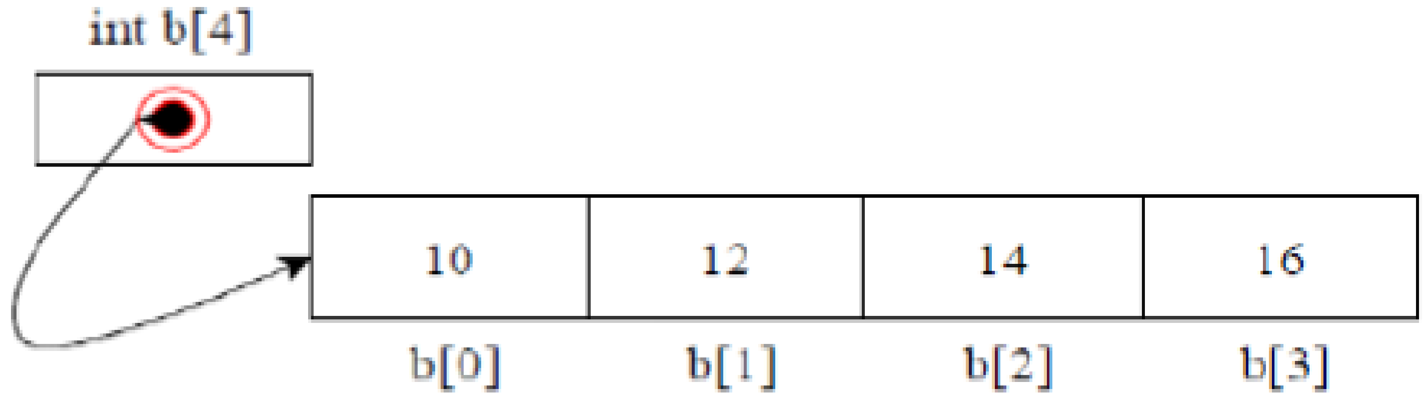
- **Static initialization:-** We can initialize the array in the same way as the ordinary values when they are declared.
- The general syntax of initialization of array is

data_type array_name[array_size]= {List of values};

- Example:
- int b[4]={10,12,14,16};
- Here each value will be stored in respective index values of the array.

Storing Values in Arrays

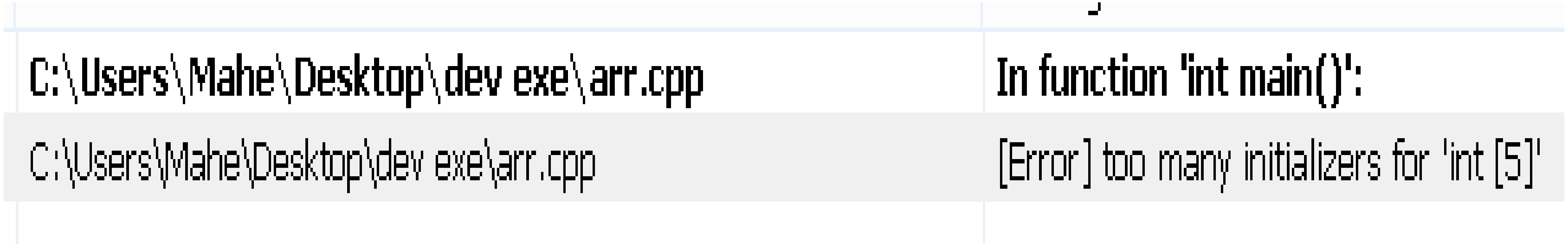
- **Static initialization:-**



- In location b[0] the value 10 is stored and in location b[1] the value 12 is stored and in location b[2] the value 14 is stored, in location b[3] the value 16 is stored.

Storing Values in Arrays

- **Static initialization:-**
- Suppose if we try to insert more values then the size of the array it will give us an error **“Excess elements in array initializer**”
- Example: int b[4]={10,12,14,16,**18**};
- The size of the array b is 4 but we are trying to store 5 values hence we will be getting the error in this case.



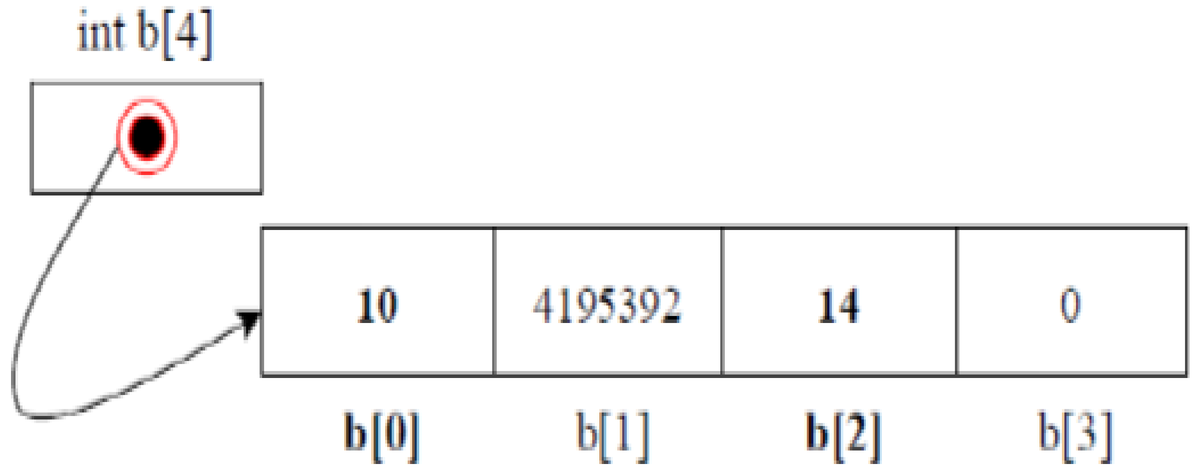
```
C:\Users\Mahe\Desktop\dev exe\arr.cpp    In function 'int main()':
C:\Users\Mahe\Desktop\dev exe\arr.cpp    [Error] too many initializers for 'int [5]'
```

Storing Values in Arrays

2. Initialization of array elements one by one:- Here the user has the liberty to select the locations and store values and the array. This type of initialization is not used much practically.

- Example :int b[4];

```
               b[0]= 10;

                b[2]=14;
```

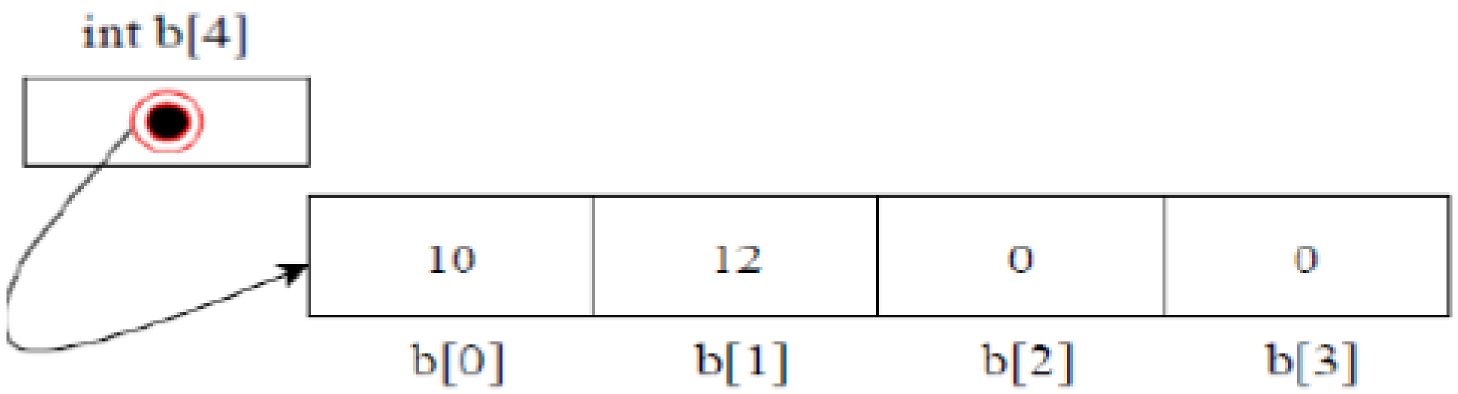


Only the array locations specified by the user will contain the values which the user wants the other locations of array will either be 0 or some garbage value.

Storing Values in Arrays

3. Partial initialization of array :- If the number of values initialized in the array is less than the size of the array then it is called partial initialization. The remaining locations in the array will be initialized to zero or NULL('\0') value automatically.

```
int b[4]={10,12};
```

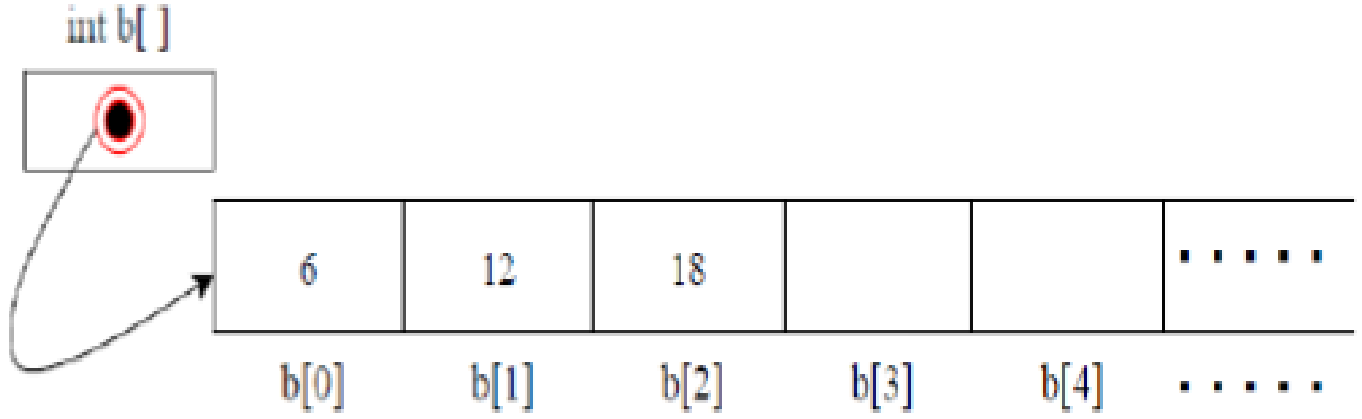


Here the remaining locations in the array will be initialized to zero.

Storing Values in Arrays

4. Array initialization without specifying the size :- Here the size or the array is not specified by the user, the compiler will decide the size based on the number of values declared in the array.

- Example: **int b[ ]={6,12,18};**



- Here the size of the array is specified and the compiler will set the array size as 3 for this example

Storing Values in Arrays

- **Run Time array Initialization:-** If the values are not known by the programmer in advance then the user makes use of run time initialization.
- It helps the programmer to read unknown values from the end users of the program from keyboard by using input function **scanf().**
- Here looping construct is used to read the input values from the keyboard and store them sequentially in the array.

Storing Values in Arrays

- **C program to demonstrate run time initialization**

```
#include<stdio.h>
void main()
{
int b[5],i;
printf(“Enter 5 elements\n”);
for(i=0;i<5;i++)
{
scanf(“%d”,&b[i]);
}
}
```

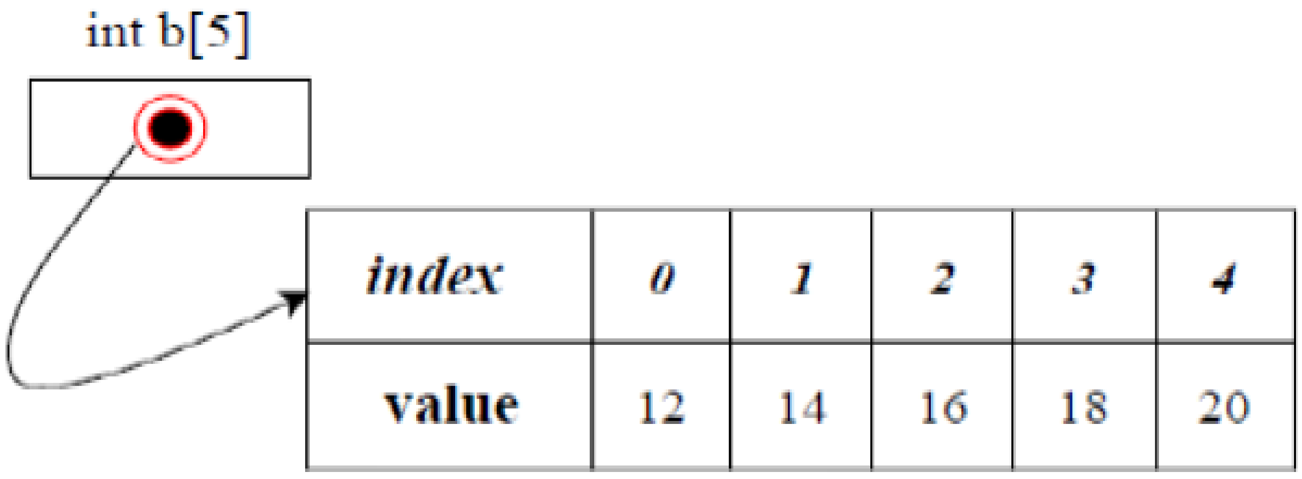
Accessing array elements:

- We can access the elements of array using **index or subscript** of element.

  An index gives the portion of element in the array .

- To access an array element make use of **array_name[index];**
- **Array index starts with 0 and goes till size of array minus 1.**
- To access value 16 we write **b[2]**:
- To print the value 18 :

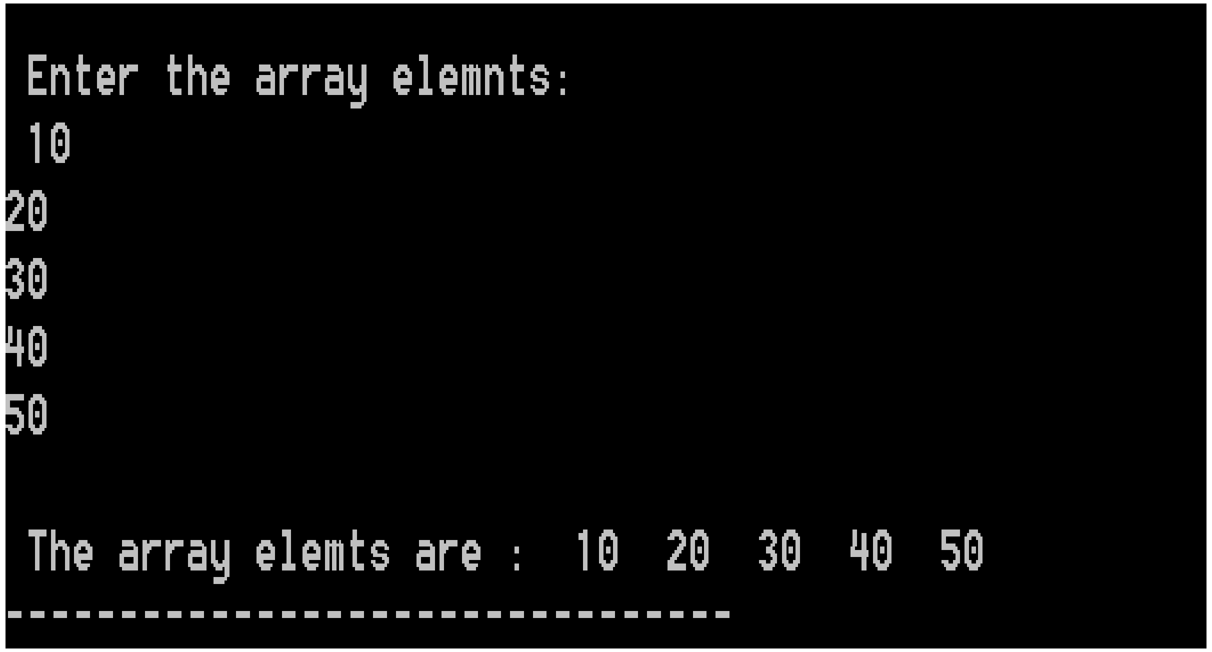
printf(“%d”,b[3]);



index	*0*	*1*	*2*	*3*	*4*
value	12	14	16	18	20

Write a C program to read and print n integer elements in an array

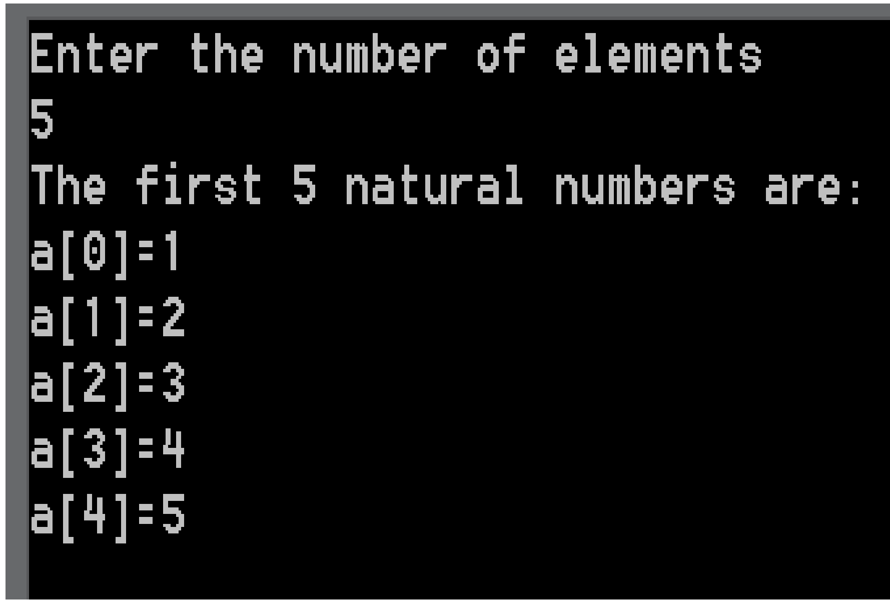
```
#include<stdio.h>
int main()
{
    int arr[5];
    int i;
    printf("\n Enter the array elemnts:\n ");
    for(i = 0; i<5; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("\n The array elements are : ");
    for(i = 0; i<5; i++)
    {
        printf(" %d ", arr[i]);
    }
    return 0;
}
```



```
 Enter the array elemnts:
 10
20
30
40
50

 The array elemts are :  10  20  30  40  50
--------------------------------
```

2. Write a c program to display first n natural numbers in an array.

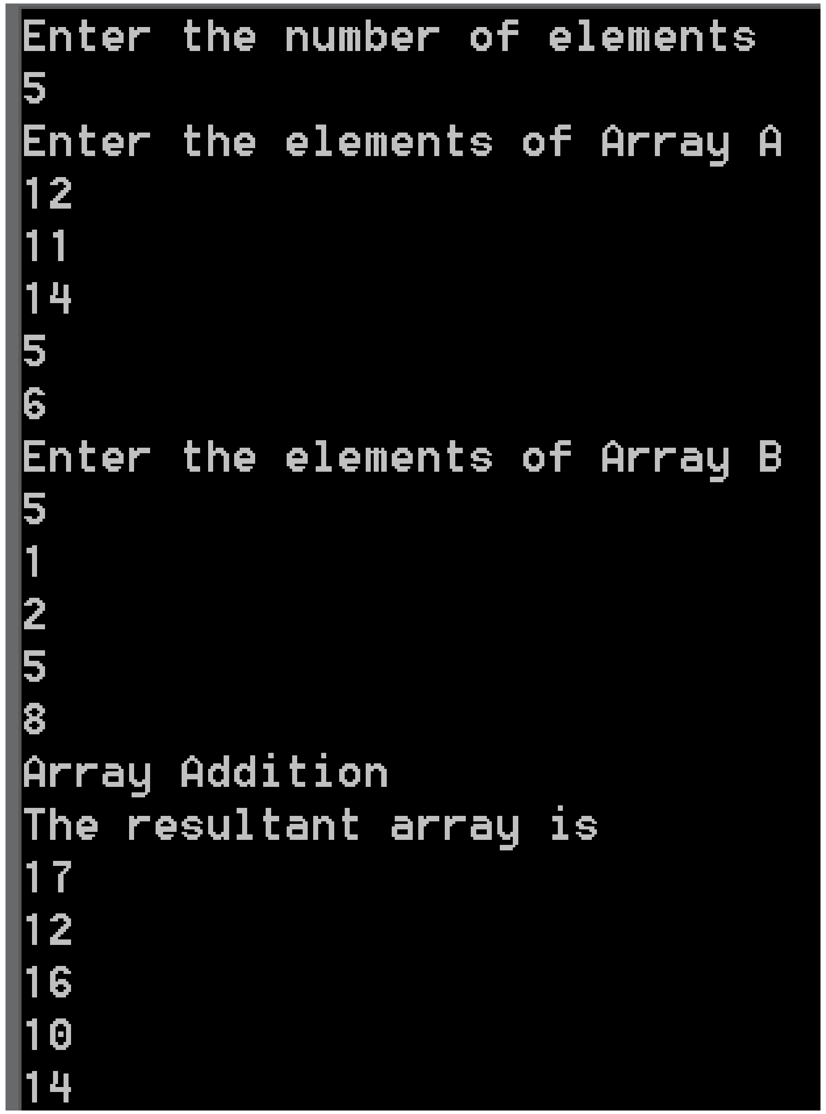
```
#include<stdio.h>
void main()
{
        int a[20 ],n,i;
        printf("Enter the number of elements\n");
        scanf("%d",&n);
        printf("The first %d natural numbers are:\n",n);
        for(i=0;i<n;i++)
        {
                a[i]=i+1;
                printf("a[%d]=%d\n",i,a[i]);
        }
}
```



```
Enter the number of elements
5
The first 5 natural numbers are:
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5
```

3. Write a c program to add two one dimensional array

```
include<stdio.h>
void main()
{
          int a[20 ],b[20],c[20],n, i;
          printf("Enter the number of elements\n");
          scanf("%d",&n);
          printf("Enter the elements of Array A\n");
          for(i=0;i<n;i++)
                    scanf("%d",&a[i]);
          printf("Enter the elements of Array B\n");
                    for(i=0;i<n;i++)
          scanf("%d",&b[i]);
          printf("Array Addition\n");
          for(i=0;i<n;i++)
                    c[i]=a[i]+b[i];
          printf("The resultant array is \n");
          for(i=0;i<n;i++)
                    printf("%d\n",c[i]);
}
```

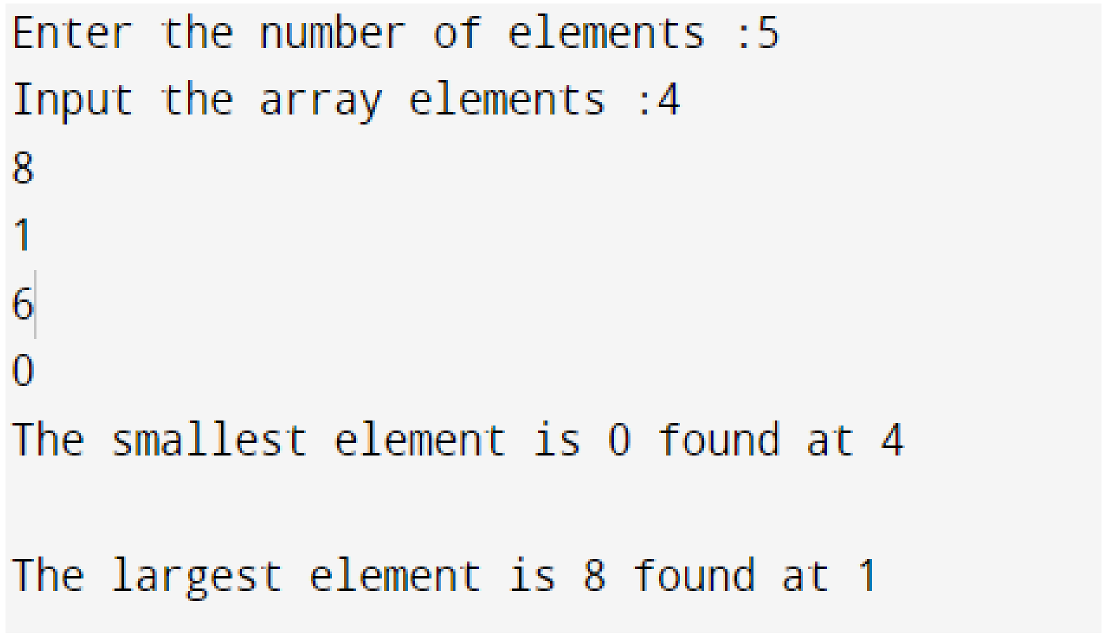


```
Enter the number of elements
5
Enter the elements of Array A
12
11
14
5
6
Enter the elements of Array B
5
1
2
5
8
Array Addition
The resultant array is
17
12
16
10
14
```

4. Write C program to find largest and smallest number in an array of n elements alon with its position.

```
#include<stdio.h>
int main()
{
int a[50],i,n,large,small,large_index,small_index;
printf("\nEnter the number of elements :");
scanf("%d",&n);
printf("\nInput the array elements :" );
for(i=0;i<n;i++)
scanf("%d",&a[i]);
large=small=a[0];
for(i=0;i<n;i++)
{
if(a[i]>large){
large=a[i];
large_index=i;
}
if(a[i]<small){
small=a[i];
small_index=i;
}
}
printf("\nThe smallest element is %d found at %d\n",small, small_index);
printf("\nThe largest element is %d found at %d\n",large, large_index);
}
```

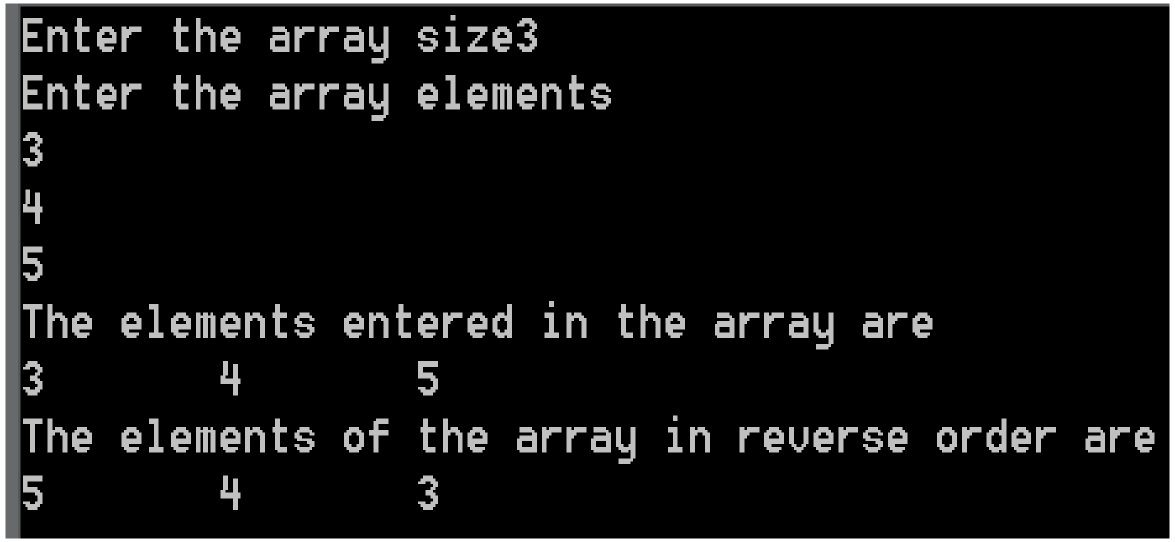
Output



```
Enter the number of elements :5
Input the array elements :4
8
1
6
0
The smallest element is 0 found at 4

The largest element is 8 found at 1
```

5. Write C program to read n integer elements in an array and print the same in reverse order

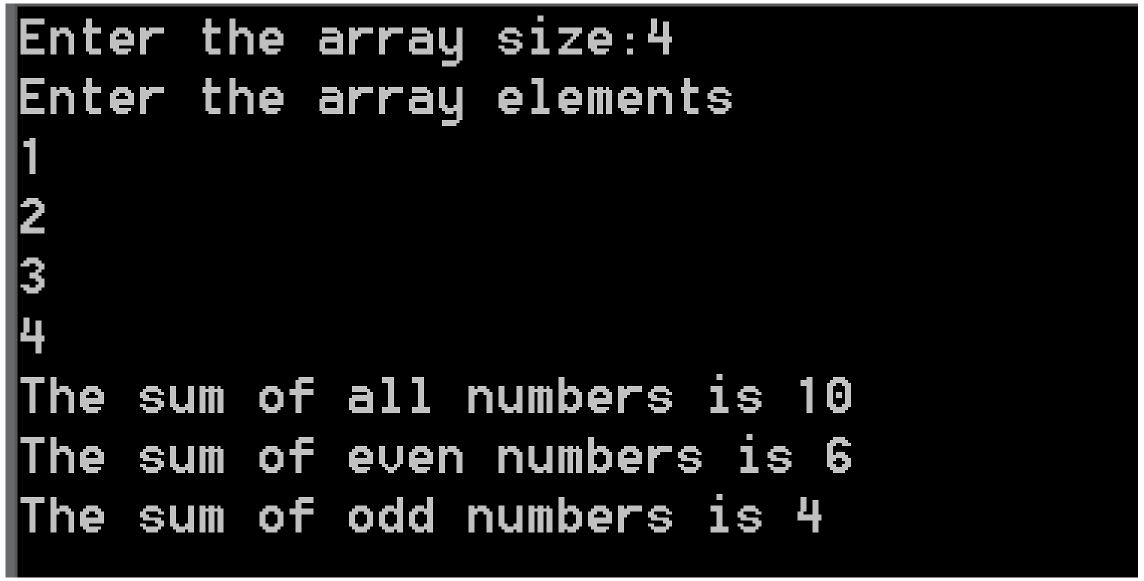
```
#include<stdio.h>
void main()
{
        int a[20 ],n,i;
        printf("Enter the array size");
        scanf("%d",&n);
        Printf("Enter the array elements\n");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        printf("The elements entered in the array are\n");
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        printf("The elements of the array in reverse order are\n");
        for(i=n-1;i>=0;i--)
                printf("%d\t",a[i]);
}
```



```
Enter the array size3
Enter the array elements
3
4
5
The elements entered in the array are
3       4       5
The elements of the array in reverse order are
5       4       3
```

6. Write a C Program to find the sum of odd, even, all of n numbers using arrays .

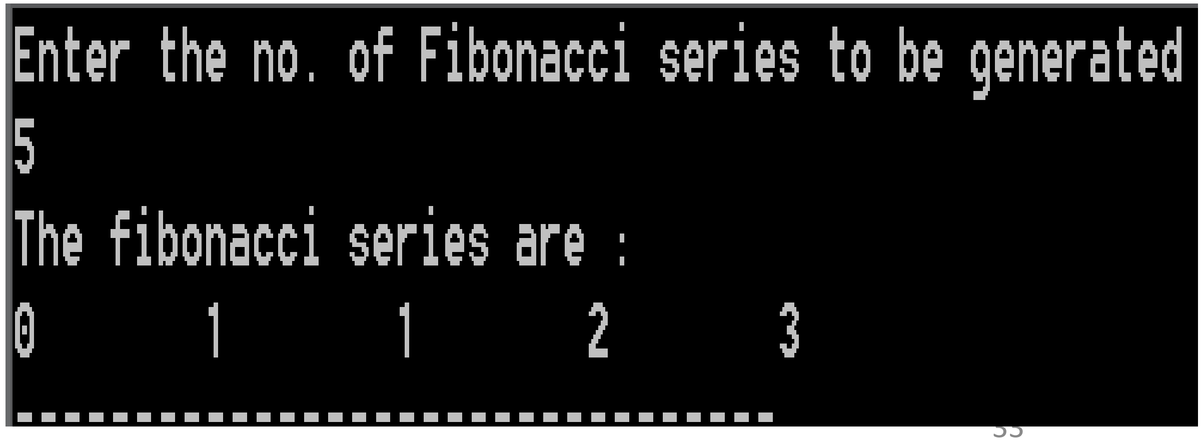
```
#include<stdio.h>
int main()
{
int a[20 ],sum=0,esum=0,osum=0,n,i;
printf("Enter the array size:");
scanf("%d",&n);
printf("Enter the array elements\n");
 for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
sum=sum+a[i]; if((a[i]%2)==0)
esum=esum+a[i];
else
osum=osum+a[i];
}
printf("The sum of all numbers is %d\n",sum);
printf("The sum of even numbers is %d\n",esum);
printf("The sum of odd numbers is %d\n",osum);
return 0;
}
```



```
Enter the array size:4
Enter the array elements
1
2
3
4
The sum of all numbers is 10
The sum of even numbers is 6
The sum of odd numbers is 4
```

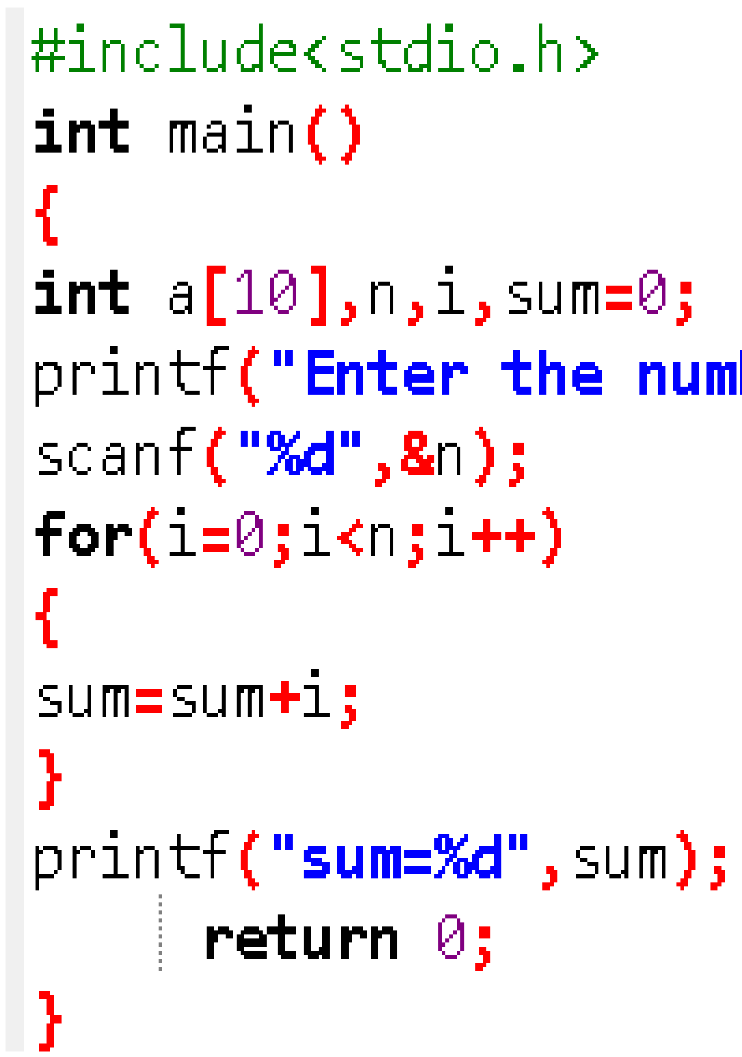
7. Write C program to generate Fibonacci series using arrays

```c
#include<stdio.h>
int main()
{
int fib[20],n,i;
printf("Enter the no. of Fibonacci series to be generated\n");
 scanf("%d",&n);
fib[0]=0;
fib[1]=1;
 if(n==1)
printf("fibonacci series is %d",fib[0]);
else if(n==2)
printf("fibonacci series is %d \t %d",fib[0],fib[1]);
else
for(i=2;i<n;i++)
{
fib[i]=fib[i-1]+fib[i-2];
}
printf("The fibonacci series are :\n");
 for(i=0;i<n;i++)
{
printf("%d\t",fib[i]);
}
return 0;
}
```

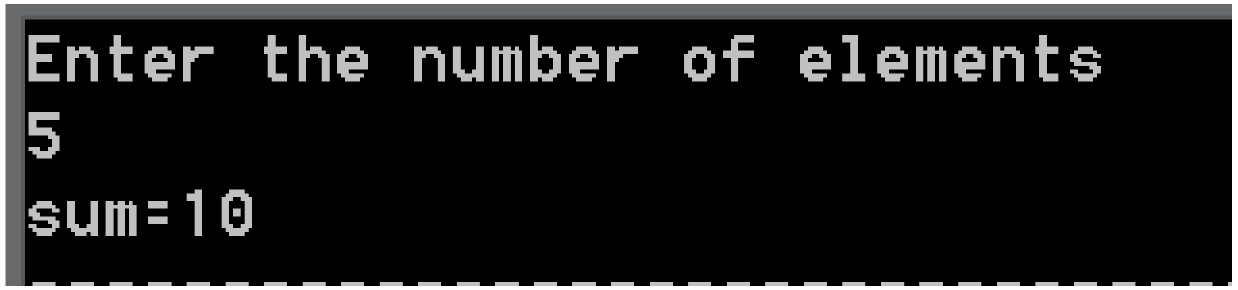


```
Enter the no. of Fibonacci series to be generated
5
The fibonacci series are :
0       1       1       2       3
```

8. Write C program to find sum of n Natural numbers



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



```
Enter the number of elements
5
sum=10
```

Sorting Techniques:-

- The Process of arranging the elements in ascending or descending order is called sorting

1) Bubble sort:

The sorting algorithm is a comparison based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

This algorithm works by repeatedly swapping the adjacent elements if they are in wrong order.

Bubble sort with **n element** required **n - 1** passes(uses outer for loop)

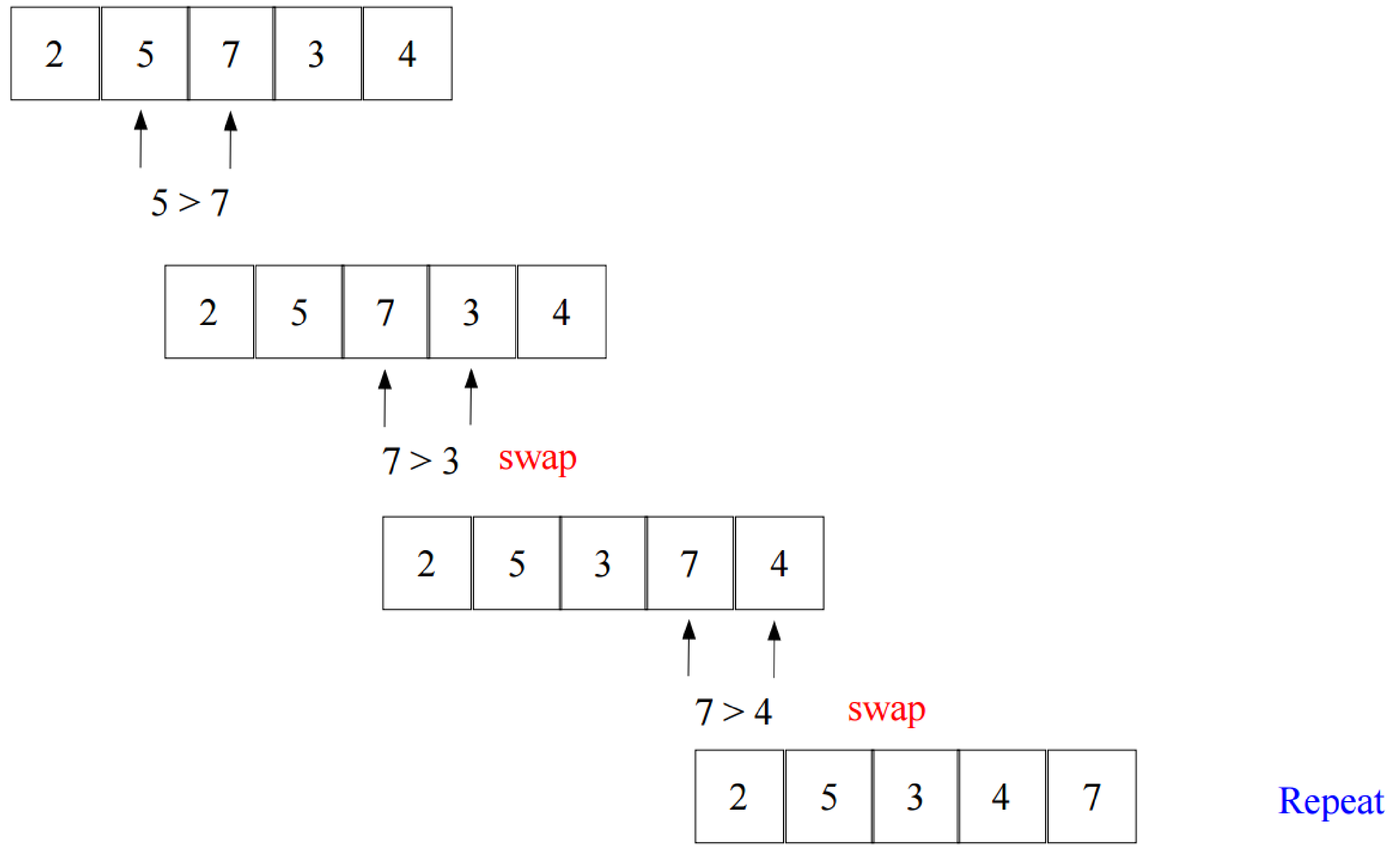
Sorting Techniques:-

1) **Bubble sort:**

```
#include <stdio.h>
void bubble_sort(int a[], int n) {
   int i , j ,temp;
   for (i = 0; i < n-1; i++) {   // loop n times - 1 per element
       for (j = 0; j < n - i - 1; j++) {    // last i elements are sorted already
           if (a[j] > a[j + 1]) {  // swop if order is broken
               temp = a[j];
               a[j] = a[j + 1];
               a[j + 1] = temp;
           }
       }
   }
}
```

Sorting Techniques:-

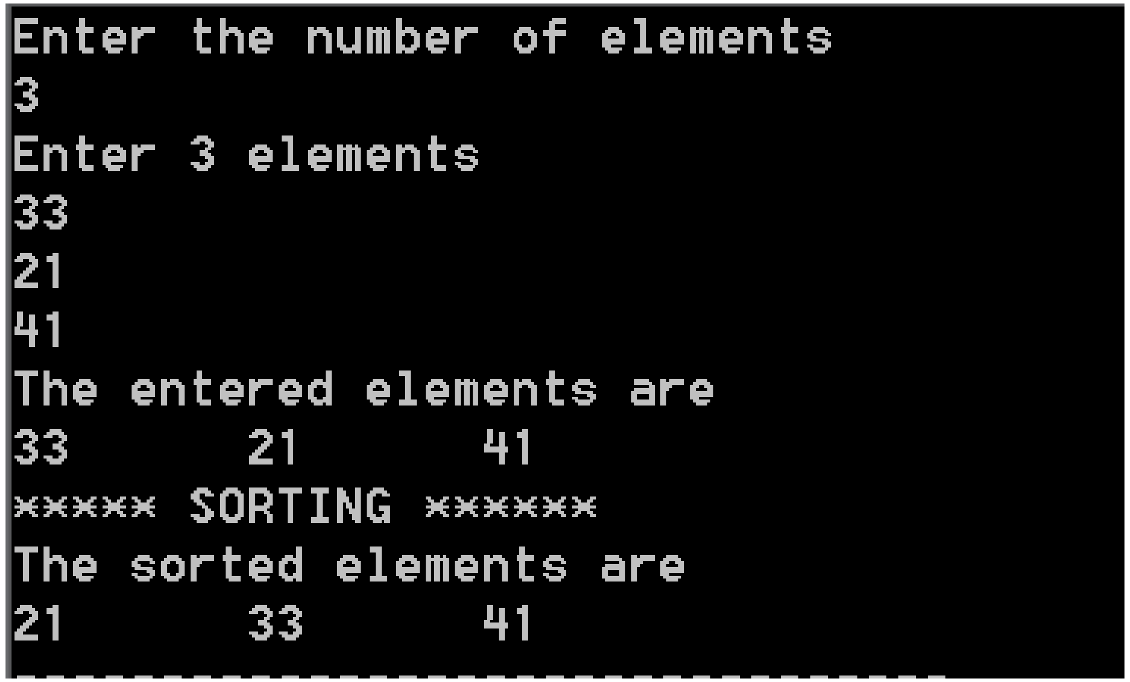
1.Bubble sort



Bubble Sort

```
#include<stdio.h>
void  main ()
{
        int a[50],n,i,j,temp;
        printf ("Enter the number of
        elements\n");scanf("%d",&n);
        printf ("Enter %delements\n",n);
        for(i=0;i<n;i++)
        {
                scanf("%d", &a[i]);
        }
        printf ("The entered elements
        are\n");for(i=0;i<n;i++)
        {
                printf("%d\t", a[i]);
        }
        Printf ("\n***** SORTING******\n");
        for(i=1; i<n-1;i++)
        {
                for(j=0;j<n-i-1;j++)
                {
                        if(a[j]>a[j+1])
                        {
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
        printf("The sorted elements are\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
}
```

Bubble Sort



```
Enter the number of elements
3
Enter 3 elements
33
21
41
The entered elements are
33      21      41
***** SORTING ******
The sorted elements are
21      33      41
```

Sorting Techniques:-

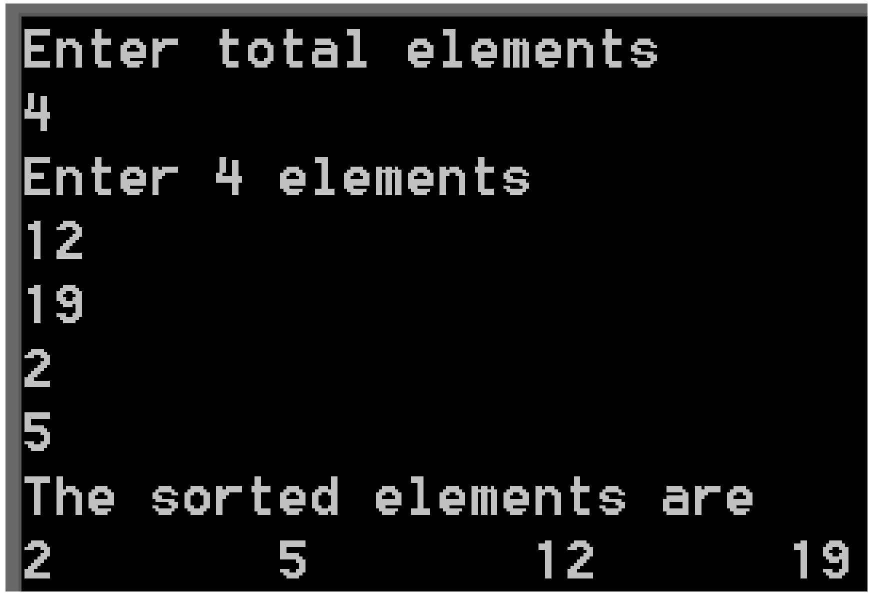
2) Selection sort:

This is an in-place comparison based algorithm It is comparison based algorithm in which list is divided into 2 parts.

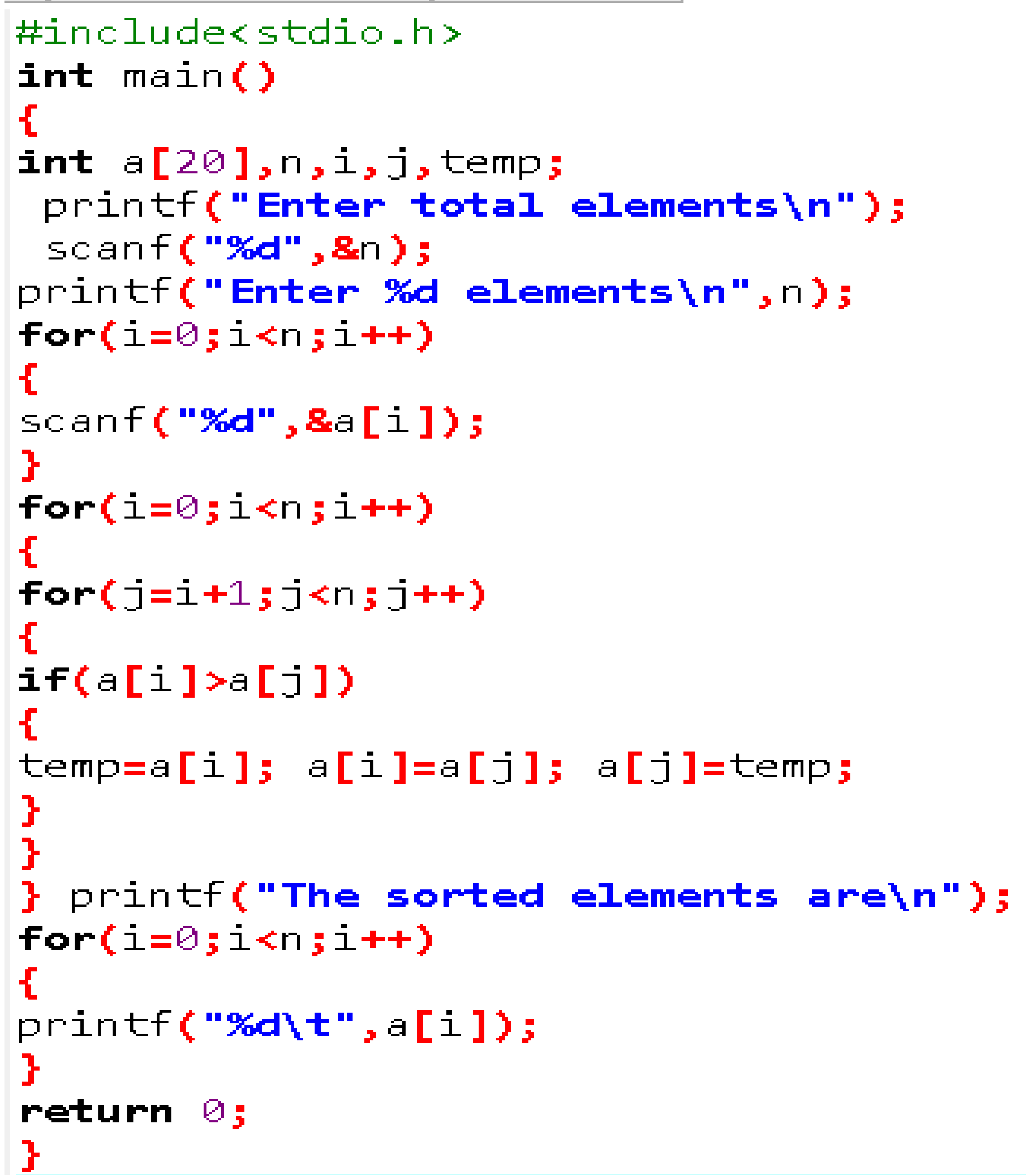
The sorted part at left and unsorted part at right end. Initially sorted part is empty and unsorted part is entire list.

The smallest element is taken from the unsorted array and swapped with the leftmost element and the element becomes the part of sorted array.

Selection Sort



```
Enter total elements
4
Enter 4 elements
12
19
2
5
The sorted elements are
2       5       12      19
```



```c
#include<stdio.h>
int main()
{
int a[20],n,i,j,temp;
 printf("Enter total elements\n");
 scanf("%d",&n);
printf("Enter %d elements\n",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 sorted elements are\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
return 0;
}
```

Searching Techniques:

The process of finding a particular element in the large amount of data is called searching.

Linear search:- A Linear search is also called as sequential Search. In this technique we search for a given specific element called as key element in the large list of data in sequential order. If the key element is present in the list of data then the search is successful otherwise search is unsuccessful.

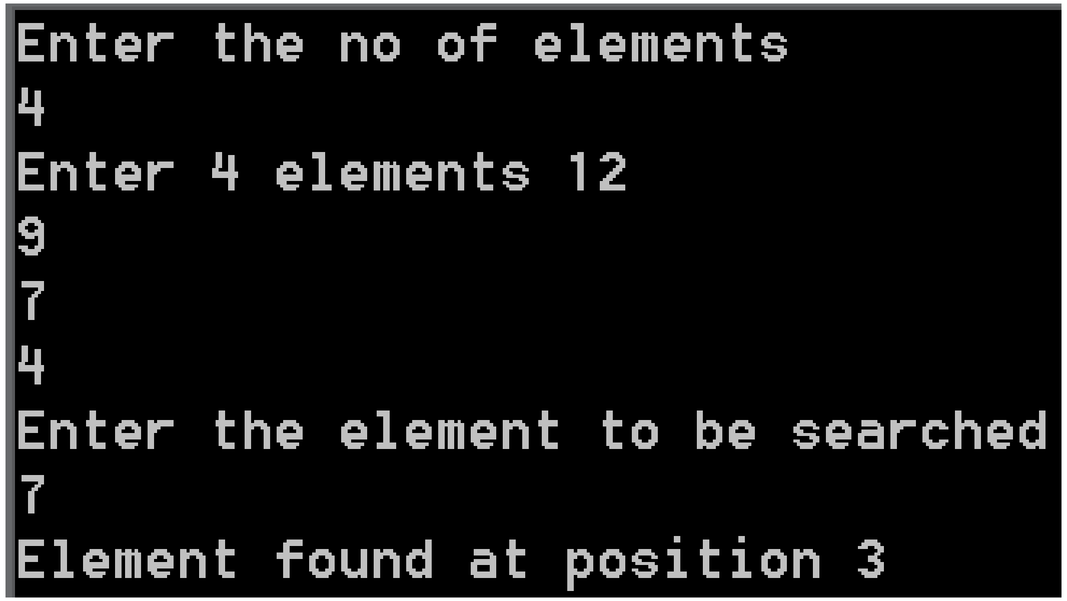
Benefits:

- Simple approach
- Works well for small arrays
- Used to search when the elements are not sorted

Disadvantages:

- Less efficient if the array is large
- If the elements are already sorted, linear search is not efficient.

Linear Search



```
Enter the no of elements
4
Enter 4 elements 12
9
7
4
Enter the element to be searched
7
Element found at position 3
```

```
#include<stdio.h>
int main()
{
int a[100],n,i,key,flag=0;
printf("Enter the no of elements\n");
 scanf("%d",&n);
printf("Enter %d elements ",n);
 for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the element to be searched\n");
scanf("%d",&key);
for(i=0;i<n;i++)
{

if(key==a[i])
{
flag=1; break;
}
}
if(flag==1)
printf("Element found at position %d\n",i+1);
else
printf("Element not found\n");
return 0;
}
```

Searching Techniques:

Binary Search: It is fast search algorithm which works on the principle of divide and conquer. for this algorithm to work properly the data collection should be in the sorted form

1. Divides the array into three sections:

– middle element

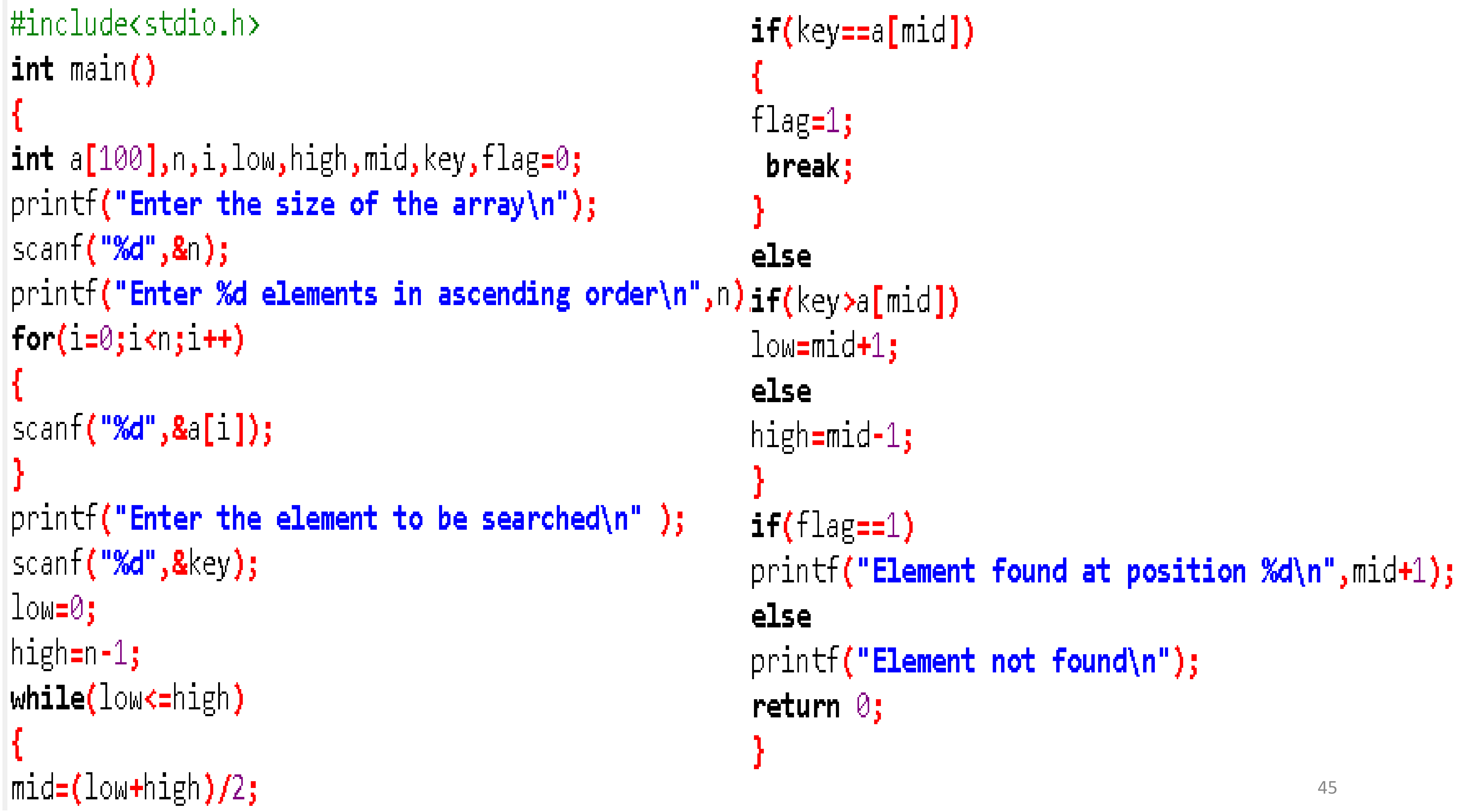
– elements on one side of the middle element

– elements on the other side of the middle element

2. If the middle element is the correct value, done. Otherwise, go to step 1. using only the half of the array that may contain the correct value.

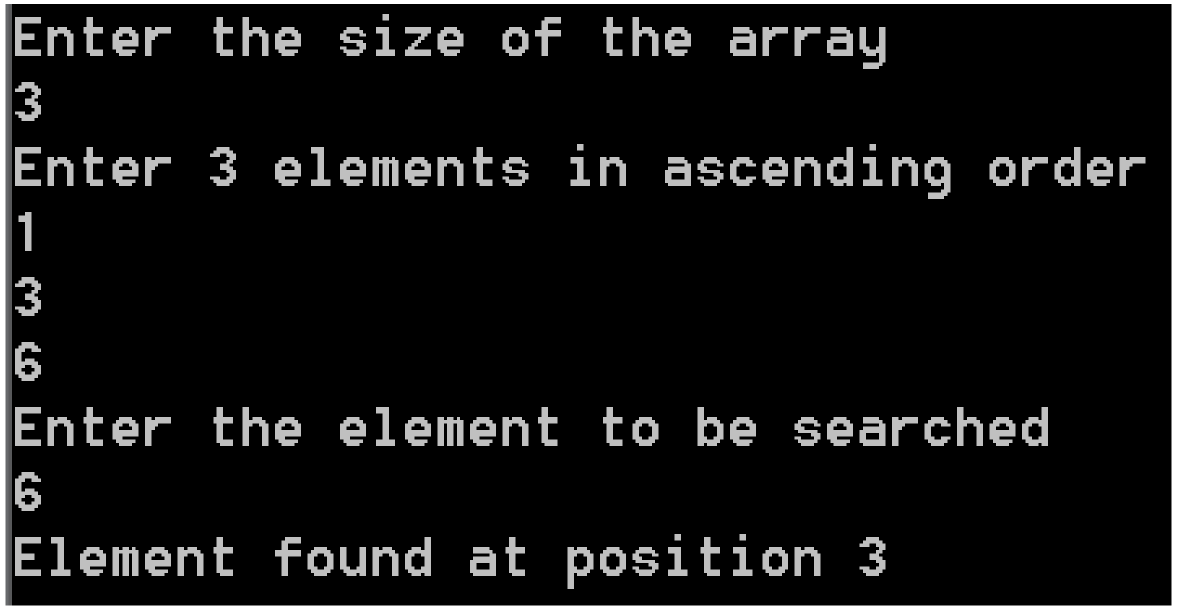
3. Continue steps 1. and 2. until either the value is found or there are no more elements to examine

Binary Searching



```
#include<stdio.h>
int main()
{
int a[100],n,i,low,high,mid,key,flag=0;
printf("Enter the size of the array\n");
scanf("%d",&n);
printf("Enter %d elements in ascending order\n",n)
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Enter the element to be searched\n" );
scanf("%d",&key);
low=0;
high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(key==a[mid])
{
flag=1;
 break;
}
else
if(key>a[mid])
low=mid+1;
else
high=mid-1;
}
if(flag==1)
printf("Element found at position %d\n",mid+1);
else
printf("Element not found\n");
return 0;
}
```

Binary Searching



```
Enter the size of the array
3
Enter 3 elements in ascending order
1
3
6
Enter the element to be searched
6
Element found at position 3
```

Two Dimensional Array

- The simplest form of multidimensional array is two dimensional array. Arrays with two or more dimensions are called multi-dimensional arrays. (in terms of rows and columns) of same data type or An array which has two subscripts are known as two dimensional arrays.
- The first subscript represents rows and the second subscript represent column.

  Syntax:

  data_type array_name[size1][size2];

Two Dimensional Array

where,

data_type: is the type of data to be stored and processed in the computer's

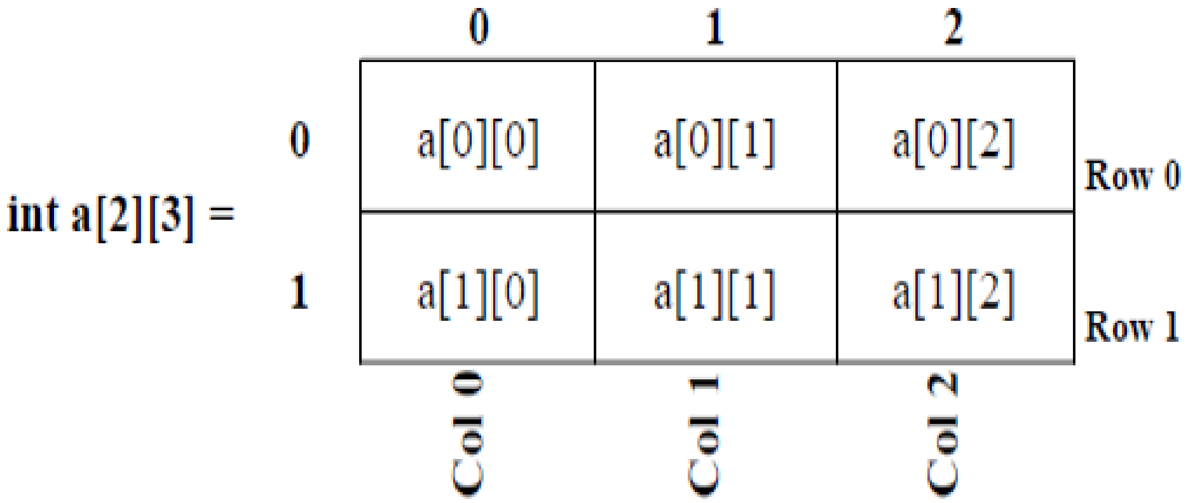
memory array_name: is a valid identifier representing name of the array

[size1]: indicates number of rows in the array

[size2]: indicates the number of columns in the array.

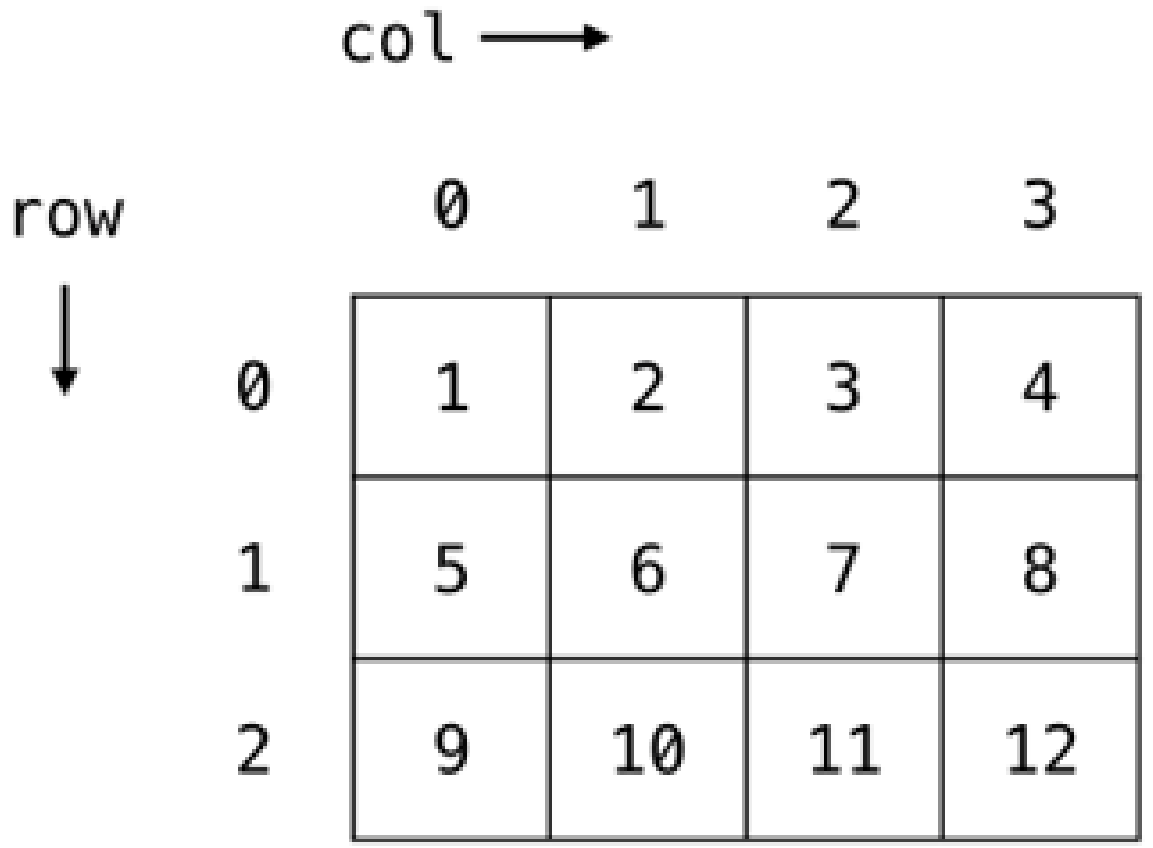
Example:

int a[2][3];



Two Dimensional Array

```
int num[3][4] = {
  {1, 2,  3,  4},
  {5, 6,  7,  8},
  {9, 10, 11, 12}
};
```



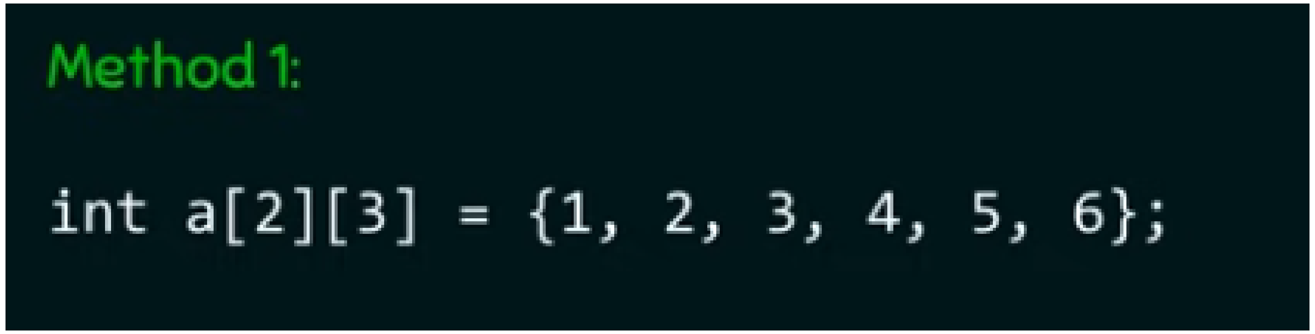
	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Two Dimensional Array

Initialization of two dimensional array :

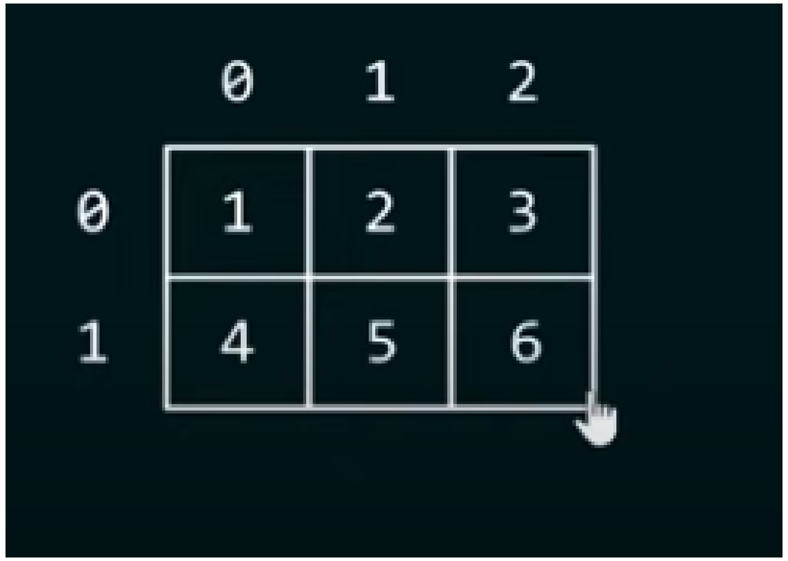
1)Initializing all elements row wise:- A multidimensional array can be initialized by specifying bracketed values for each row.

Example:



Method 1:

```
int a[2][3] = {1, 2, 3, 4, 5, 6};
```

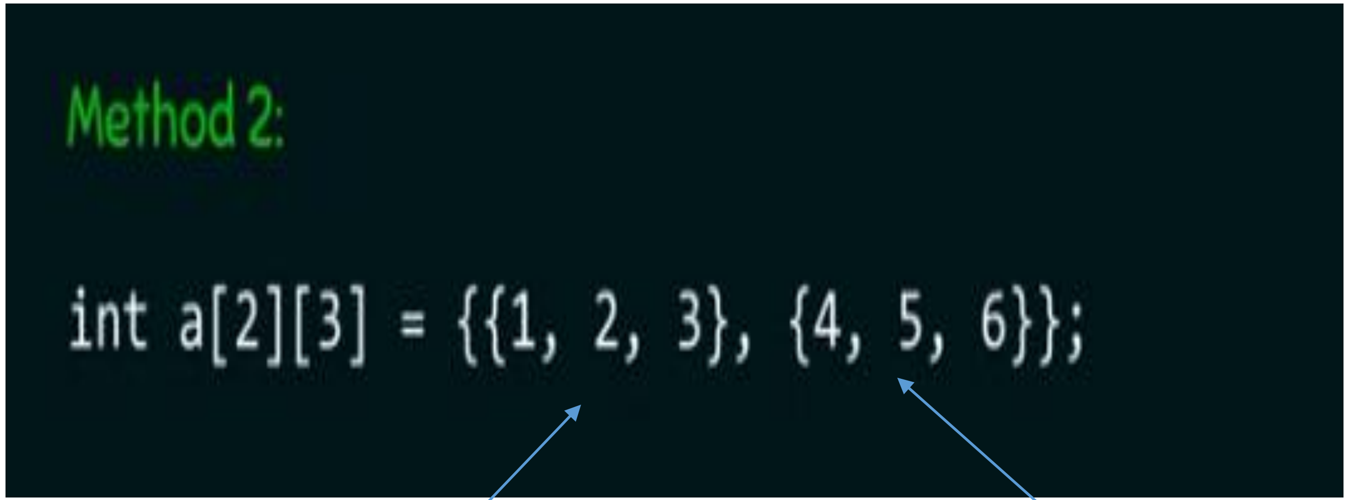


	0	1	2
0	1	2	3
1	4	5	6

Two Dimensional Array

Initialization of two dimensional array :

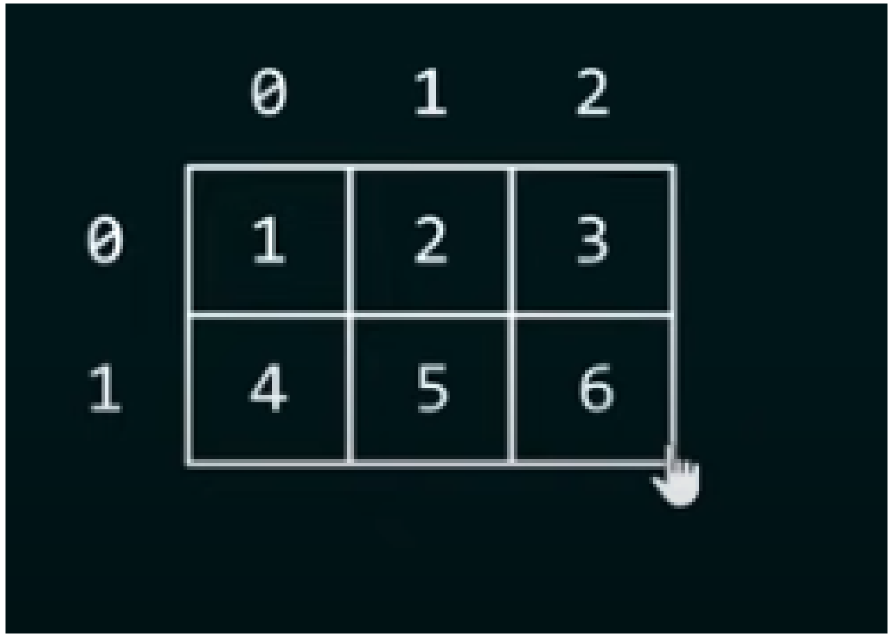
Example:



Method 2:

```
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

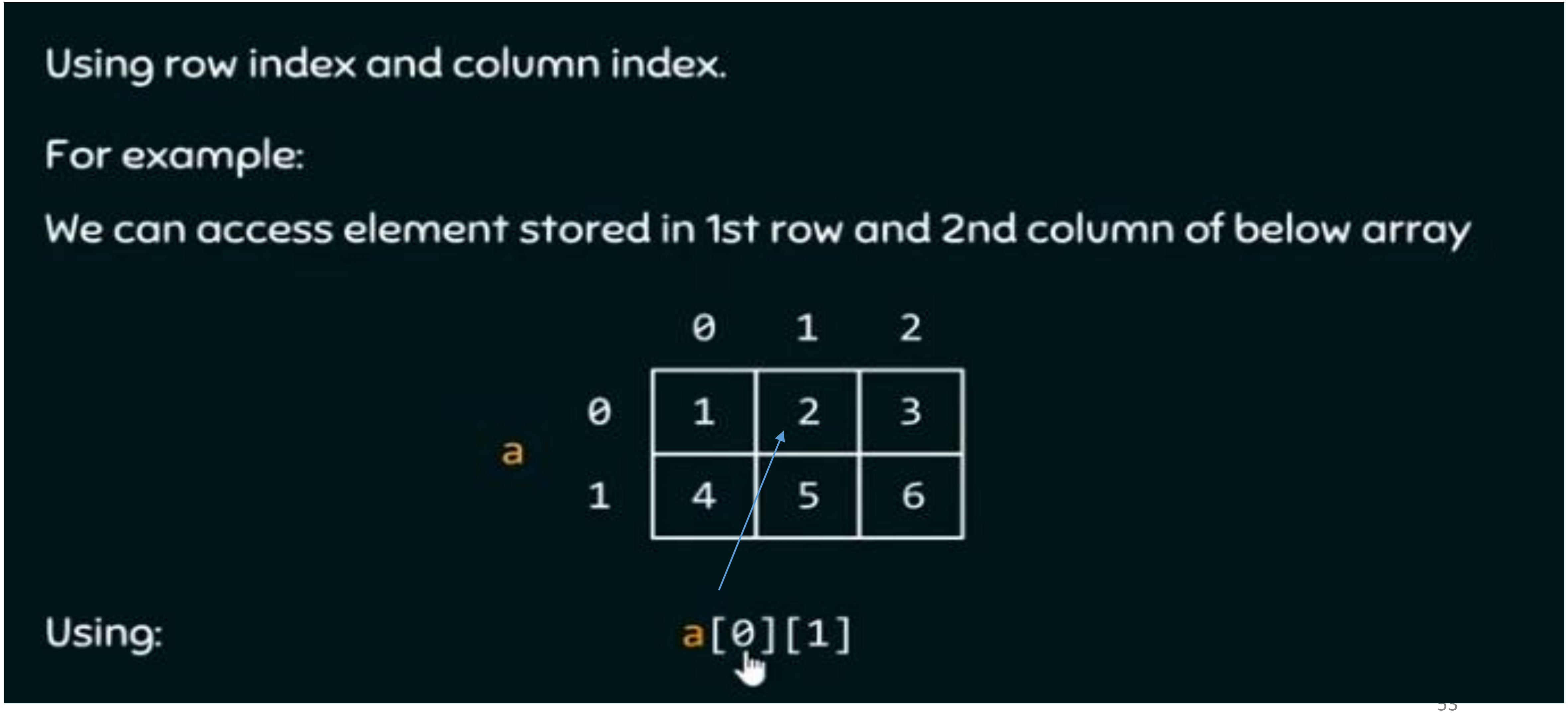
Row 1 **Row 2**



$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \end{array}$$

Two Dimensional Array

Accessing the 2 dimensional array:-



Using row index and column index.

For example:

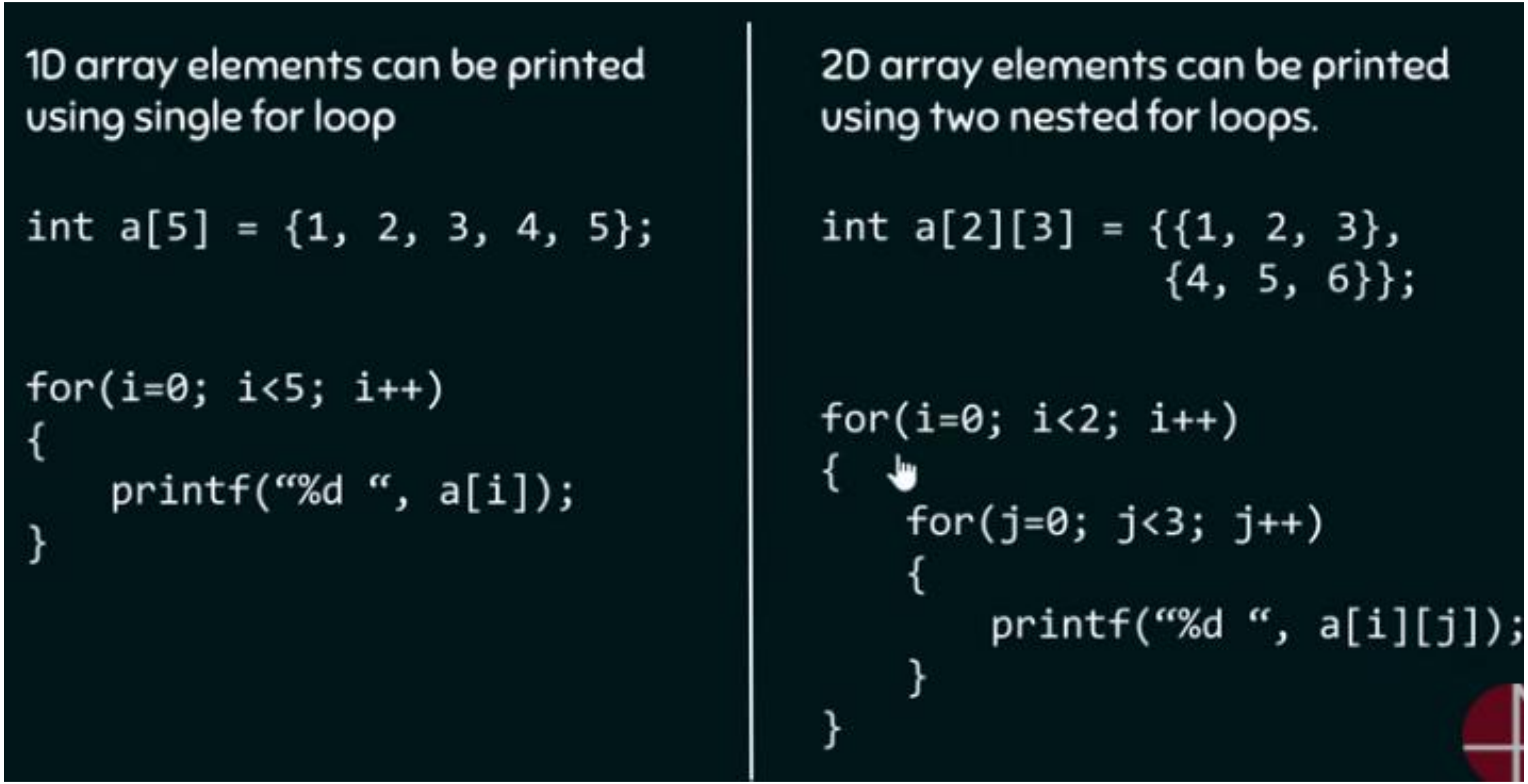
We can access element stored in 1st row and 2nd column of below array

$$a \quad \begin{array}{c|c|c|c|} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ \hline 1 & 4 & 5 & 6 \\ \hline \end{array}$$

Using: `a[0][1]`

Two Dimensional Array

Reading and printing 2 dimensional array:-



1D array elements can be printed using single for loop

```
int a[5] = {1, 2, 3, 4, 5};


for(i=0; i<5; i++)
{
    printf(“%d “, a[i]);
}
```

2D array elements can be printed using two nested for loops.

```
int a[2][3] = {{1, 2, 3},
               {4, 5, 6}};


for(i=0; i<2; i++)
{
    for(j=0; j<3; j++)
    {
        printf(“%d “, a[i][j]);
    }
}
```

Two Dimensional Array

Reading and printing 2 dimensional array:-

where **m-rowsize, n-columnsize, i-row index and j-column index**

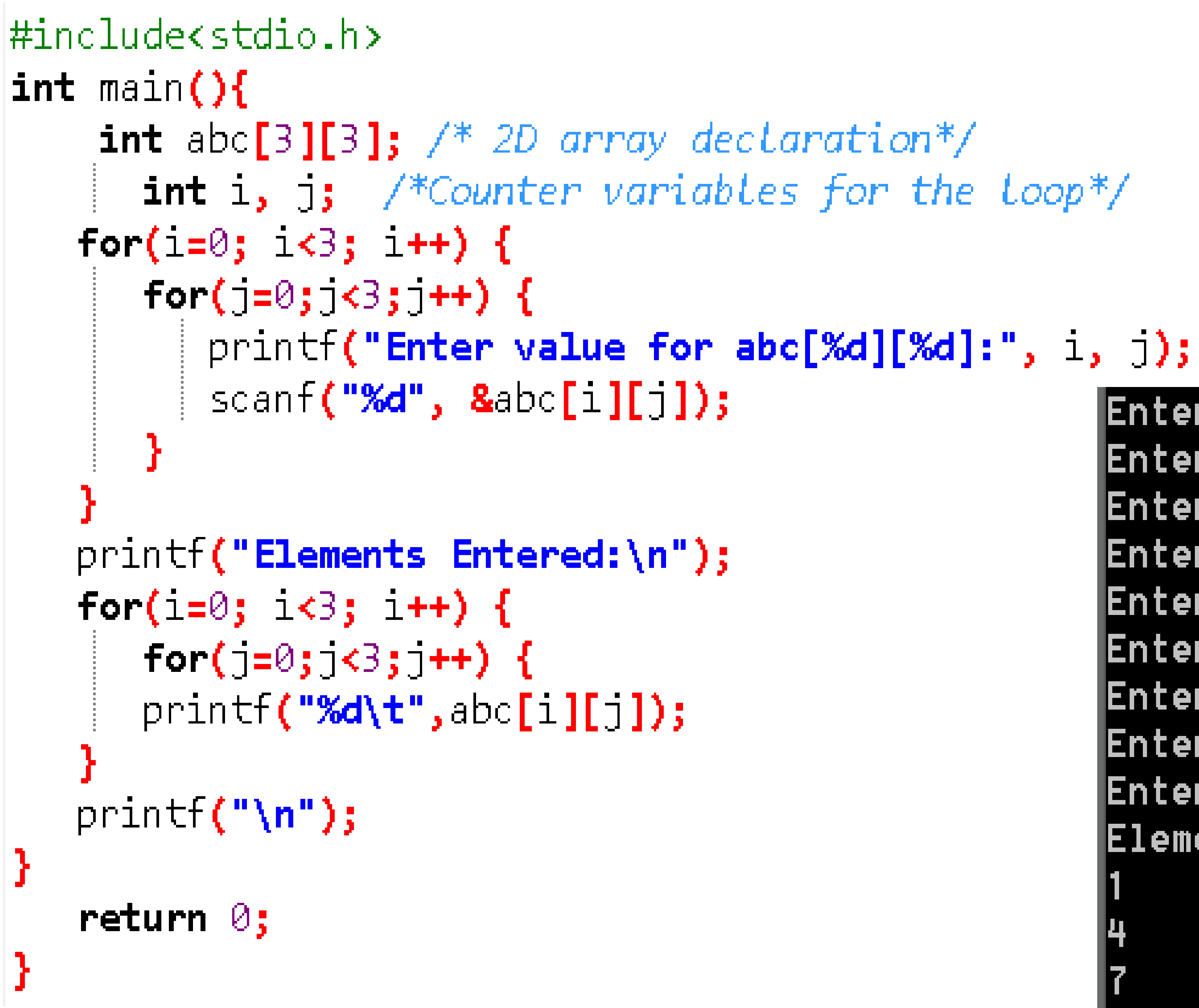
Reading

```
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d", &a[i][j]);
}
```

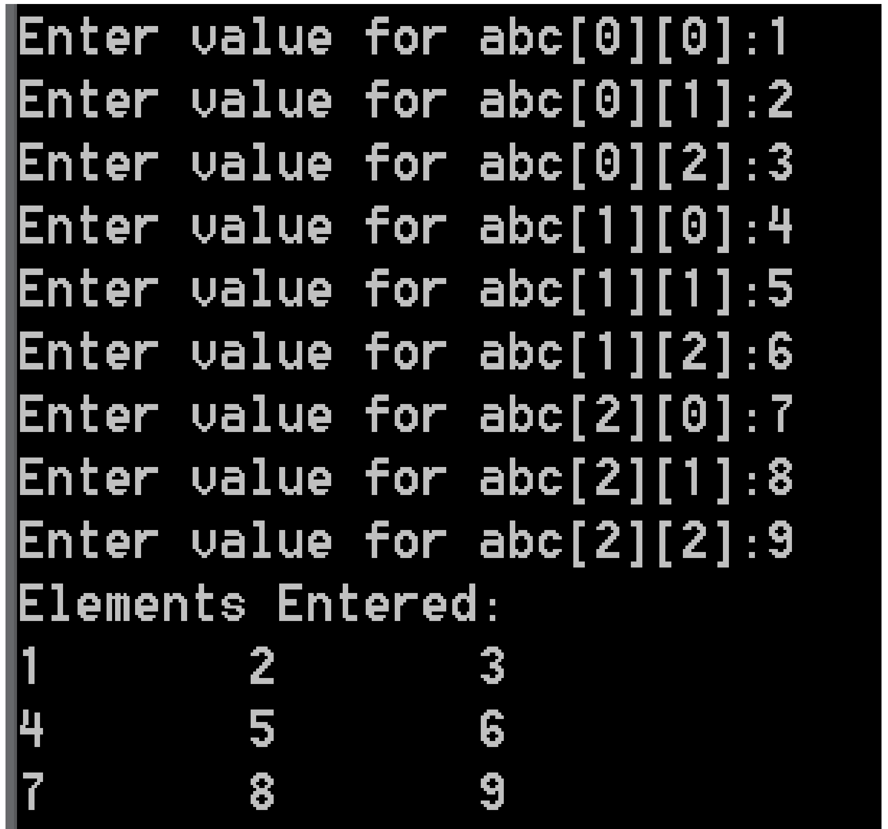
Printing

```
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d", a[i][j]);
}
```

Write a c program to read and print the matrix of m rows and n columns



```c
#include<stdio.h>
int main(){
     int abc[3][3]; /* 2D array declaration*/
       int i, j;  /*Counter variables for the loop*/
    for(i=0; i<3; i++) {
       for(j=0;j<3;j++) {
          printf("Enter value for abc[%d][%d]:", i, j);
          scanf("%d", &abc[i][j]);
       }
    }
    printf("Elements Entered:\n");
    for(i=0; i<3; i++) {
       for(j=0;j<3;j++) {
       printf("%d\t",abc[i][j]);
    }
    printf("\n");
}
    return 0;
}
```

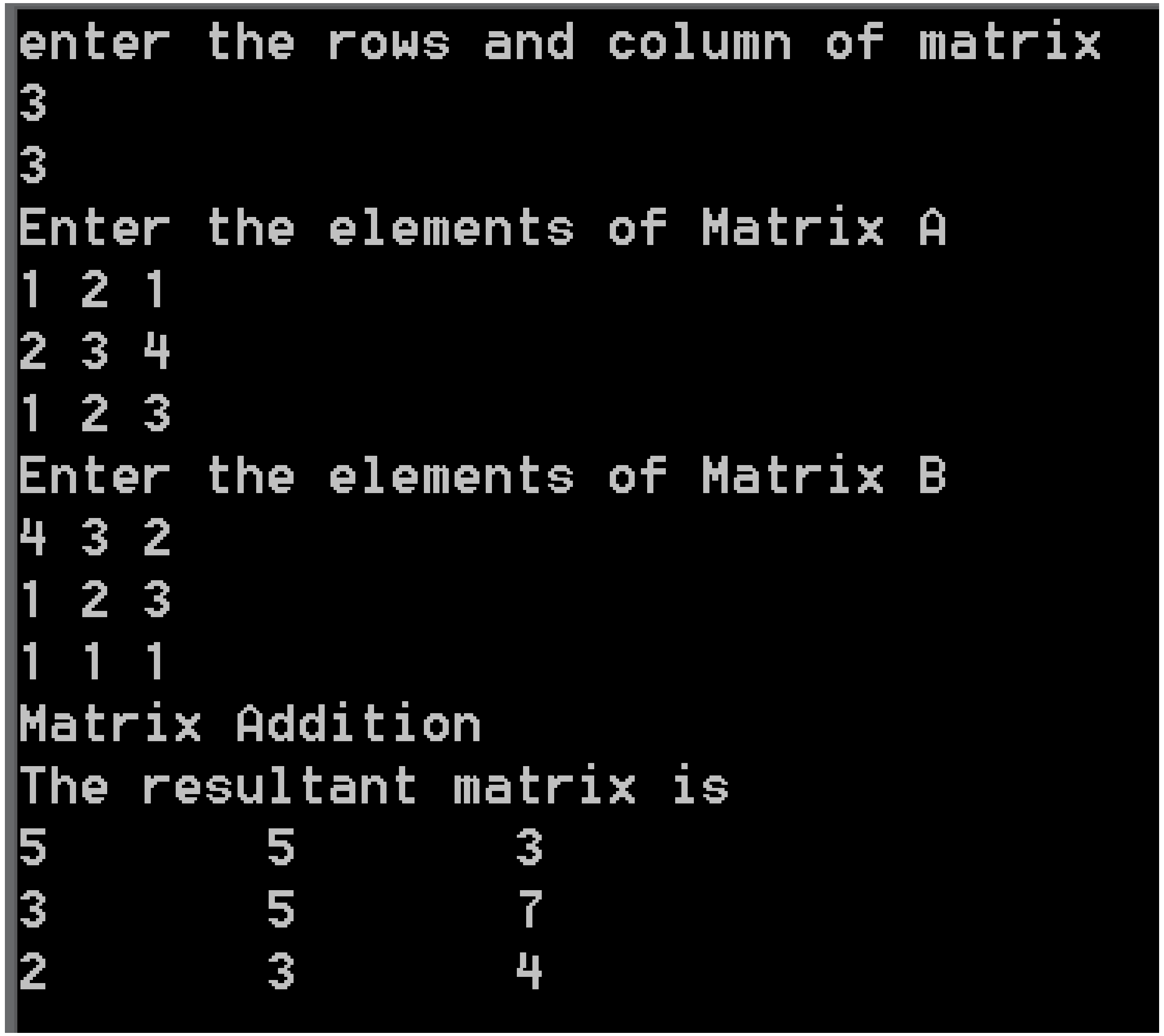


```
Enter value for abc[0][0]:1
Enter value for abc[0][1]:2
Enter value for abc[0][2]:3
Enter value for abc[1][0]:4
Enter value for abc[1][1]:5
Enter value for abc[1][2]:6
Enter value for abc[2][0]:7
Enter value for abc[2][1]:8
Enter value for abc[2][2]:9
Elements Entered:
1       2       3
4       5       6
7       8       9
```

Write a c program to add two matrices

```
#include<stdio.h>
int main()
{
int a[20 ][20],b[20][20],c[20][20],m,n, i,j;
printf("enter the rows and column of matrix\n");
scanf("%d%d",&m,&n);
printf("Enter the elements of Matrix A\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("Enter the elements of Matrix B\n");
 for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("Matrix Addition\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}
printf("The resultant matrix is\n");
 for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
return 0;
}
```

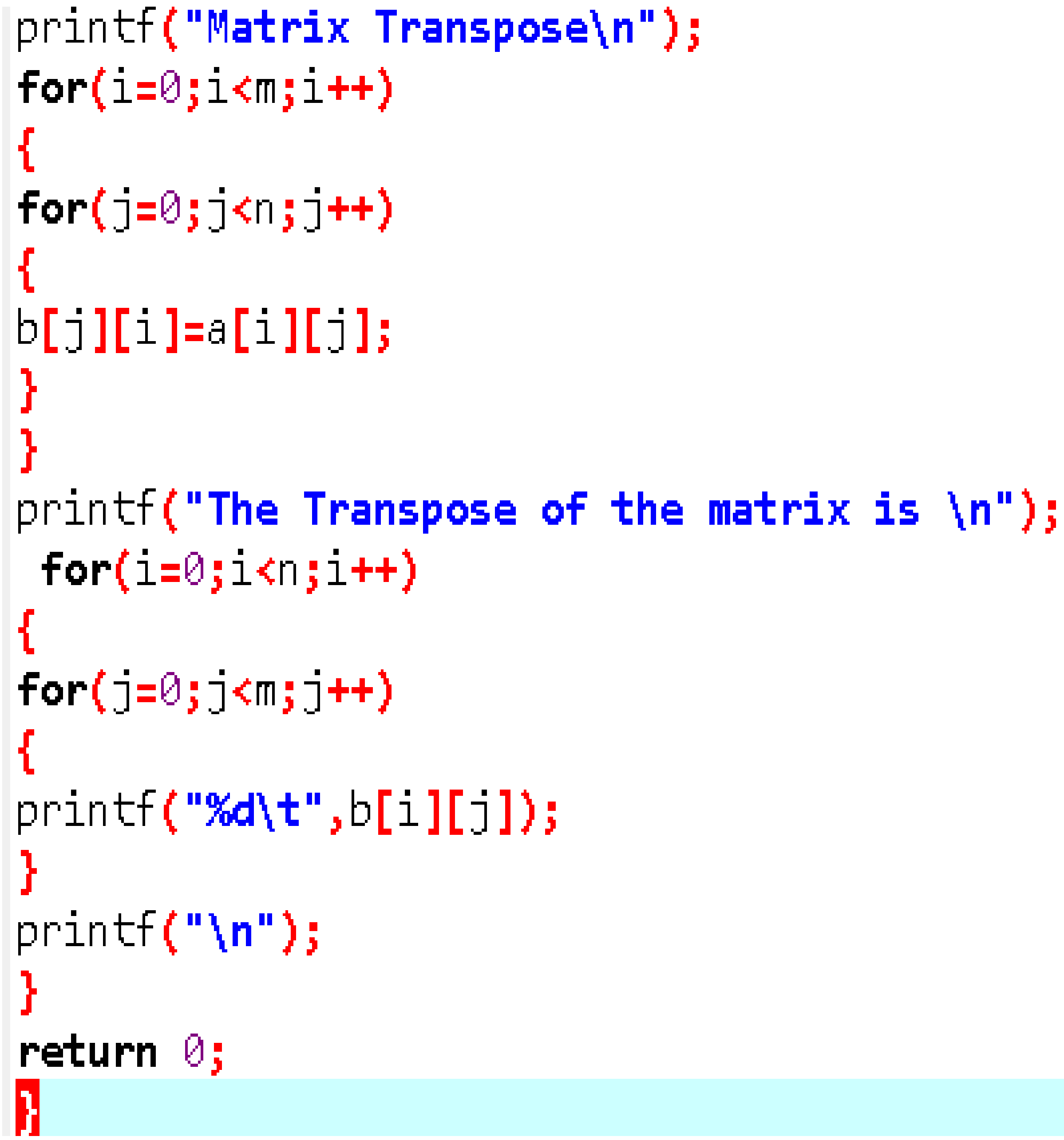
Write a c program to add two matrices



```
enter the rows and column of matrix
3
3
Enter the elements of Matrix A
1 2 1
2 3 4
1 2 3
Enter the elements of Matrix B
4 3 2
1 2 3
1 1 1
Matrix Addition
The resultant matrix is
5       5       3
3       5       7
2       3       4
```

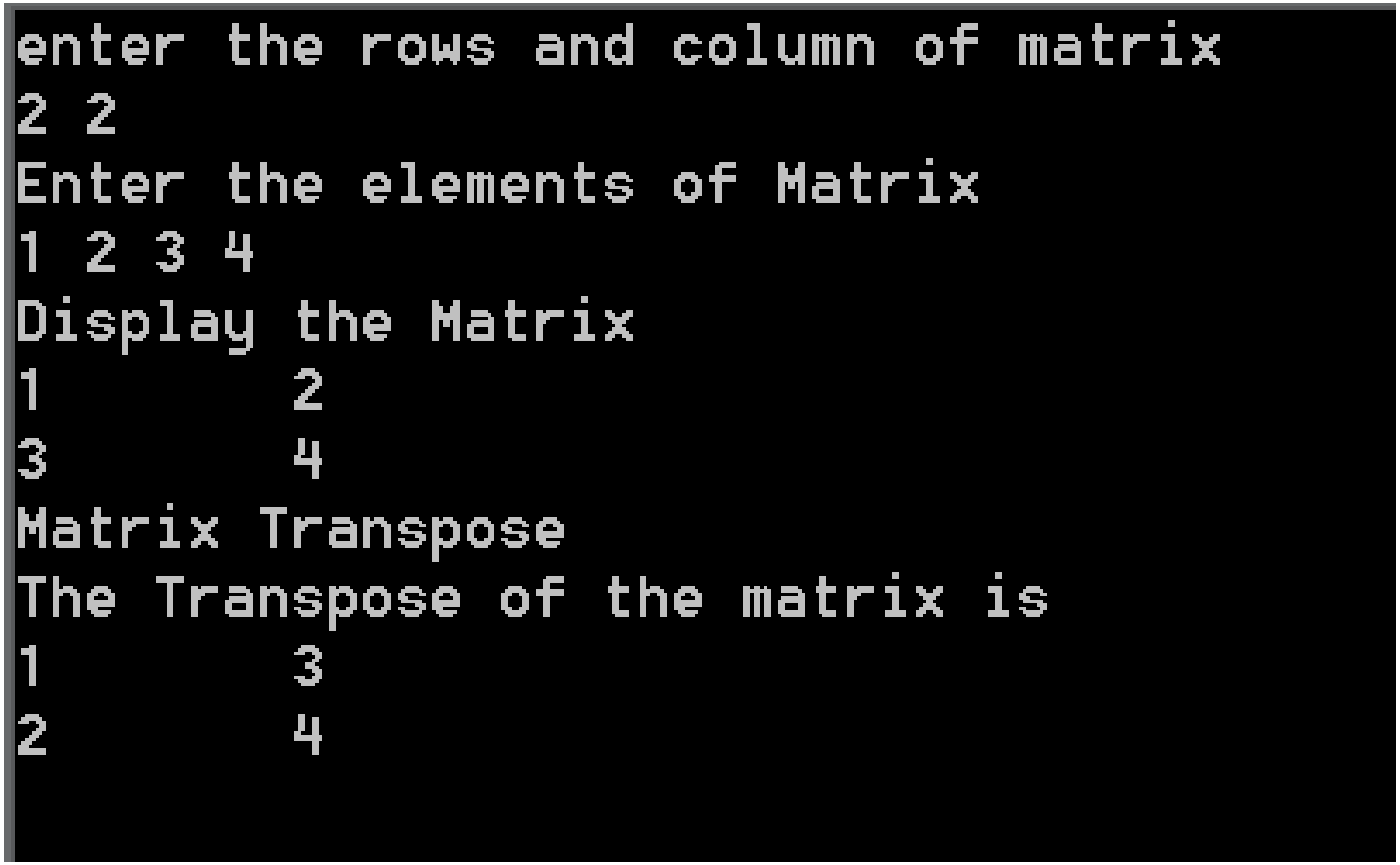
3. Write a C Program to find Transpose of matrix

```
include<stdio.h>
void main()
{
        int a[20 ][20],b[20][20], m,n, i,j;
        printf(“enter the rows and column of matrix\n”);
        scanf(“%d%d”,&m,&n);
        printf(“Enter the elements of Matrix\n”);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf(“%d”,&a[i][j]);
                }
        }
        printf(“Enter the elements of Matrix are\n”);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf(“%d\t”,a[i][j]);
                }
                printf(“\n”);
        }
```



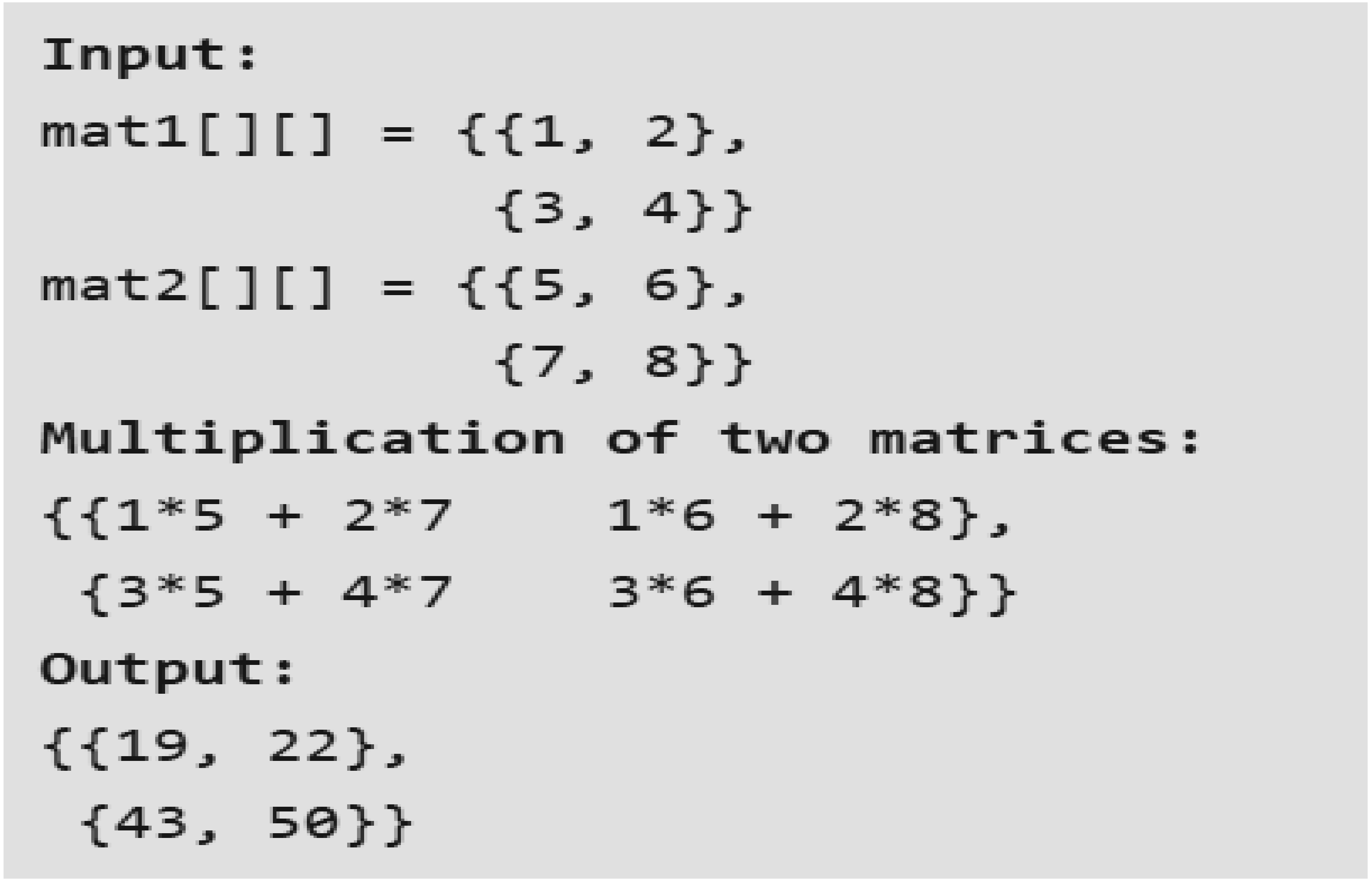
```
printf("Matrix Transpose\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
b[j][i]=a[i][j];
}
}
printf("The Transpose of the matrix is \n");
 for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
return 0;
}
```

3. Write a C Program to find Transpose of matrix

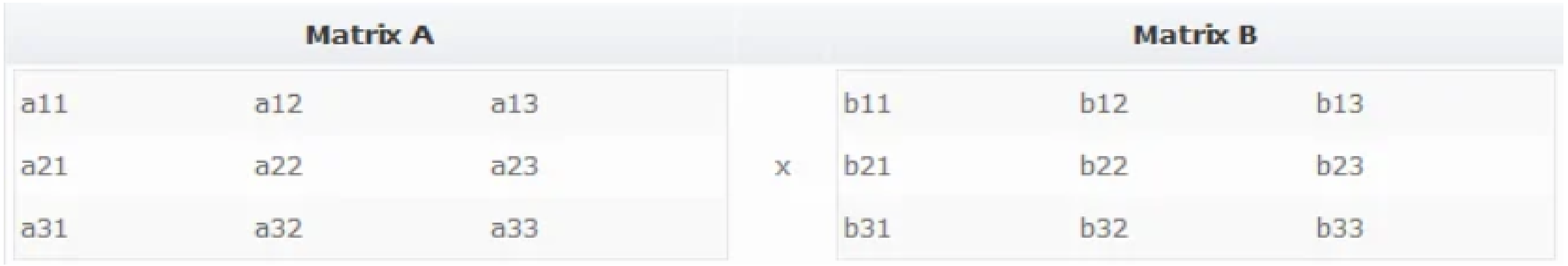


```
enter the rows and column of matrix
2 2
Enter the elements of Matrix
1 2 3 4
Display the Matrix
1       2
3       4
Matrix Transpose
The Transpose of the matrix is
1       3
2       4
```

Write a C program to read 2 matrices of size m*n and p*q. Compute the matrix multiplication by considering the rules for the same and print appropriate result.

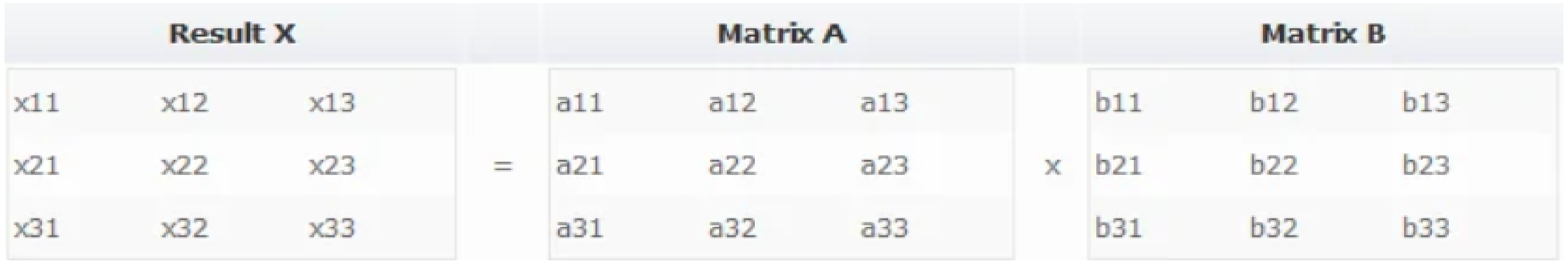


```
Input:
mat1[][] = {{1, 2},
            {3, 4}}
mat2[][] = {{5, 6},
            {7, 8}}
Multiplication of two matrices:
{{1*5 + 2*7    1*6 + 2*8},
 {3*5 + 4*7    3*6 + 4*8}}
Output:
{{19, 22},
 {43, 50}}
```



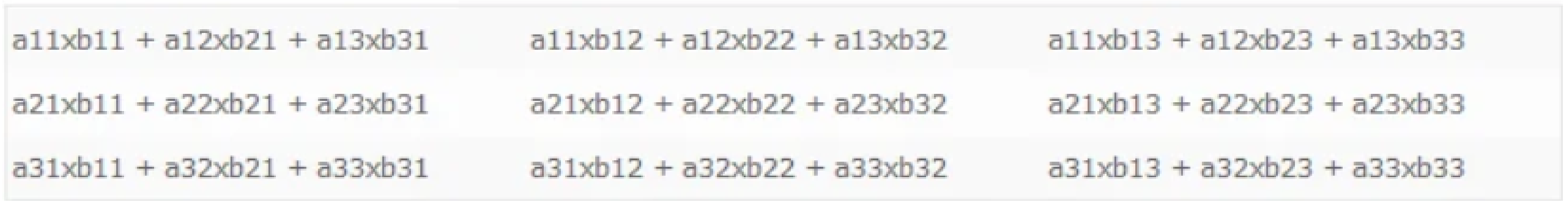
Matrix A				Matrix B		
a11	a12	a13		b11	b12	b13
a21	a22	a23	x	b21	b22	b23
a31	a32	a33		b31	b32	b33

Let the resultant matrix upon multiplication of A and B be X with elements denoted by x*ij* as shown.



Result X				Matrix A				Matrix B		
x11	x12	x13		a11	a12	a13		b11	b12	b13
x21	x22	x23	=	a21	a22	a23	x	b21	b22	b23
x31	x32	x33		a31	a32	a33		b31	b32	b33

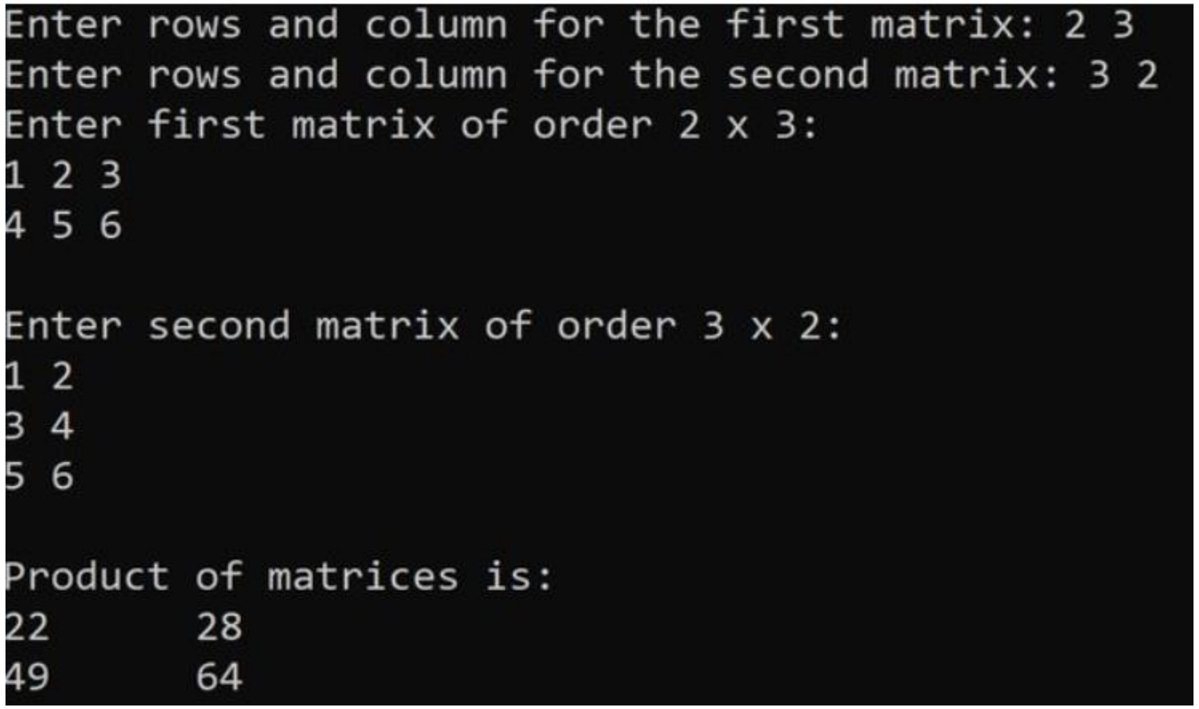
The matrix multiplication takes place as shown below, and this same procedure is is used for multiplication of matrices using C.



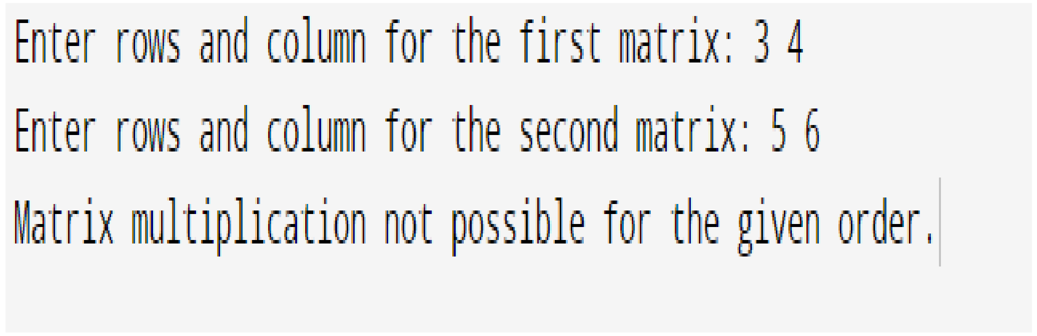
a11xb11 + a12xb21 + a13xb31	a11xb12 + a12xb22 + a13xb32	a11xb13 + a12xb23 + a13xb33
a21xb11 + a22xb21 + a23xb31	a21xb12 + a22xb22 + a23xb32	a21xb13 + a22xb23 + a23xb33
a31xb11 + a32xb21 + a33xb31	a31xb12 + a32xb22 + a33xb32	a31xb13 + a32xb23 + a33xb33

```
#include <stdio.h>
void main()
{
    int i, j, k, r1, c1, r2, c2;
    printf("Enter rows and column for the first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and column for the second matrix: ");
    scanf("%d %d", &r2, &c2);

    if(c1==r2)
    {
      int a[r1][c1], b[r2][c2], c[r1][c2];

      printf("Enter first matrix of order %d x %d:\n",r1,c1);
      for(i=0; i<r1; i++)     for(j=0; j<c1; j++)     scanf("%d",&a[i][j]);

            printf("\nEnter second matrix of order %d x %d:\n",r2,c2);
      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(i=0; i<r1; i++)     for(j=0; j<c2; j++) for(k=0; k<c1; k++) c[i][j] += a[i][k] * b[k][j];

      printf("\nProduct of matrices is:\n");
      for(i=0; i<r1; i++)     {
                  for(j=0; j<c2; j++)     {
                        printf("%d\t",c[i][j]);
                  }
                  printf("\n");
            }
      }
      else {
            printf("Matrix multiplication not possible for the given order.");
      }
}
```

Output



```
Enter rows and column for the first matrix: 2 3
Enter rows and column for the second matrix: 3 2
Enter first matrix of order 2 x 3:
1 2 3
4 5 6

Enter second matrix of order 3 x 2:
1 2
3 4
5 6

Product of matrices is:
22      28
49      64
```



```
Enter rows and column for the first matrix: 3 4
Enter rows and column for the second matrix: 5 6
Matrix multiplication not possible for the given order.
```

Strings

Strings

- String constant or a string literal can be defined as a **sequence of characters** enclosed in double quotes that will be treated as a single data element followed by a null character **‘\0’ (Null character indicates the end of string).**

- Syntax: **char string_name[size];**

  Where,

  char is the data type of strings

  string_name is a valid identifier which is the name for the string variable size indicates the length of the string which is to be stored.

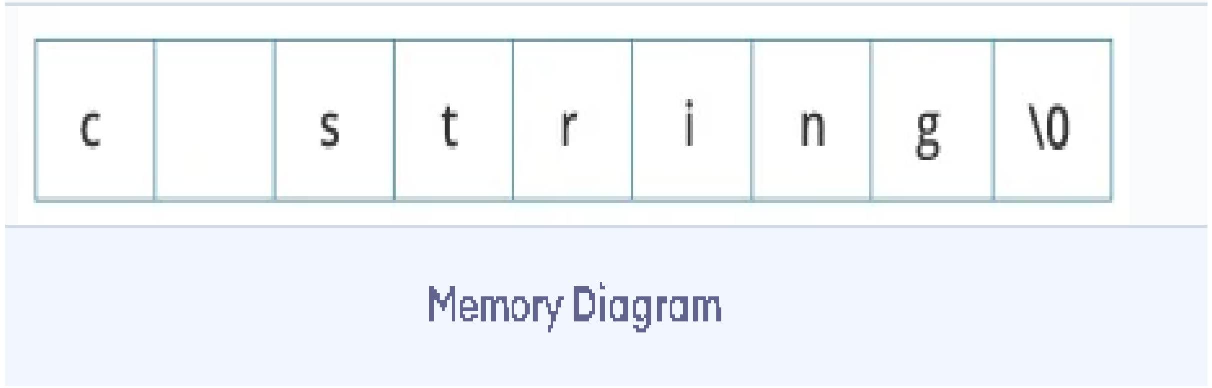
Strings

- Syntax: **char string_name[size];**
- There is no other data type for String variable.
- Strings are treated as arrays of type **char**
- **So, the variable which is used store an array of characters is called String variable.**

Initialization of strings

- **char str1[10]= “Ajay”; or**
- **char str1[10]={‘A’,‘j’,‘a’,‘y’};**
- Both the initializations mentioned are same
- if we directly specify the entire string at a time we use **“ ”**
- if we specify the characters separately we should use **‘ ’,** for each character and finally enclosing all the characters within **{ }**

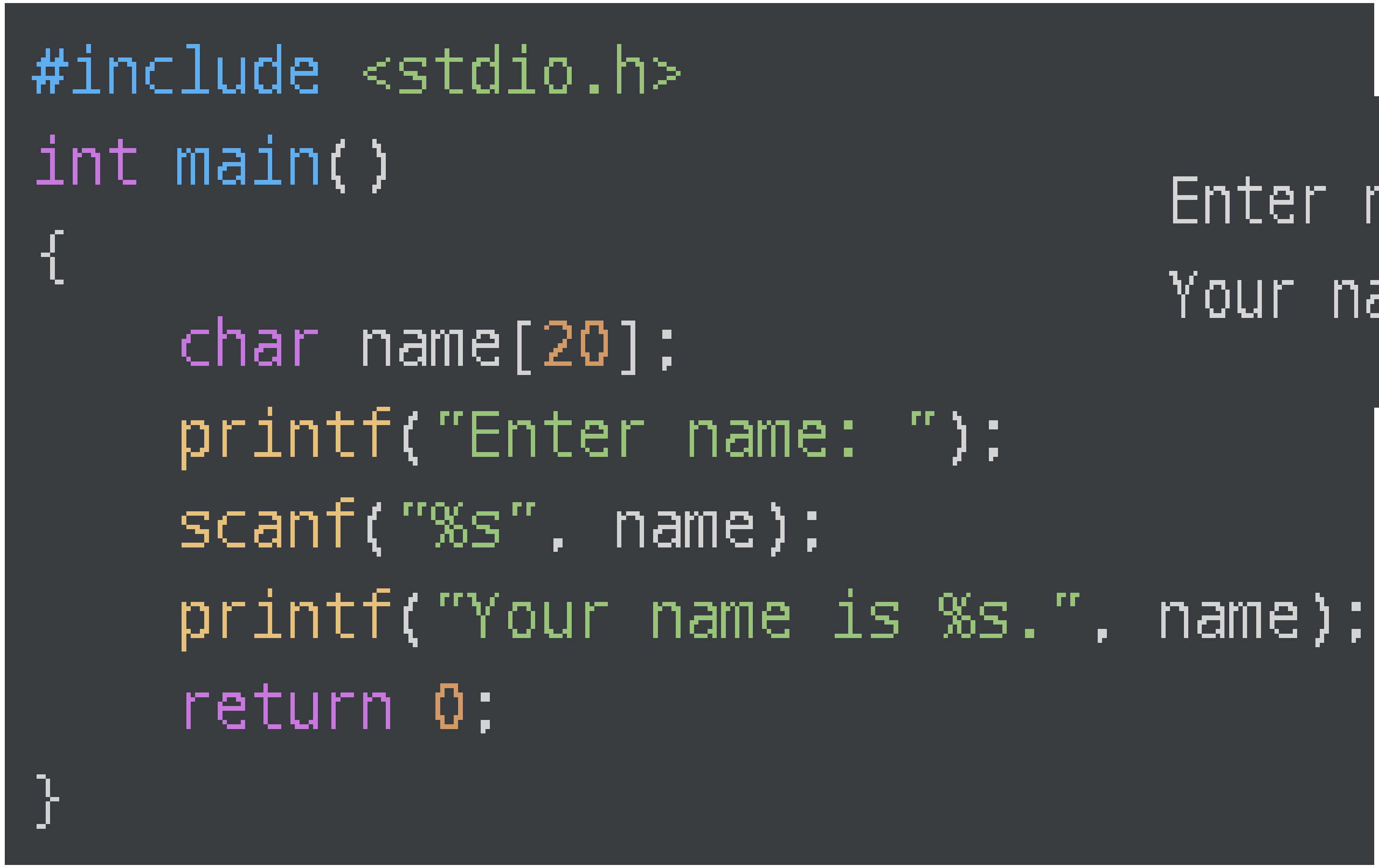
Initialization of strings



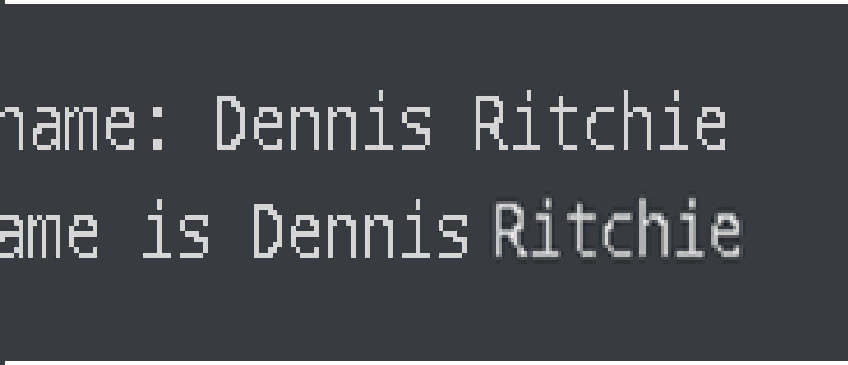
- **char str1[10]= “string”; or**
- **char str1[10]={‘s’,‘t’,‘r’,‘i’,‘n’,g’};**
- Both the initializations mentioned are same
- if we directly specify the entire string at a time we use **“ ”**
- if we specify the characters separately we should use **‘ ’,** for each character and finally enclosing all the characters within **{ }**

Reading a string value using Formatted input and output

scanf() and printf()



```c
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```



```
Enter name: Dennis Ritchie
Your name is Dennis Ritchie
```

Array of Strings/Multi dimensional Strings

The two dimensional array of strings is an array of one dimensional character array which consist of strings as its individual elements.

Syntax:

char string_name[size1][size2];

where, char is a datatype of strings

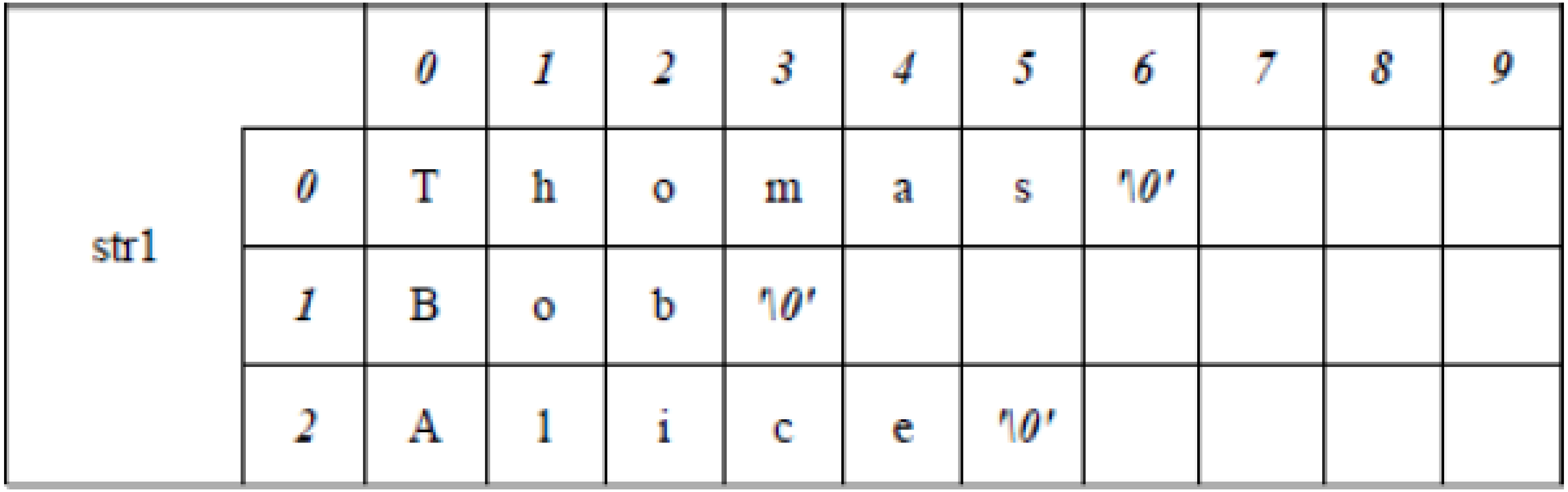
string_name is a valid identifier which is the name of the string variable

size1 indicates the **number of strings** in the array

size 2 indicates the **maximum length** of each string.

Static Initialization of strings:-

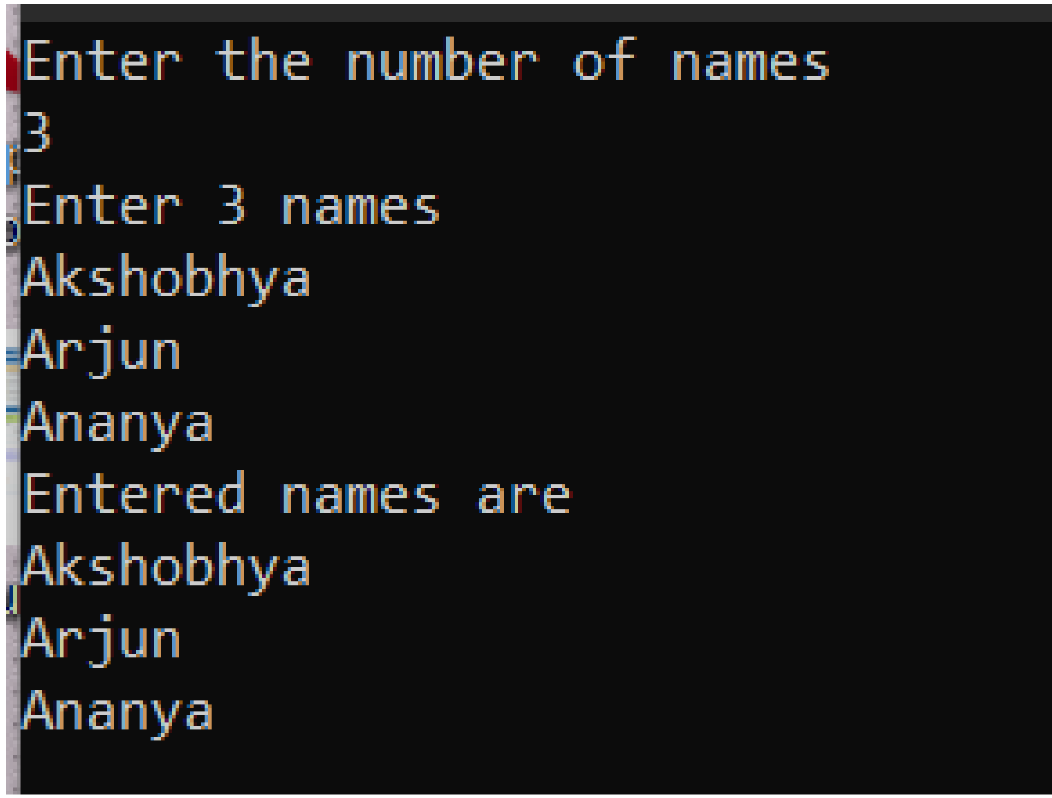
char str1[3][10]={“Thomas”,“Bob ”,“Alice”};



		0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
str1	*0*	T	h	o	m	a	s	'\0'			
	1	B	o	b	'\0'						
	2	A	l	i	c	e	'\0'				

Dynamic Initialization of strings:-

```
#include<stdio.h>
void main()
{
char str[50][50];
int n,i;
printf("Enter the number of names\n");
scanf("%d",&n);
printf("Enter %d names\n",n);
for(i=0;i<n;i++)
{   scanf("%s",str[i]);
}
printf("Entered names are \n");
for(i=0;i<n;i++)
{    printf("%s\n",str[i]);

}
}
```



```
Enter the number of names
3
Enter 3 names
Akshobhya
Arjun
Ananya
Entered names are
Akshobhya
Arjun
Ananya
```

String Manipulating Functions:-

C supports different string handling functions to perform different operations on the strings. All the string handling functions are stored in the header file **“#include<string.h>”.**

Some of the most common used built-in string manipulation functions are

- **strlen() :** Returns the number of character in the given string
- **strcmp() :** Compares two string for their similarity
- **strcpy() :** Copies source string into destination string variable
- **strcat() :** Concatenates (joins) two string into single string
- **strrev() :** Reverses a given string
- **strupr() :** Converts characters to upper case
- **strlwr() :** Converts characters to lower case

String Manipulating Functions:-

String Length :- The function **strlen()** is used to find the length of the string in terms of number of characters in it.

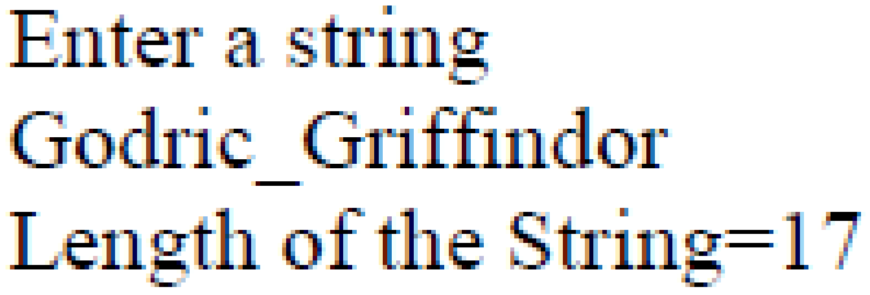
Syntax: **strlen(string_data);**

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str1[50];
        int len;
        printf("Enter a string\n");
        scanf("%s",str1);
        len=strlen(str1);
        printf("Length of the String=%d\n",len);
}
```

```
Enter a string
mangalore
Length of the String=9
```

C program to find length of the string with out using strlen function

```
#include<stdio.h>
 #include<string.h>
 void main()
{
char str1[50];
 int i,len;
printf("Enter a string\n");
scanf("%s",str1);
len=0;
 for(i=0;str1[i]!='\0';i++)
{
len=len+1;
}
printf("Length of the String=%d\n",len);
}
```



```
Enter a string
Godric_Griffindor
Length of the String=17
```

String Manipulating Functions:-

String Compare:- The function **strcmp()** is used to compare the string data every character of one string is compared with the corresponding position character of second string

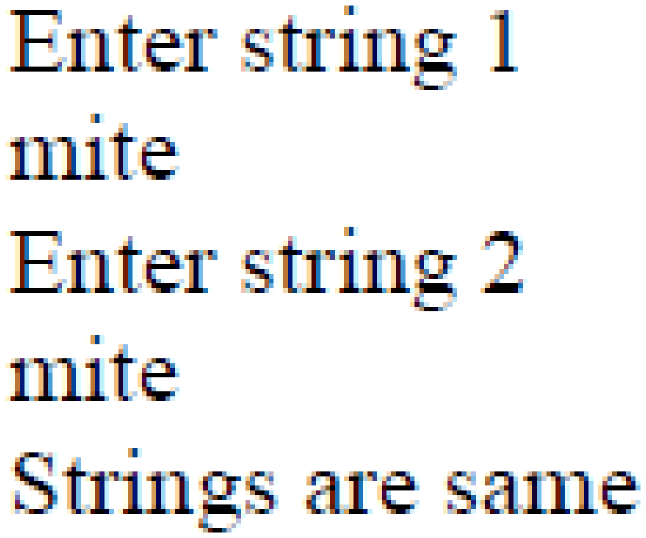
Syntax: **strcmp(str1,str2)**

The function returns 0 if there is complete match (str1==str2)

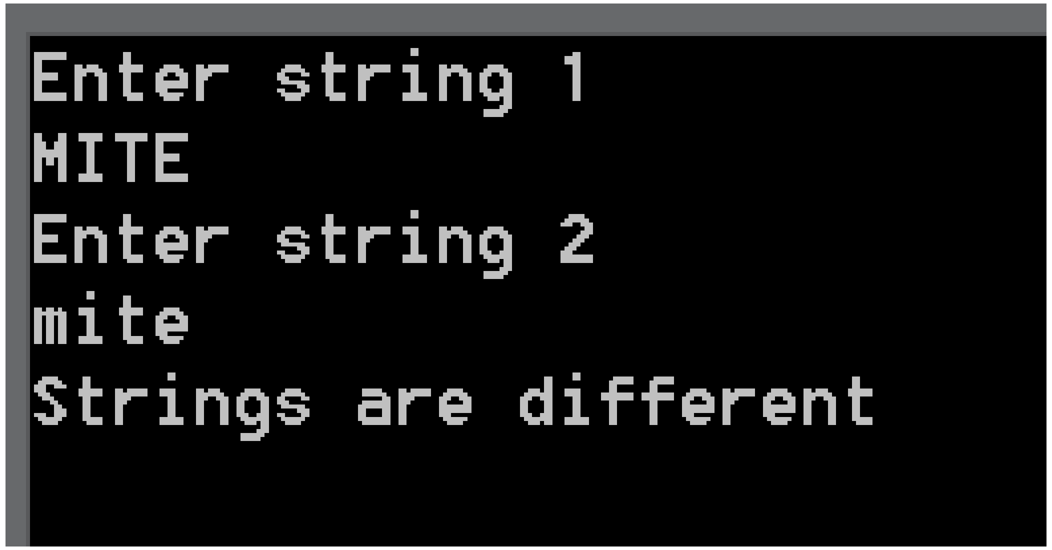
- Returns positive value if str1>str2
- Returns negative value if str1<str2

C program to compare two strings using strcmp( ) function

```
#include<stdio.h>
 #include<string.h>
 void main()
{
char str1[20],str2[20];
 int k;
printf("Enter string 1\n");
scanf("%s",str1);
printf("Enter string 2\n");
scanf("%s",str2);
 k=strcmp(str1,str2);
if(k==0)
printf("Strings are same\n");
else
   printf("Strings are different\n");
}
```



```
Enter string 1
mite
Enter string 2
mite
Strings are same
```

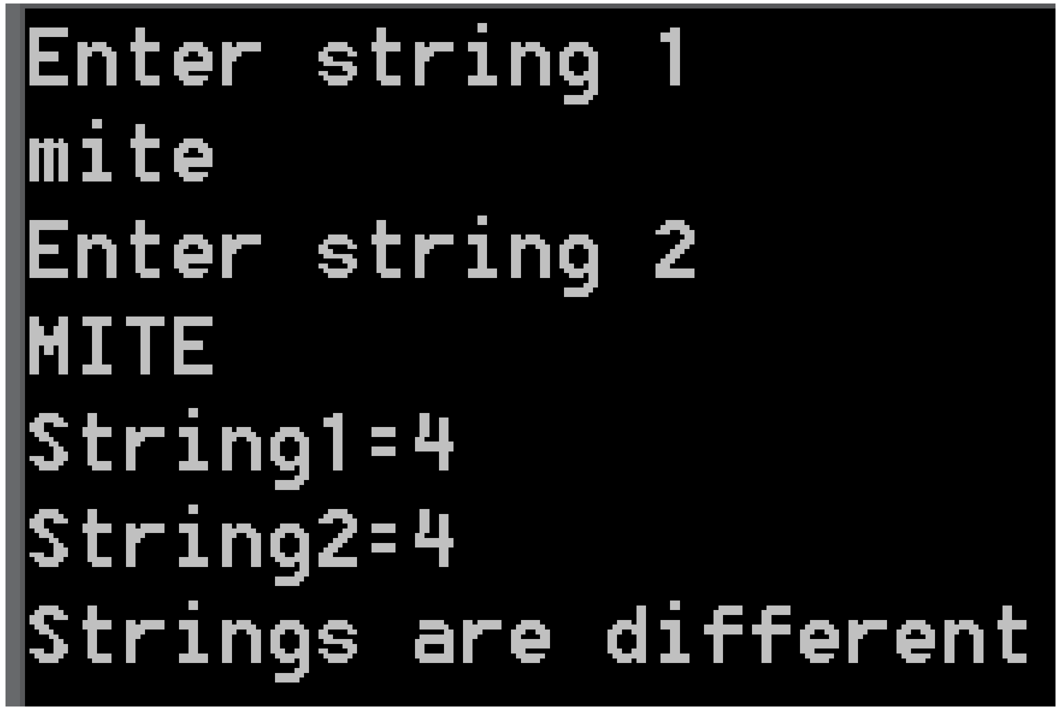


```
Enter string 1
MITE
Enter string 2
mite
Strings are different
```

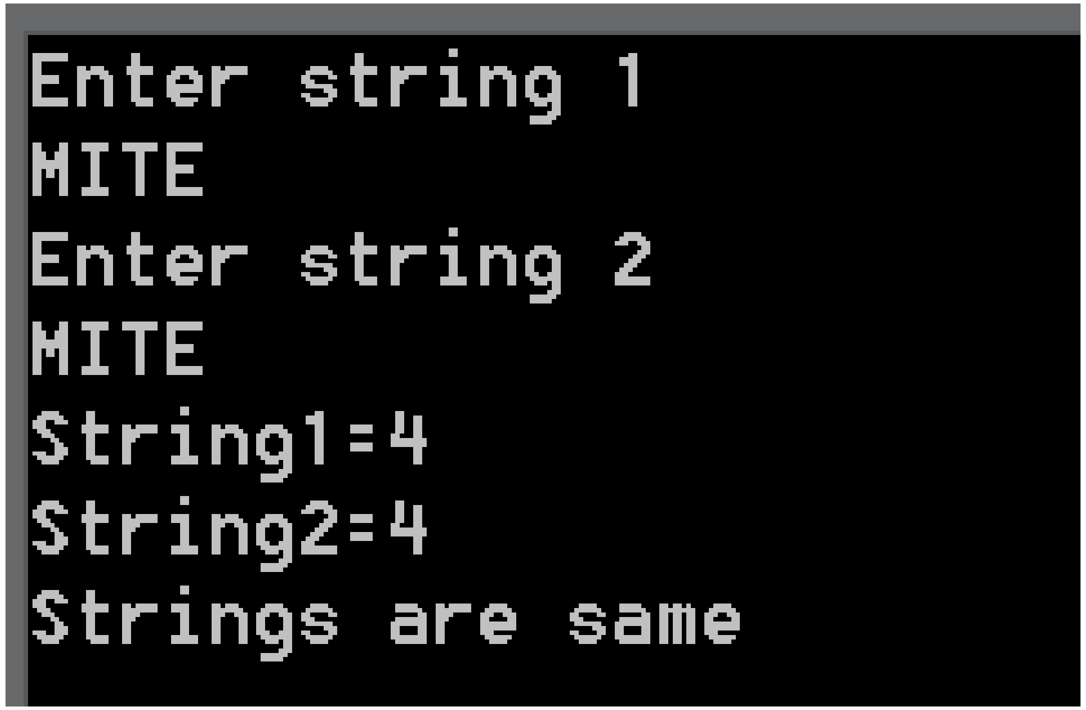
C program to compare two strings without using strcmp( ) function

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
char str1[20],str2[20]; int len1,len2,i;
printf("Enter string 1\n");
 scanf("%s",str1);
 printf("Enter string 2\n");
 scanf("%s",str2);
 len1=strlen(str1);
 len2=strlen(str2);
 printf("String1=%d\n",len1);
  printf("String2=%d\n",len2);
 if(len1!=len2)
printf("Strings are different\n");
else
{
for(i=0;str1[i]!='\0';i++)
{
if(str1[i]!=str2[i])
{
printf("Strings are different\n");
exit(0);
}
}
printf("Strings are same\n");
}
return 0;
}
```

C program to compare two strings without using strcmp( ) function



```
Enter string 1
mite
Enter string 2
MITE
String1=4
String2=4
Strings are different
```

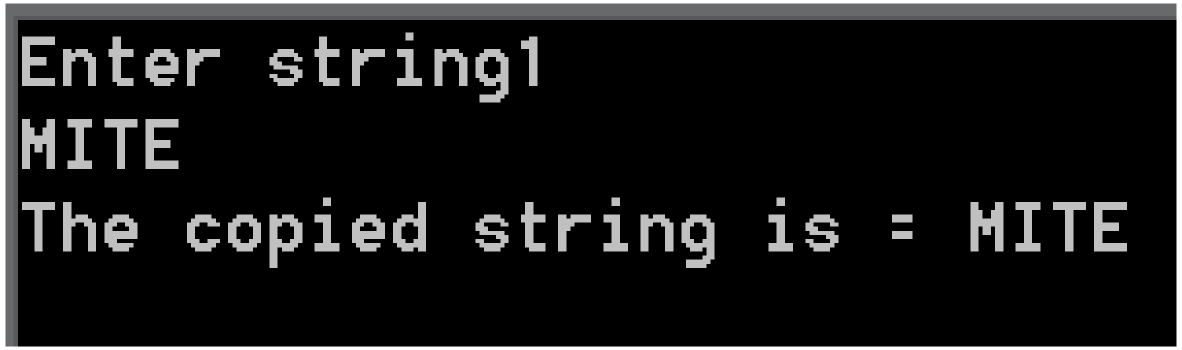


```
Enter string 1
MITE
Enter string 2
MITE
String1=4
String2=4
Strings are same
```

String Manipulating Functions:-

String Copy:- The function **strcpy()** copies the content from one string to another string Syntax: **strcpy(str2,str1);**

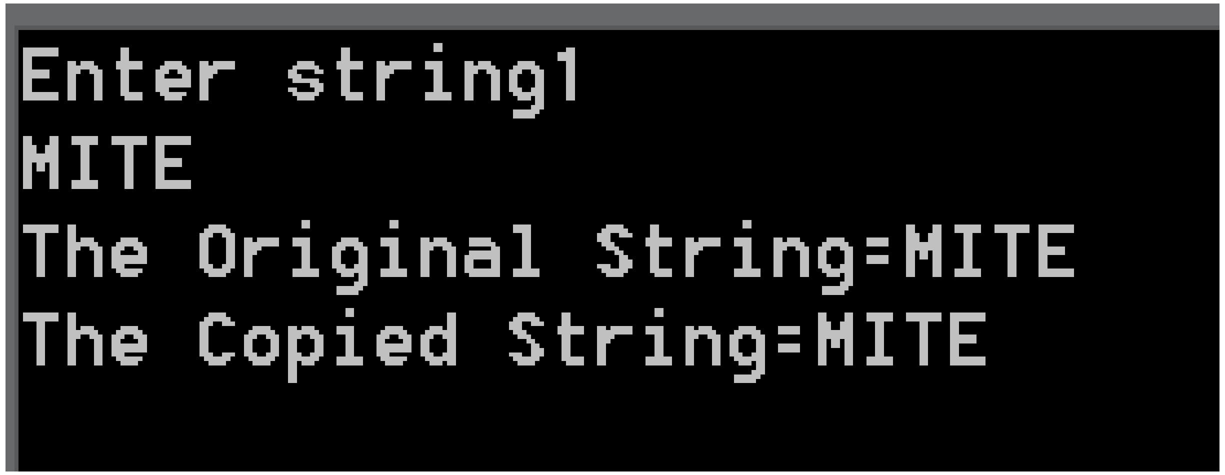
```
#include<stdio.h>
#include<string.h>
void main()
{
char str1[30],str2[30];
printf("Enter string1\n");
scanf("%s",str1);
 strcpy(str2,str1);
printf("The copied string is = %s\n",str2);
}
```



```
Enter string1
MITE
The copied string is = MITE
```

program to copy the string without using strcpy( ) function

```c
#include<stdio.h>
#include<string.h>
int main()
{
char str1[30],str2[30];
 int i=0;
printf("Enter string1\n");
 scanf("%s",str1);
while(str1[i]!='\0')
{
str2[i]=str1[i]; i++;
}
str2[i]='\0';
printf("The Original String=%s\n",str1);
 printf("The Copied String=%s\n",str2);
 return 0;
}
```



```
Enter string1
MITE
The Original String=MITE
The Copied String=MITE
```

String Manipulating Functions:-

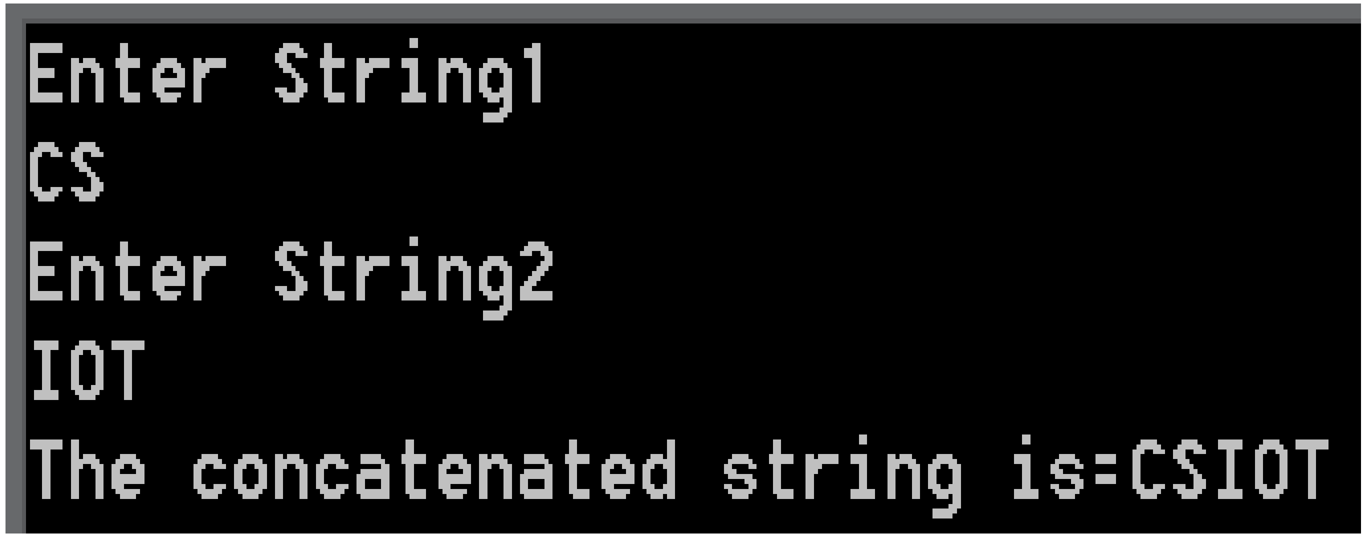
String Concatenate :- The function **strcat()** is used to concatenate (attach) two strings.

Syntax: **strcat(str1,str2);**

string str2 is attached to the end of string str1

C program to concatenate two strings using strcat function

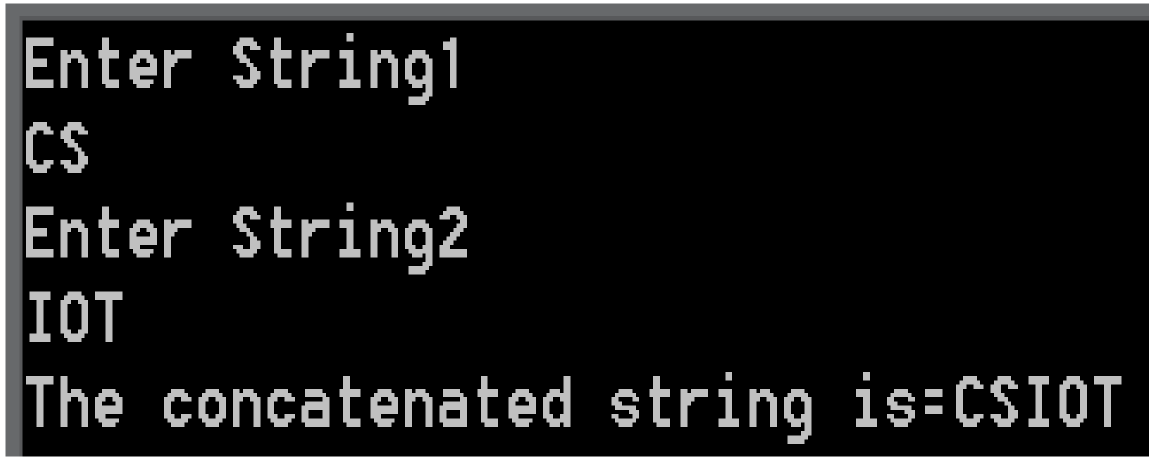
```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
char str1[30],str2[30];
printf("Enter String1\n");
scanf("%s",str1);
printf("Enter String2\n");
scanf("%s",str2);
 strcat(str1,str2);
printf("The concatenated string is=%s\n",str1);
return 0;
}
```



```
Enter String1
CS
Enter String2
IOT
The concatenated string is=CSIOT
```

C program to concatenate two strings without using strcat function

```c
#include<stdio.h>
#include<string.h>
int main()
{
char str1[20],str2[20],str3[50];
int i,k;
printf("Enter String1\n");
scanf("%s",str1);
 printf("Enter String2\n");
 scanf("%s",str2);
k=0;
for(i=0;str1[i]!='\0';i++)
{
str3[k]=str1[i];
 k=k+1;
}
for(i=0;str2[i]!='\0';i++)
{
str3[k]=str2[i];
 k=k+1;
}
str3[k]='\0';
printf("The concatenated string is=%s\n",str3);
return 0;
}
```



```
Enter String1
CS
Enter String2
IOT
The concatenated string is=CSIOT
```

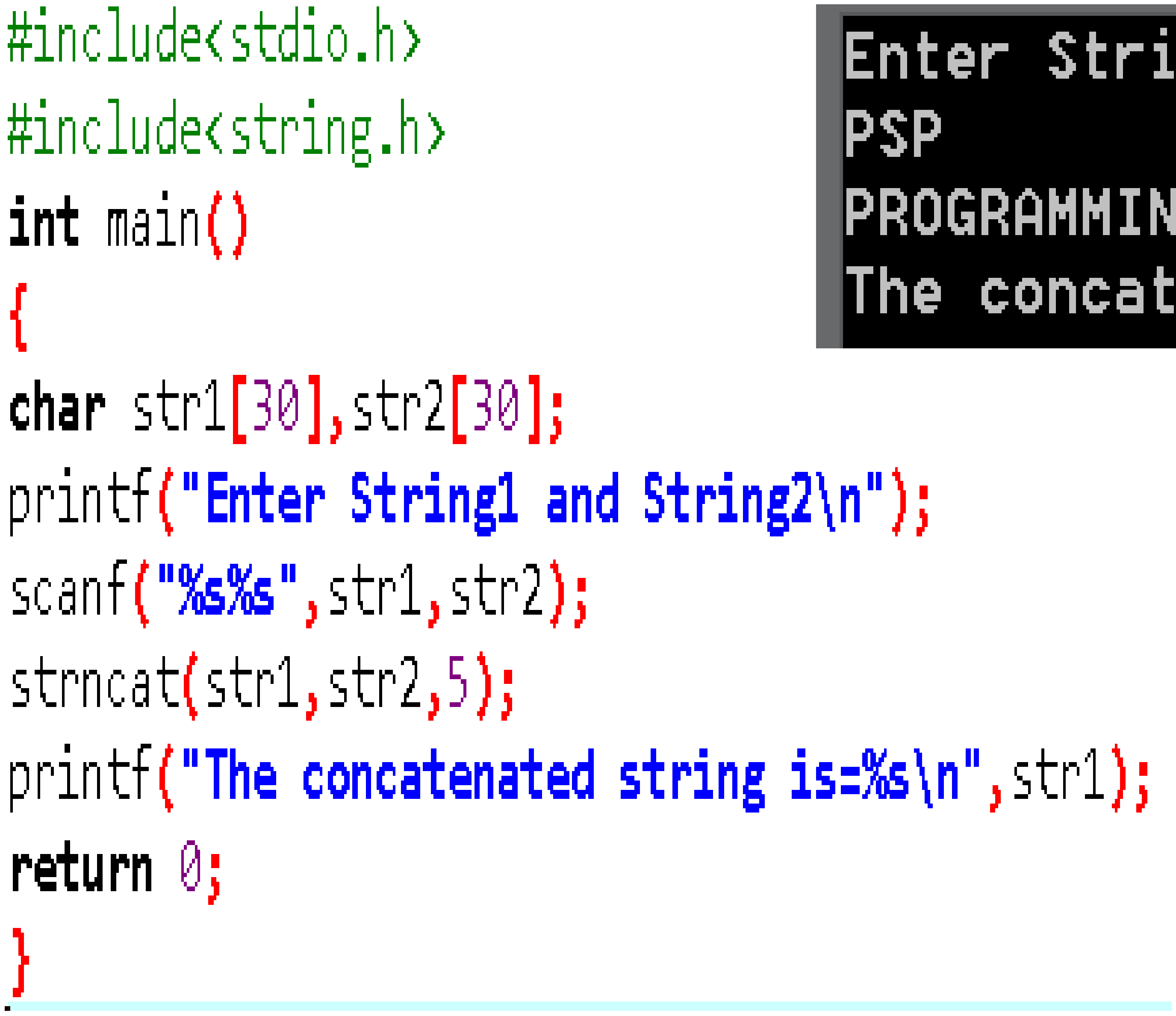
String Manipulating Functions:-

String n Concatenate: - The function **strncat()** is used to concatenate the specified number of characters only.

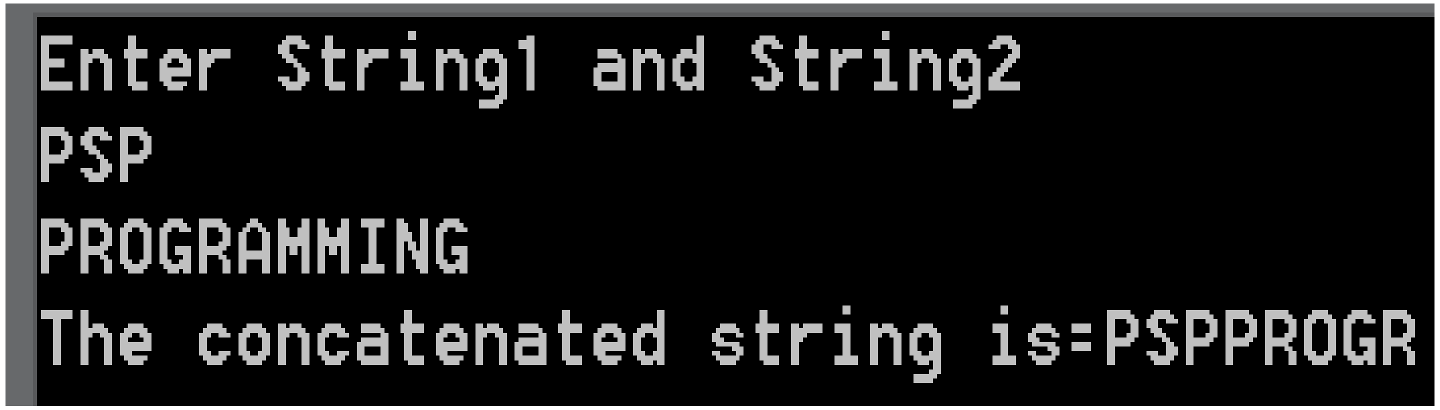
Syntax: **strncat(str1,str2,n);**

Where **n** is an integer value which concatenates only n characters of string2 to string1

C program to concatenate two strings using strncat function



```c
#include<stdio.h>
#include<string.h>
int main()
{
char str1[30],str2[30];
printf("Enter String1 and String2\n");
scanf("%s%s",str1,str2);
strncat(str1,str2,5);
printf("The concatenated string is=%s\n",str1);
return 0;
}
```



```
Enter String1 and String2
PSP
PROGRAMMING
The concatenated string is=PSPPROGR
```

String Manipulating Functions:-

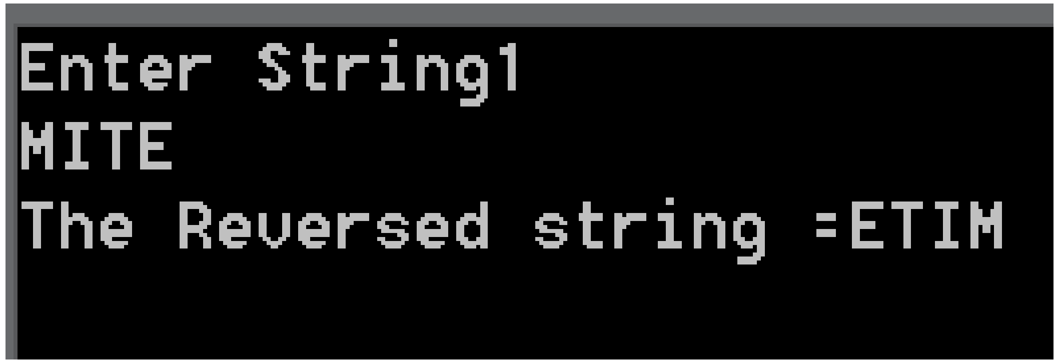
String Reverse:- The function **strrev()** is used to reverse the string .

The characters from left to right in the original string are placed in the reverse order

Syntax: **strrev(str1)**

C program to reverse a string using strrev( ) function

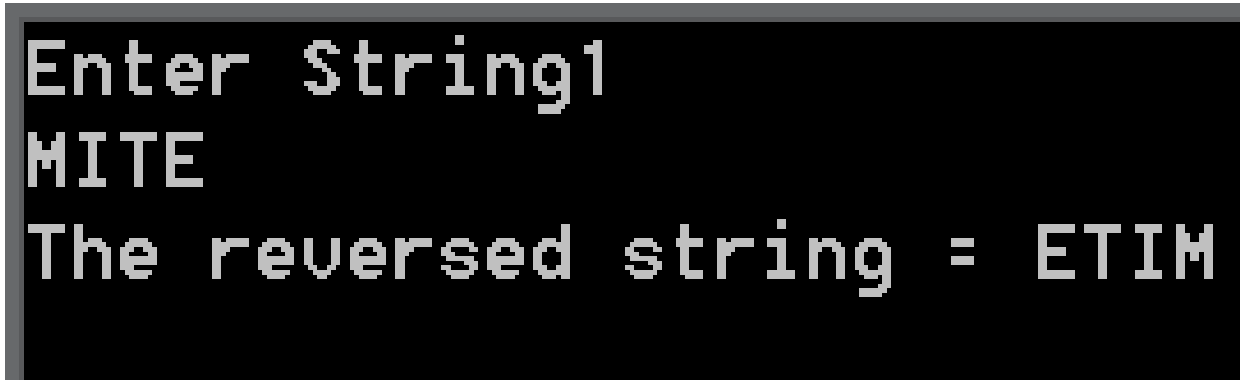
```
#include<stdio.h>
 #include<string.h>
 int main()
{
char str1[30];
 printf("Enter String1\n");
scanf("%s",str1);
strrev(str1);
printf("The Reversed string =%s\n",str1);
return 0;
}
```



```
Enter String1
MITE
The Reversed string =ETIM
```

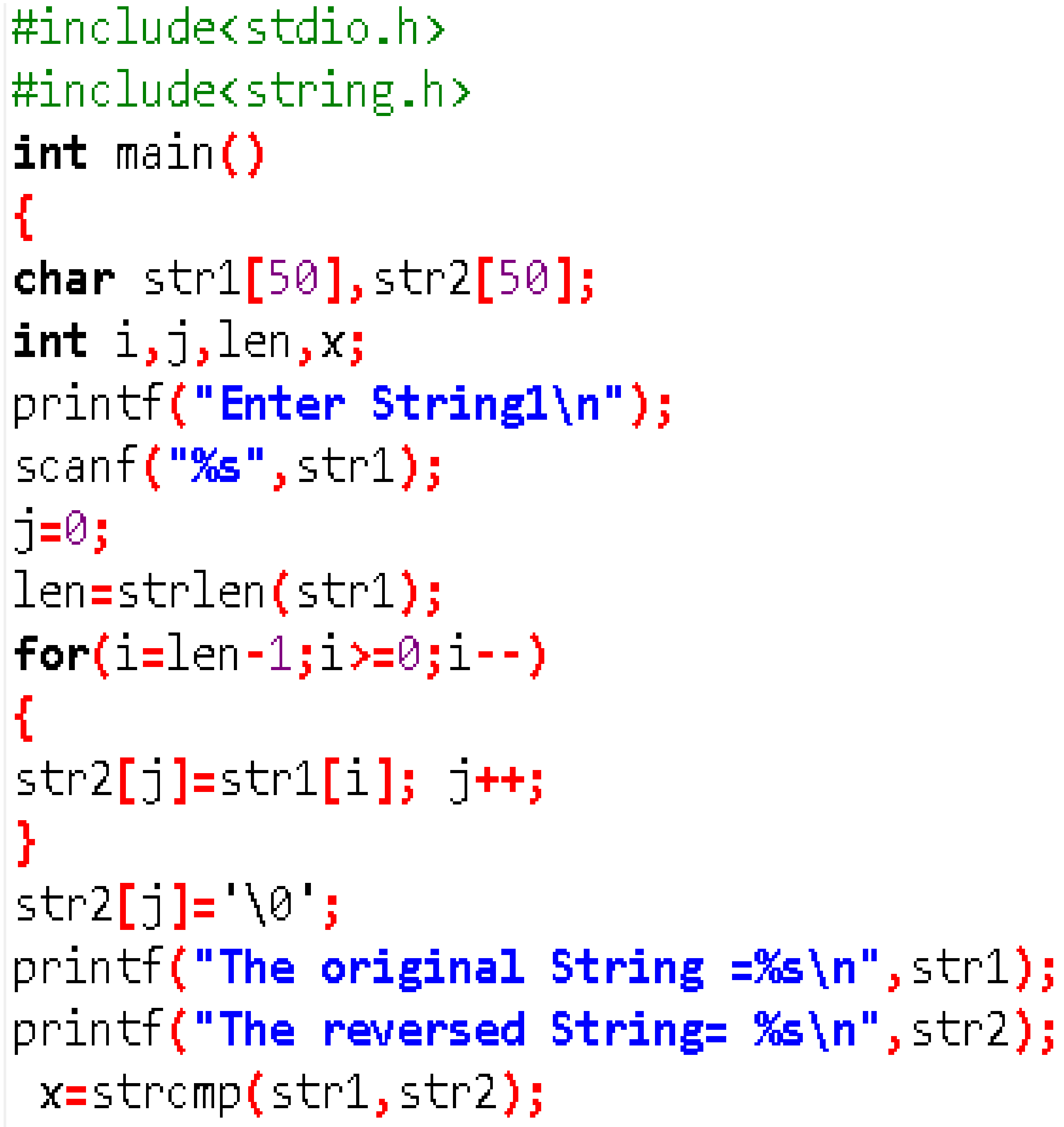
C program to reverse a string without using strrev() function

```c
#include<stdio.h>
#include<string.h>
int main()
{
char str1[50],str2[50];
 int i,j,len;
printf("Enter String1\n");
scanf("%s",str1);
j=0;
len=strlen(str1);
for(i=len-1;i>=0;i--)
{
str2[j]=str1[i]; j++;
}
str2[j]='\0';
printf("The reversed string = %s\n",str2);
return 0;
}
```

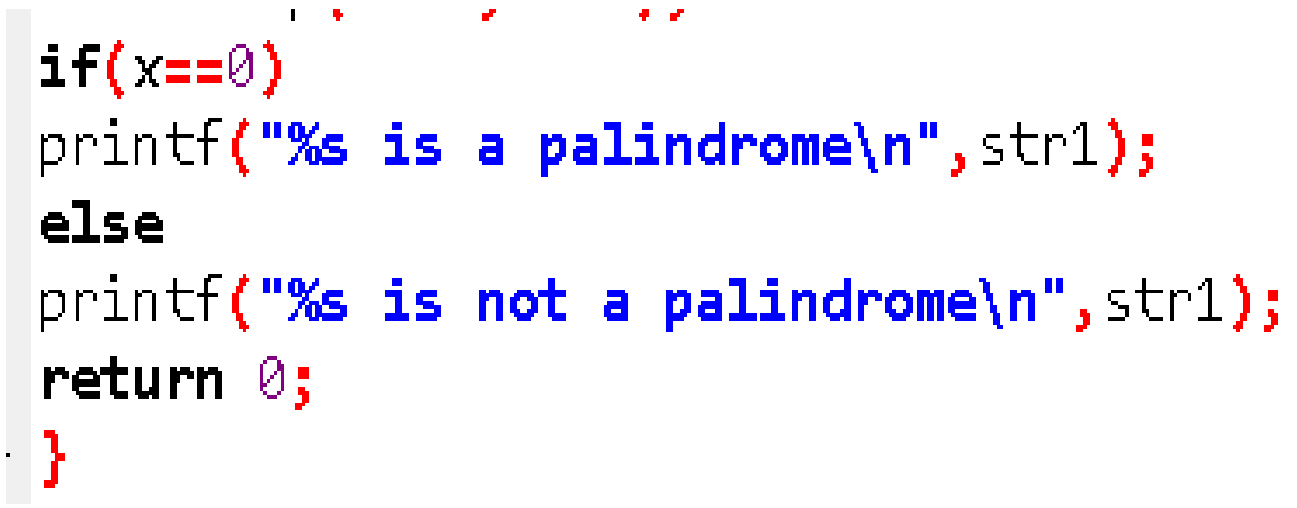


```
Enter String1
MITE
The reversed string = ETIM
```

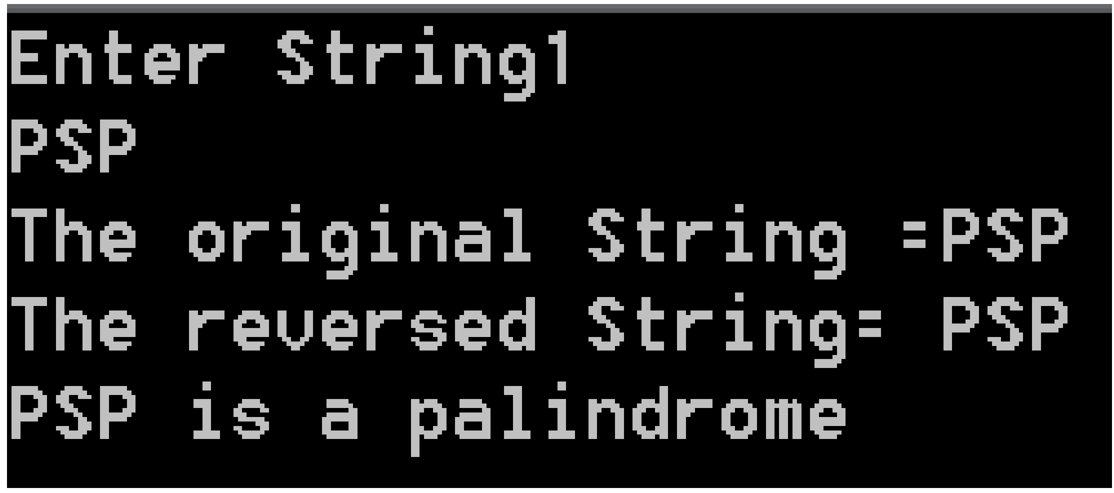
C program to check if the given string is a Palindrome or not a Palindrome



```
#include<stdio.h>
#include<string.h>
int main()
{
char str1[50],str2[50];
int i,j,len,x;
printf("Enter String1\n");
scanf("%s",str1);
j=0;
len=strlen(str1);
for(i=len-1;i>=0;i--)
{
str2[j]=str1[i]; j++;
}
str2[j]='\0';
printf("The original String =%s\n",str1);
printf("The reversed String= %s\n",str2);
 x=strcmp(str1,str2);
```



```
if(x==0)
printf("%s is a palindrome\n",str1);
else
printf("%s is not a palindrome\n",str1);
return 0;
}
```



```
Enter String1
PSP
The original String =PSP
The reversed String= PSP
PSP is a palindrome
```

String Manipulating Functions:-

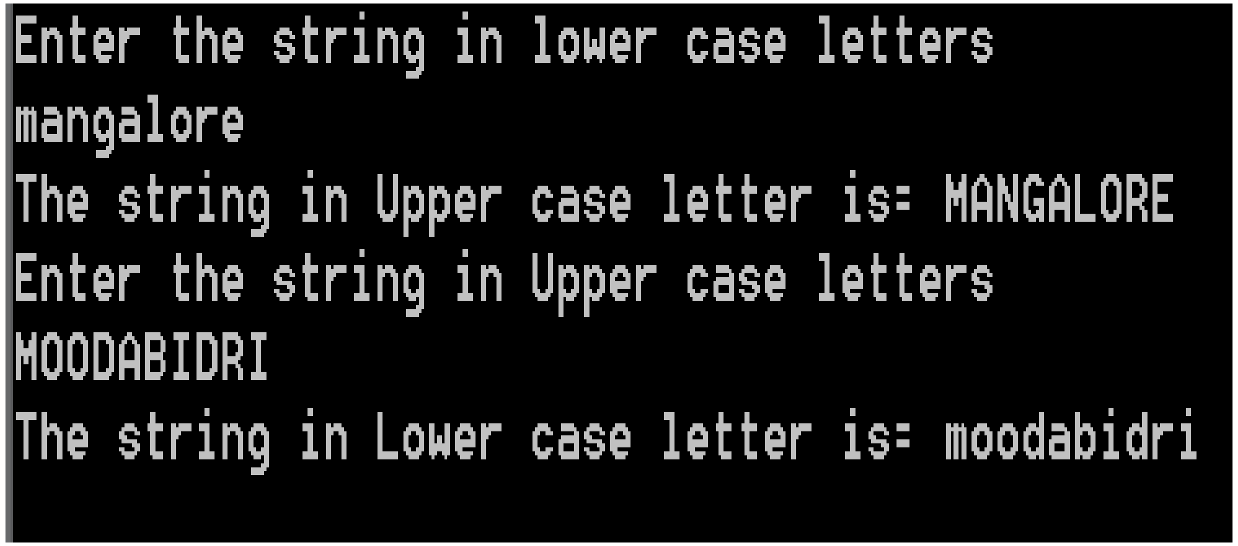
String Lower :- The function **strlwr()** converts each character of the string to lowercase.

SYNTAX : **strlwr(string_data);**

String Upper :- The function **strupr()** converts each character of the string to uppercase.

SYNTAX : **strupr(string_data);**

C program to convert strings to uppercase and lowercase using strupr( ) and strlwr ( ) functions



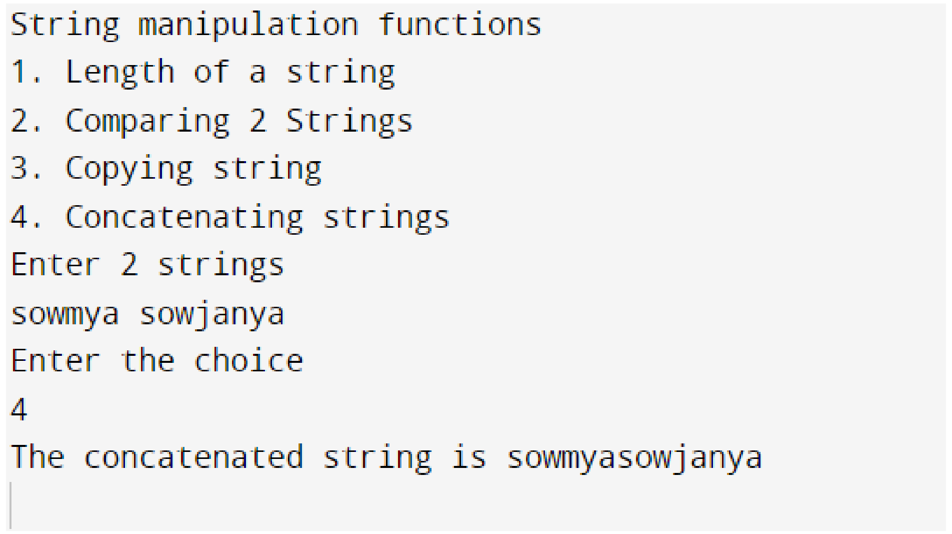
```
Enter the string in lower case letters
mangalore
The string in Upper case letter is= MANGALORE
Enter the string in Upper case letters
MOODABIDRI
The string in Lower case letter is= moodabidri
```

```c
#include<stdio.h>
#include<string.h>
int main()
{
char str1[50],str2[50];
printf("Enter the string in lower case letters\n");
scanf("%s",str1);
printf("The string in Upper case letter is= %s\n",strupr(str1));
printf("Enter the string in Upper case letters\n");
scanf("%s",str2);
printf("The string in Lower case letter is= %s\n",strlwr(str2));
return 0;
}
```

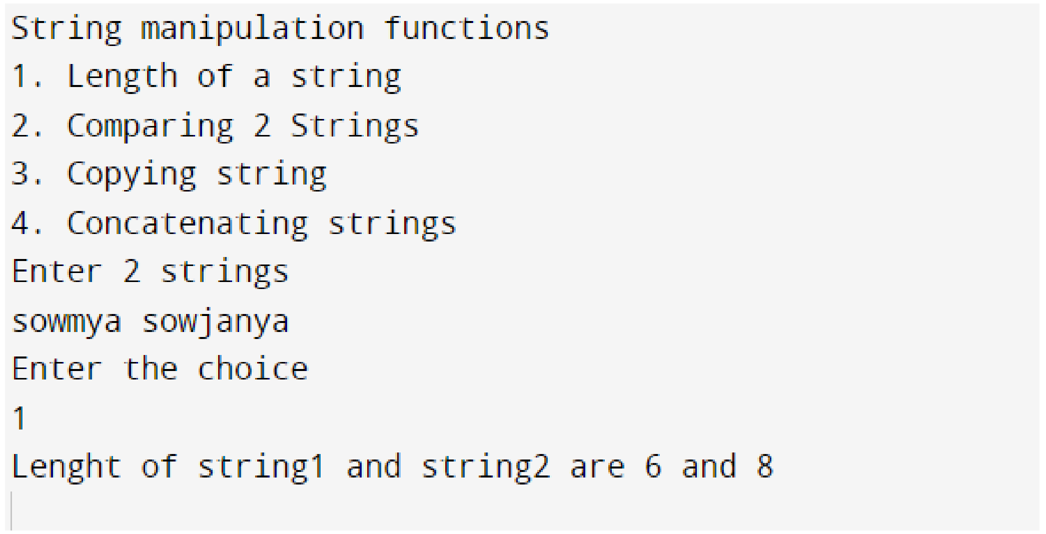
- C program to perform string manipulation functions using switch statement.

```c
#include<stdio.h>
void main()
{   int len1,len2, choice,k;
    char str1[50], str2[50];
    printf("String manipulation functions\n");
    printf("1. Length of a string\n");
    printf("2. Comparing 2 Strings\n");
    printf("3. Copying string\n");
    printf("4. Concatenating strings\n");
    printf("Enter 2 strings\n");
    scanf("%s%s", str1,str2);
    printf("Enter the choice\n");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1:len1=strlen(str1);
               len2=strlen(str2);
               printf("Lenght of string1 and string2 are %d and %d\n", len1,len2);
               break;
        case 2:  k=strcmp(str1,str2);
                 if(k==0)
                   printf("Strings are same\n");
                 else
                   printf("Strings are different\n");
                 break;
        case 3: strcpy(str2,str1);
                printf("After copying string2 contains %s\n", str2);
                break;
        case 4: strcat(str1,str2);
                printf("The concatenated string is %s\n",str1);
                break;
        default: printf("Invalid choice\n");
                break;

    }
}
```

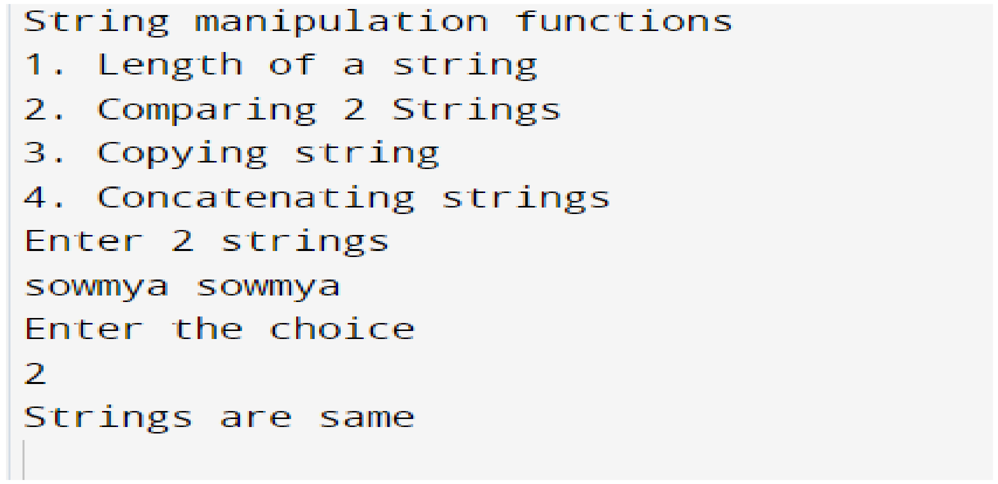
Output



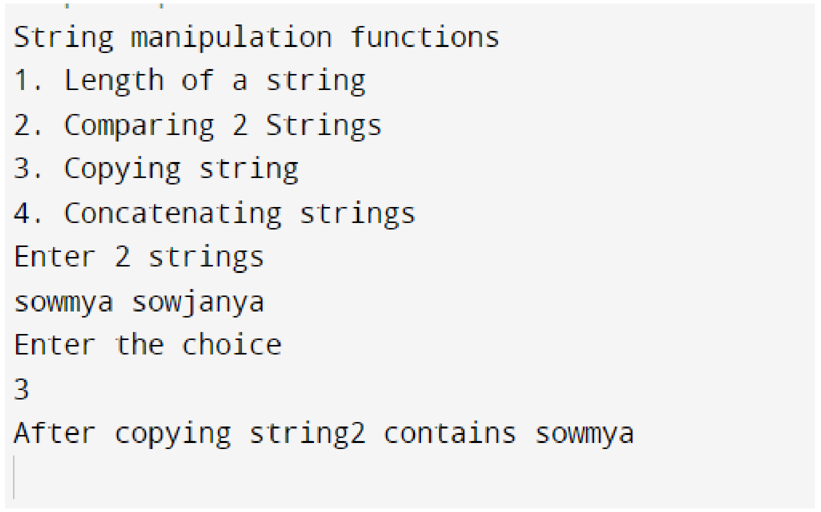
```
String manipulation functions
1. Length of a string
2. Comparing 2 Strings
3. Copying string
4. Concatenating strings
Enter 2 strings
sowmya sowjanya
Enter the choice
4
The concatenated string is sowmyasowjanya
```



```
String manipulation functions
1. Length of a string
2. Comparing 2 Strings
3. Copying string
4. Concatenating strings
Enter 2 strings
sowmya sowjanya
Enter the choice
1
Lenght of string1 and string2 are 6 and 8
```



```
String manipulation functions
1. Length of a string
2. Comparing 2 Strings
3. Copying string
4. Concatenating strings
Enter 2 strings
sowmya sowmya
Enter the choice
2
Strings are same
```



```
String manipulation functions
1. Length of a string
2. Comparing 2 Strings
3. Copying string
4. Concatenating strings
Enter 2 strings
sowmya sowjanya
Enter the choice
3
After copying string2 contains sowmya
```

Other Applications of strings. :-

- The String functions are also used to count the number of vowels and consonants in the given string. And also gives the frequency of occurrence of each vowel in the given string. This program makes use of the header file **“#include<ctype.h>”**which is useful for testing and mapping characters.
- The function **“isalpha()”** is used to check if the parsed character is an alphabet or not.