Chapter-07
Array

This Chapter Includes:

Array:

- An array is defined as a collection of elements of similar data type.
- All the elements of an array are stored in consecutive memory locations i.e. one after the other or in a sequence in main memory.
- All the elements shares common array name and they are accessed using index numbers.

Eg: Consider an array of Marks of 5 student as shown below:

80	90	45	99	100
Marks [0]	Marks [1]	Marks [2]	Marks [3]	Marks [4]

Here, **Marks is name of the array and [0], [1], [2], [3], [4] are the index numbers.**

- To refer an item in the array, we specify the name of the array along with position of the item. The position of the item must be written within square brackets '[]'.
- The position of the item enclosed within square brackets is called 'subscript' or 'index'.
- For example, the above figure represents an integer array called marks where marks of 5 students are stored. The marks of each student can be accessed as shown below:
- marks[0] i.e. 80 - represent marks of first student
- marks[l] i.e. 90 - represent marks of second student
- marks[2] i.e. 45 - represent marks of third student
- marks[3] i.e. 99 - represent marks of fourth student
- marks[4] i.e. 100 - represent marks of fifth student

Thus, using marks[0] through marks[4] we can access the marks of 5 students.
Note: Using marks[0] through marks[n-1] we can access the marks of n students in general.

In an array it is not possible to have a group of items with different data types. For example,

80	"abc"	45.54	'c'	10
Marks [0]	Marks [1]	Marks [2]	Marks [3]	Marks [4]

It is invalid way of storing the elements in an array. This is because, it is a collection of int, float, char and string datatypes.

Properties of Array:

1. Array elements are stored in contiguous memory.
2. Array name represents its base address. The base address is the address of the first element of the array.
3. Array's index starts with 0 and ends with N-1. Here, N stands for the number of elements. For Example, there is an integer array of 5 elements, then it's indexing will be 0 to 4.
4. Array elements are accessed by using an integer index.

Types of Array:

1. Single Dimensional Array(1-D)
2. Multi-Dimensional Array (2-D OR 3-D)

1. Single Dimensional Array(1-D):

Arrays with one set of square bracket '[]' are called single dimensional array .A single-dimensional array or 1-D array is a linear list consisting of related and similar data items.

Declaration of Single Dimensional Array:

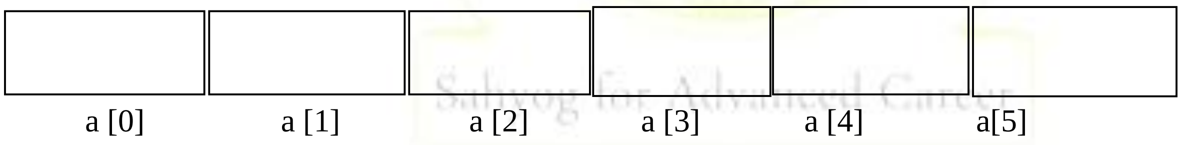
Syntax:

Data_type array_Name [array size] ;

Example:

int a[6];

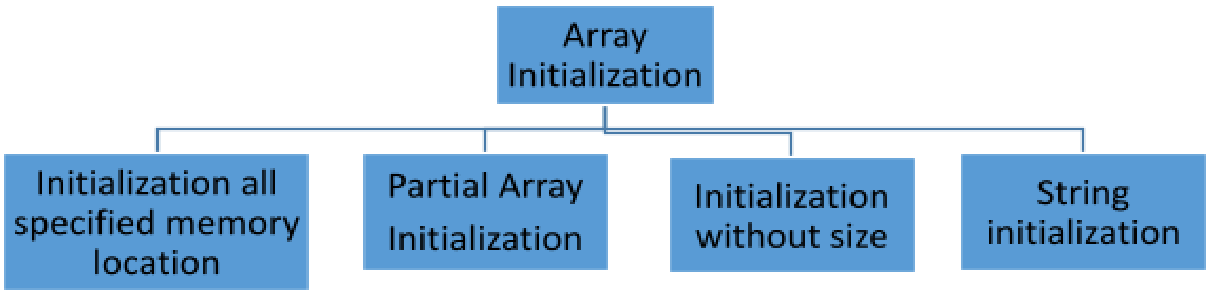
Here, the compiler will reserve 6 locations (4*6=24 bytes) for the array a. In each of the location we can store an integer value. The memory allocated by the compiler is pictorially represented as:



Initialization of Single-dimensional Array:

Assigning the required data to a variable before processing is called initialization.

Different ways of initializing arrays:


Array Initialization
Initialization all specified memory location
Partial Array Initialization
Initialization without size
String initialization

1. **Initialization all specified memory location:**

Array can be initialized at the time of declaration when their initial values are known in advance.

Example:

```
int m[5]={12,32,43,54,65};
```

During Compilation, 5 contiguous memory locations are reserved by the compiler for the array ‘m’ and all the locations are initialized as shown below:

12	32	43	54	65
m[0]	m[1]	m[2]	m[3]	m[4]

2. **<u>Partial array initialization:</u>**

If the number of values to be initialized is less than the size of the array, then the elements are initialized in the order from 0th location. The remaining location will be initialized to zero automatically.

Example:

int m[5]={14,23};

Here Compiler will allocate 5 memory locations and initializes first two locations with 14 and 23.The next set of memory locations will automatically initialize to zero by the compiler.

14	23	0	0	0
m[0]	m[1]	m[2]	m[3]	m[4]

3. **Initialization without size:**

While creating an array size is not mandatory but if you are creating an array without specifying the array size then initialization is mandatory.

Example:

int p[ ]={32,54,53,78};

Here,We have not mentioned the array size, so array size will be the toal number of elements specified.so the array size will be set to 4 automatically.

32	54	53	78
P[0]	p[1]	p[2]	p[3]

4. **Array initializing of a string:**

In array string ends with a NULL character (‘\0’) and this NULL character also takes one byte in the memory, so in case of array of string size will be total character + 1 (for NULL).

Example:

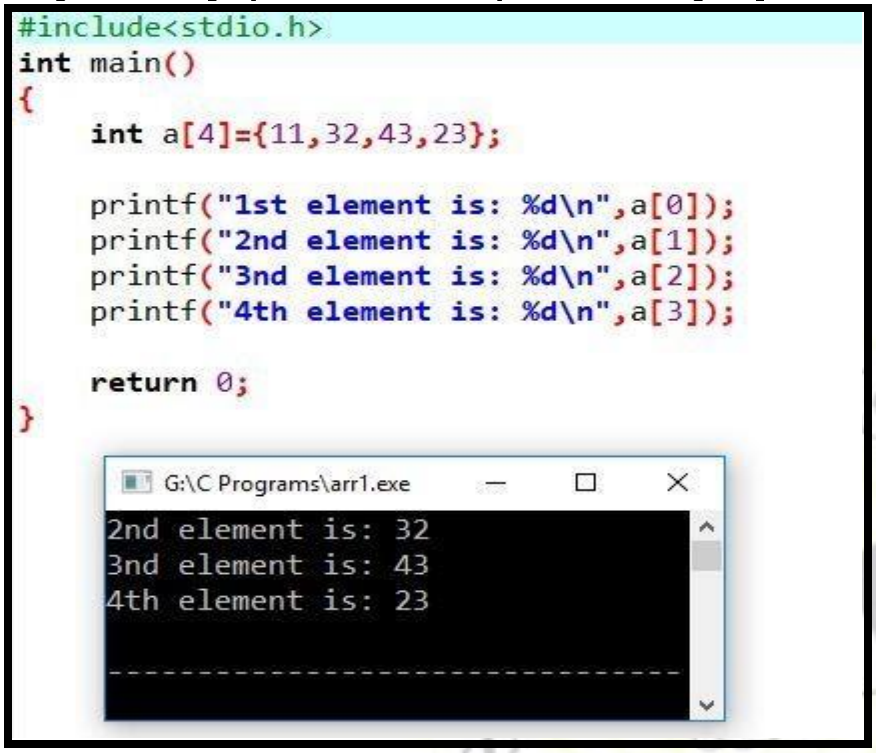
```
char c[7]={‘s’,’a’,’h’,’y’,’o’,’g’,’\0’};
                    OR
 char c[ ]= ”Sahyog”;
```

Reading & Writing Single Dimensional Array in C Language:

Reading single dimensional array:

- Using index no. we can access the element of the array.
- Eg. int a[]={10,11,12,13,14};
- Here, index no of 10 is 0,index of 11 is 1,index of 12 is 2 ,index of 13 is 3,index of 14 is 4

Program to display elements of array without using loop:

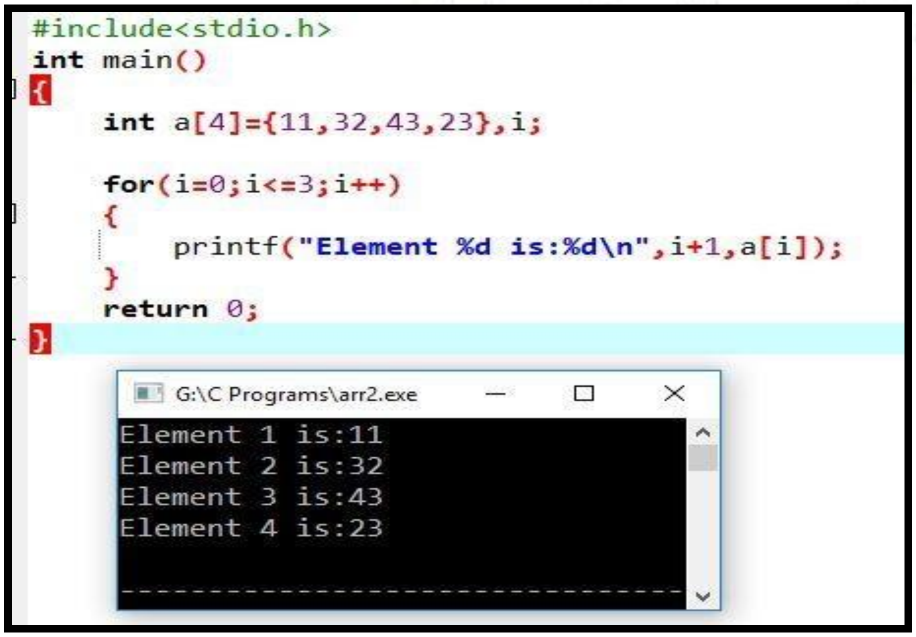


```
#include<stdio.h>
int main()
{
    int a[4]={11,32,43,23};

    printf("1st element is: %d\n",a[0]);
    printf("2nd element is: %d\n",a[1]);
    printf("3nd element is: %d\n",a[2]);
    printf("4th element is: %d\n",a[3]);

    return 0;
}
```

G:\C Programs\arr1.exe

```
2nd element is: 32
3nd element is: 43
4th element is: 23

--------------------------------
```

Program to display elements of an array using any loop.

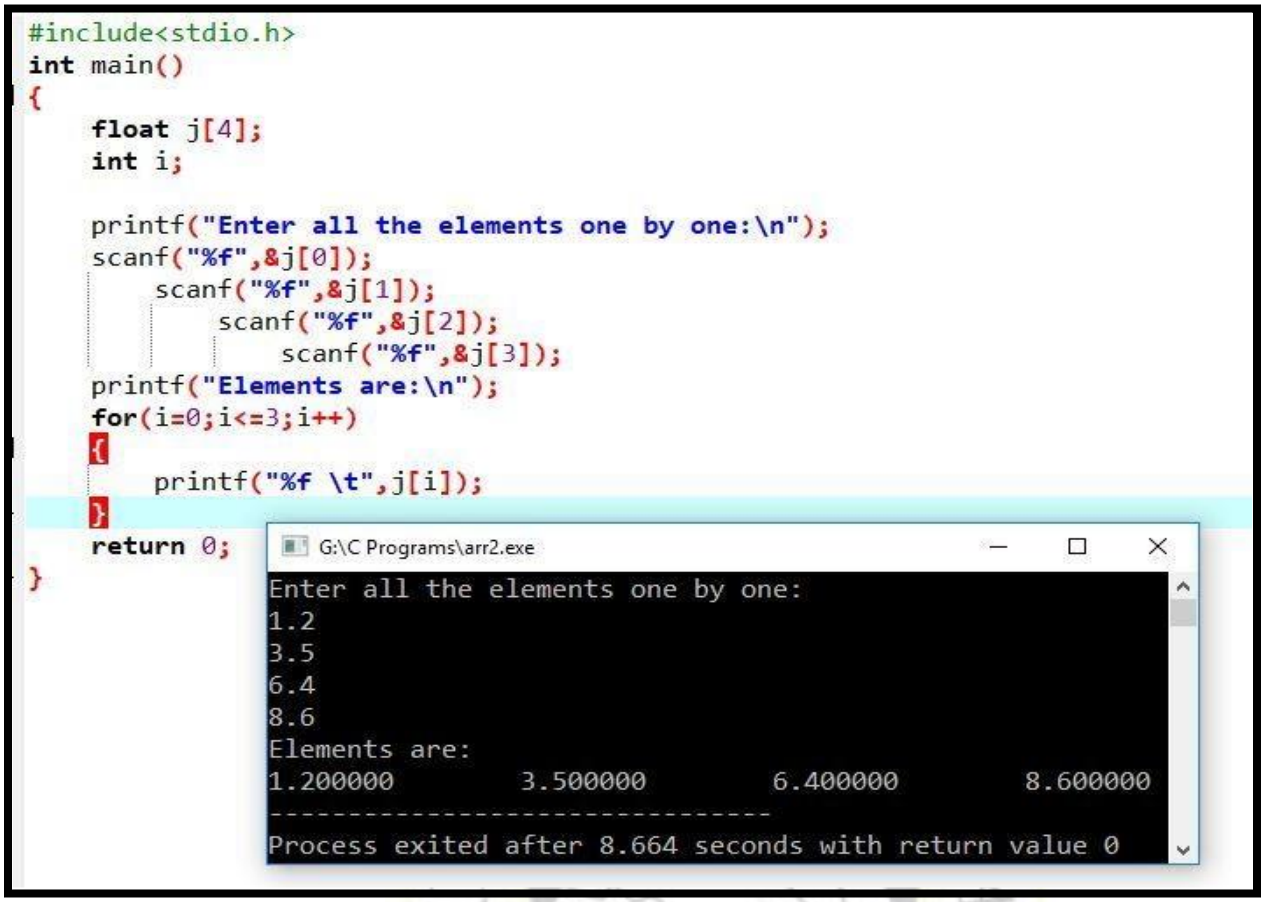


```
#include<stdio.h>
int main()
{
    int a[4]={11,32,43,23},i;

    for(i=0;i<=3;i++)
    {
        printf("Element %d is:%d\n",i+1,a[i]);
    }
    return 0;
}
```

G:\C Programs\arr2.exe

```
Element 1 is:11
Element 2 is:32
Element 3 is:43
Element 4 is:23

--------------------------------
```

WAP to take input from the user in an array without using loop.

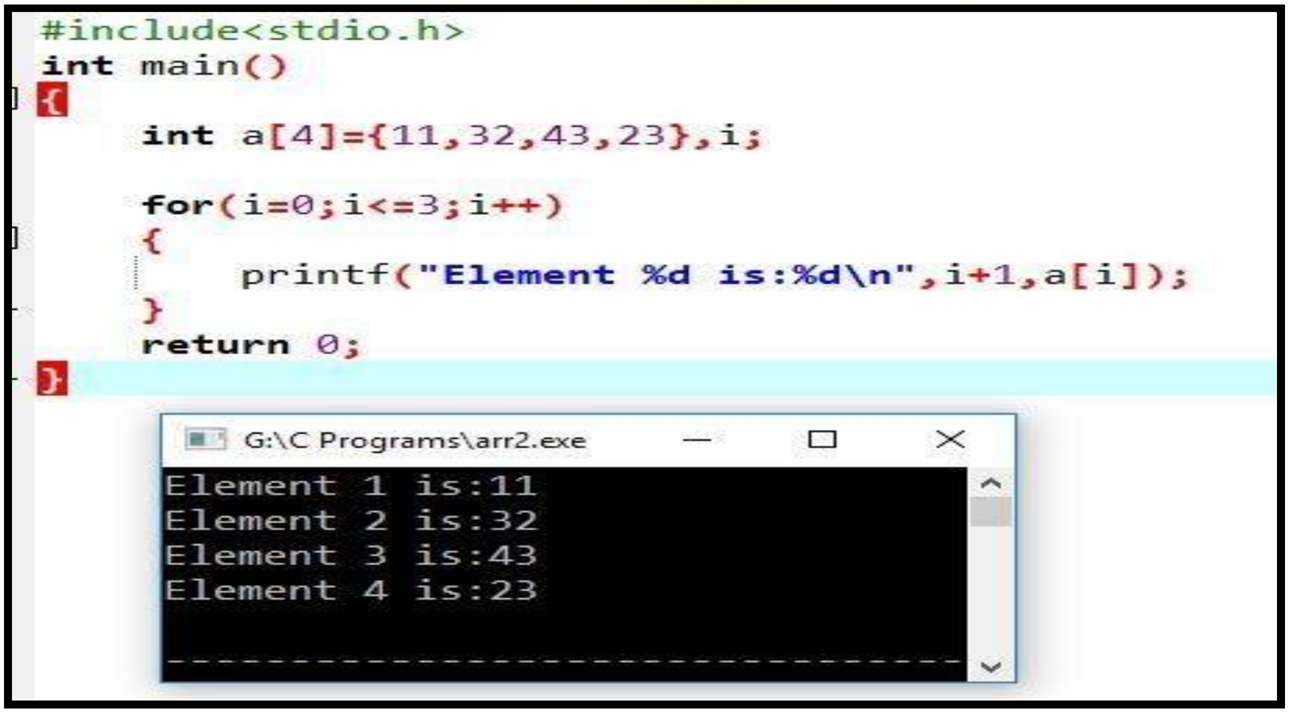


```
#include<stdio.h>
int main()
{
    float j[4];
    int i;

    printf("Enter all the elements one by one:\n");
    scanf("%f",&j[0]);
        scanf("%f",&j[1]);
            scanf("%f",&j[2]);
                scanf("%f",&j[3]);
    printf("Elements are:\n");
    for(i=0;i<=3;i++)
    {
        printf("%f \t",j[i]);
    }
    return 0;
}
```

G:\C Programs\arr2.exe

```
Enter all the elements one by one:
1.2
3.5
6.4
8.6
Elements are:
1.200000        3.500000        6.400000        8.600000
--------------------------------
Process exited after 8.664 seconds with return value 0
```

WAP in C to display elements of an array using any loop.



```
#include<stdio.h>
int main()
{
    int a[4]={11,32,43,23},i;

    for(i=0;i<=3;i++)
    {
        printf("Element %d is:%d\n",i+1,a[i]);
    }
    return 0;
}
```

G:\C Programs\arr2.exe

```
Element 1 is:11
Element 2 is:32
Element 3 is:43
Element 4 is:23

--------------------------------
```

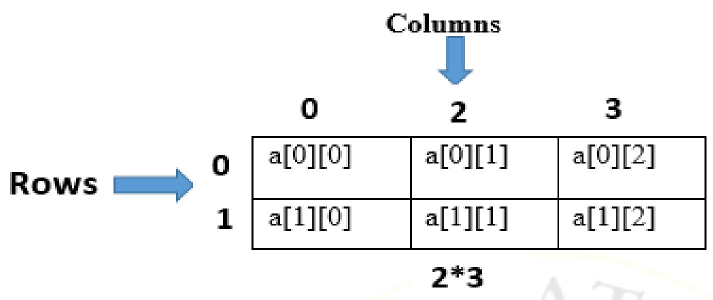
2. Multi-Dimensional Array (2-D OR 3-D):

An array with 2 set of square brackets '[] []' are called 2-dimensional array. Array with 2 or more dimensions are called multi-dimensional array.

Declaration of Multi-Dimensional Array:

Syntax: Data_type array_name[size1][size2];

Example: int a[2][3];



Here, array 'a' is 2-dimensional array with 2 rows and 3 columns. This declaration informs the compiler to reserve 6 locations (2*3=6) contiguously one after the other. Total memory reserved is 6*4 (int takes 4 byte)=24 bytes.

Initialization of 2-D Array:

Assigning required value to a variable before processing is called initialization.

Syntax:

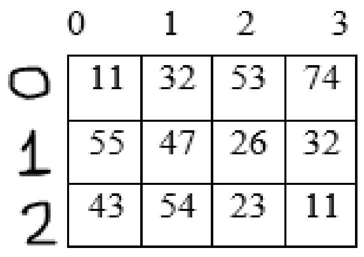
data_type variable_name[size1][size2]={{val1,val2},{val3,val4}};

i. Initialization all specified memory location:

Example:

```
int first[3][4]={ {11,32,53,74},
                  {55,47,26,32},
                  {43,54,23,11}};
```

The declaration indicates that array 'first' has 3 rows and 4 columns. The pictorial representation of this 2-D array is:



	0	1	2	3
0	11	32	53	74
1	55	47	26	32
2	43	54	23	11

ii. Partial array Initialization:

If the number of values to be initialized is less than the size of the array, then the elements are initialized from left to right one after the other and the remaining locations will be initialized to zero automatically.

Example:

```
int first[3][4]={ {11,32 },
                 {55,47 },
                  {43,54}
                };
```

The declaration indicates that array 'first' has 3 rows and 4 columns. The pictorial representation is of this partial array is shown below:

	0	1	2	3
0	11	32	0	0
1	55	47	0	0
2	43	54	0	0

iii. Initialization without size:

Example:

```
int first[][3]={ {11,32,32 },{3,4,6}};
```

In 2-d array size of row is not mandatory but size of column is mandatory.

Reading & Writing Multi Dimensional Array:

1. **WAP to display element of 2-D array without using loop:**

```
 #include<stdio.h>
 int main()
{
int a[2][2]={ {21,42},{53,74} };
printf("first element is: %d\n",a[0][0]);
printf("second element is: %d\n",a[0][1]);
printf("third element is: %d\n",a[1][0]);
printf("fourth element is: %d\n",a[1][1]);
return 0;
}
```

Output:

```
first element is: 21
second element is: 42
third element is: 53
fourth element is: 74
```

2. **WAP to display element of 2-D array using loop:**

```
 #include<stdio.h>
int main()
{
int a[2][2]={{21,42},{53,74}};
for(int i=0;i<=1;i++)
{
for(int j=0;j<=1;j++)
{
  printf("%d\t",a[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:

```
21    42
53    74
```

3. WAP to take input from user without using loop:

```
 #include<stdio.h>
int main()
{
int a[2][2];
printf("Enter 2 Elements of 1st row:\n");
scanf("%d",&a[0][0]);
scanf("%d",&a[0][1]);
printf("Enter 2 Elements of 2nd row:\n");
scanf("%d",&a[1][0]);
scanf("%d",&a[1][1]);
printf("Entered Elements are:\n");
for(int i=0;i<=1;i++)
{
for(int j=0;j<=1;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
return 0;
}
```

Output:

```
Enter 2 Elements of 1st row:
54
43
Enter 2 Elements of 2nd row:
34
22
Entered Elements are:
54   43
34   22
```

4. WAP to take input from user using loop:

```
#include<stdio.h>
using namespace std;
int main()
{
  int a[2][2];
  printf("Enter Elements one by one:\n");
  for(int i=0;i<=1;i++)
  {
   for(int j=0;j<=1;j++)
   {
    scanf("%d",&a[i][j]);
   }
 }
printf("Entered Elements are:\n");
for(int i=0;i<=1;i++)
{
 for(int j=0;j<=1;j++)
 {
 printf("%d\t",a[i][j]);
 }
 printf("\n");
}
return 0;
}
```

Output:

```
Enter Elements one by one:
4
3
2
1
Entered Elements are:
4   3
2   1
```

5. **WAP to perform addition of 2-D Matrix:**

```
#include<stdio.h>
int main()
{
int a[2][3],b[2][3],c[2][3];
printf("Enter Elements of first matrix:\n");
for(int i=0;i<=1;i++) // 2 rows
{
 for(int j=0;j<=2;j++) // 3 columns
 {
 scanf("%d",&a[i][j]);
 }
 printf("\n");
}
printf("Enter elements of 2nd matrix:\n");
for(int i=0;i<=1;i++)
```

```
            {
             for(int j=0;j<=2;j++)
             {
             scanf("%d",&b[i][j]);
             }
             printf("\n");
            }
            printf("Addition of matrix is:\n");
            for(int i=0;i<=1;i++)
            {
             for(int j=0;j<=2;j++)
             {
             c[i][j]=a[i][j]+b[i][j]; //Addition of Matrix A and B
             printf("%d\t",c[i][j]); // Displaying the addition
             }
             printf("\n");
            }
            return 0;
            }
```

Output:

```
Enter Elements of first matrix:
1 2 4
4 5 6
Enter elements of 2nd matrix:
3 2 1
1 4 3
Addition of the matrix is:
4  4  5
5  9  9
```

Additional Programs:

1. program to search for element in an array

```
#include<stdio.h>
int main()
{
            int n[]={11,32,44,34,22},num,len,i;
            printf("Enter the element you are searching for\n");
            scanf("%d",&num);

            len=sizeof(n)/sizeof(int); //calculating length of array

          for(i=0;i<len;i++)
         {
            if(n[i]==num)
            {
                        printf("Element found at %d index",i);
```

```
                    break;
            }
        }
          if(i==len)
          {
                    printf("Element not found");
          }
   return 0;
}
```

Output 1:

```
Enter the element you are searching for
12
Element not found
```

Output 2:

```
Enter the element you are searching for
32
Element found at 1 index
```

2. Program to generate list of even and odd numbers

```
#include<stdio.h>
int main()
{
     int a[6],i;
     printf("Enter 6 Elements \n ");
     for(i=0;i<5;i++)
     {
               scanf("%d",&a[i]);
     }
     printf("**List of even no.**\n");
     for(i=0;i<5;i++)
     {
               if(a[i]%2==0)
               {
                    printf("%d\n",a[i]);
               }
     }
     printf("**List of odd no.**\n");
     for(i=0;i<5;i++)
     {
               if(a[i]%2!=0)
               {
                    printf("%d\n",a[i]);
               }
     }
   return 0;
}
```

Output:

```
Enter 6 Elements
11
12
13
17
18

**List of even no.**
12
18
**List of odd no.**
11
13
17
```

3. Program to search for smallest no from an array.

```
#include<stdio.h>
int main()
{
        int a[6]={11,34,54,10,3,7},min,i;
        min=a[0];
        for(i=0;i<6;i++)
        {
                if(min>a[i])
                {
                   min=a[i];
                }
        }
        printf("Smallest no is %d",min);
        return 0;
}
```

Output:

```
Smallest no is 3
```

4. Program to calculate sum of element of 2-D matrix

```
#include<stdio.h>
int main()
{
        int m[2][2]={{2,3},{4,5}},sum=0;
        int i,j;
        printf("Elements are:\n");
        for(i=0;i<=1;i++)
        {
                for(j=0;j<=1;j++)
                {
                        printf("%d ",m[i][j]);
                }
                printf("\n");
        }
        for(i=0;i<=1;i++)
        {
                for(j=0;j<=1;j++)
                {
                        sum=sum+m[i][j];
                }
        }
        printf("\nSum=%d",sum);
  return 0;
}
```

Output:

```
Elements are:
2 3
4 5
Sum=14
```

5. Biggest element in a given matrix

```
#include<stdio.h>
int main()
{
        int m[2][3]={{12,11,10},{14,15,13}},big;
        int i,j;
        big=m[0][0];
        printf("Elements are\n");
        for(i=0;i<=1;i++)
        {
                for(j=0;j<=2;j++)
                {
                        printf("%d ",m[i][j]);
                }
                printf("\n");
        }
        for(i=0;i<=1;i++)
        {
                for(j=0;j<=1;j++)
                {
                  if(big<m[i][j])
                  {
                  big=m[i][j];
                  }
```

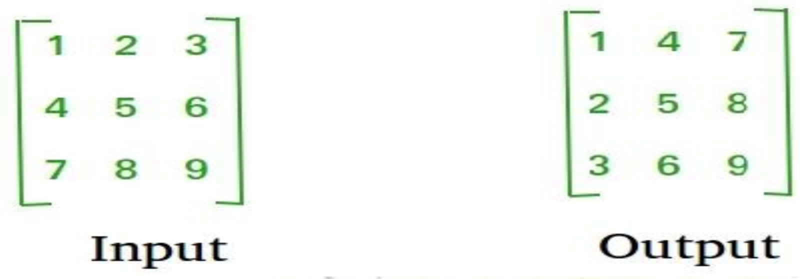
```
            }
        }
        printf("\nBiggest Element is %d",big);
        return 0;
}
```

Output:

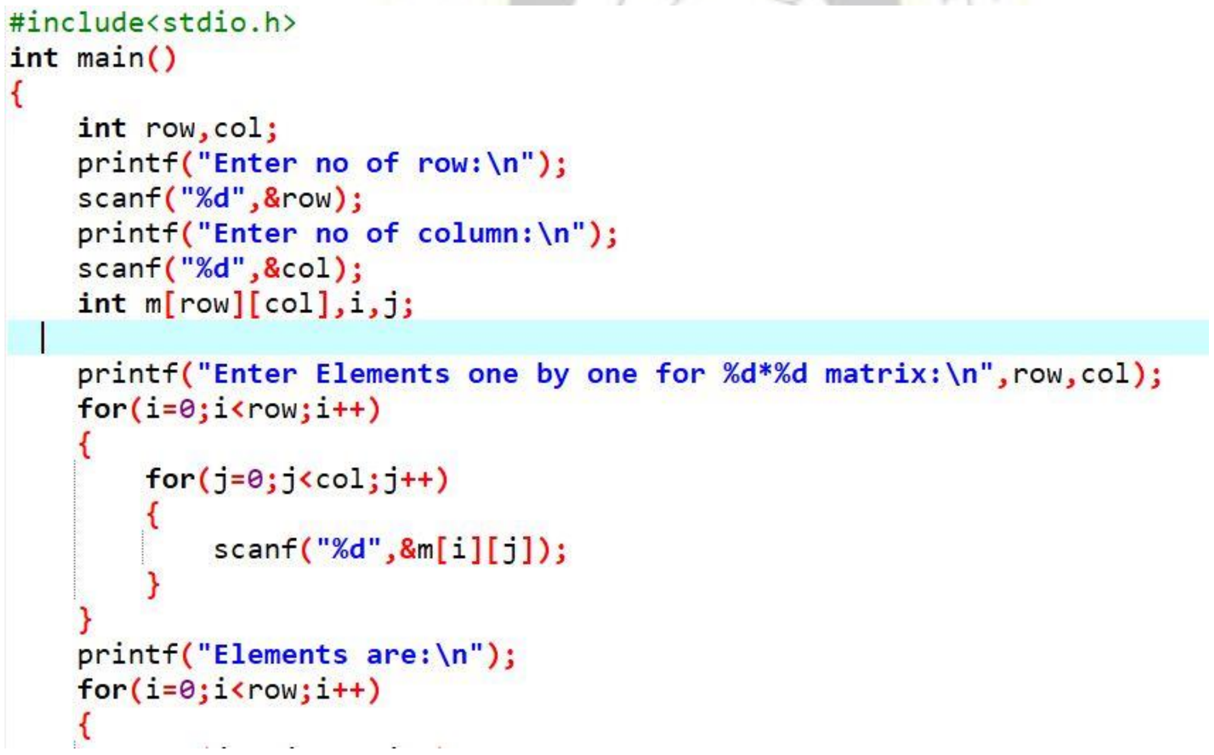
```
Elements are
12 11 10
14 15 13
Biggest Element is: 15
```

Transpose of matrix:

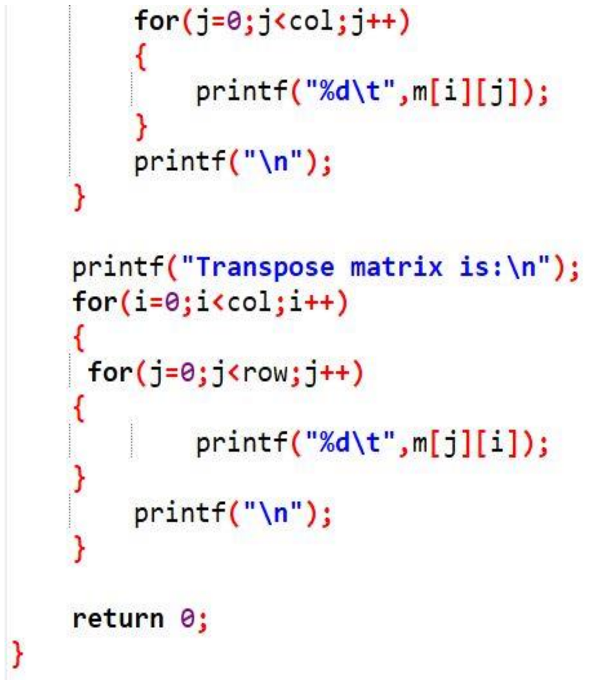
Transpose of a matrix is obtained by changing rows to columns and columns to rows.



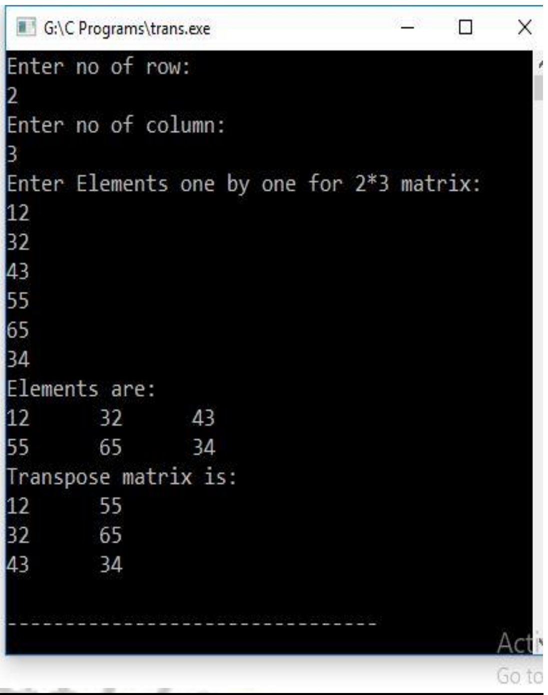
Program to get transpose matrix:



```
#include<stdio.h>
int main()
{
    int row,col;
    printf("Enter no of row:\n");
    scanf("%d",&row);
    printf("Enter no of column:\n");
    scanf("%d",&col);
    int m[row][col],i,j;

    printf("Enter Elements one by one for %d*%d matrix:\n",row,col);
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            scanf("%d",&m[i][j]);
        }
    }
    printf("Elements are:\n");
    for(i=0;i<row;i++)
    {
```



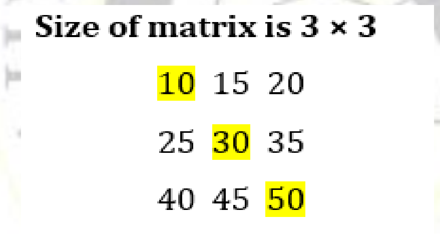
```c
        for(j=0;j<col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }

    printf("Transpose matrix is:\n");
    for(i=0;i<col;i++)
    {
     for(j=0;j<row;j++)
    {
            printf("%d\t",m[j][i]);
    }
        printf("\n");
    }

    return 0;
}
```



```
Enter no of row:
2
Enter no of column:
3
Enter Elements one by one for 2*3 matrix:
12
32
43
55
65
34
Elements are:
12      32      43
55      65      34
Transpose matrix is:
12      55
32      65
43      34

------------------------------
```

Trace of matrix:

The trace of a matrix is the sum of the elements of diagonal in a given matrix. For example, consider the matrix shown below:



Size of matrix is 3 × 3

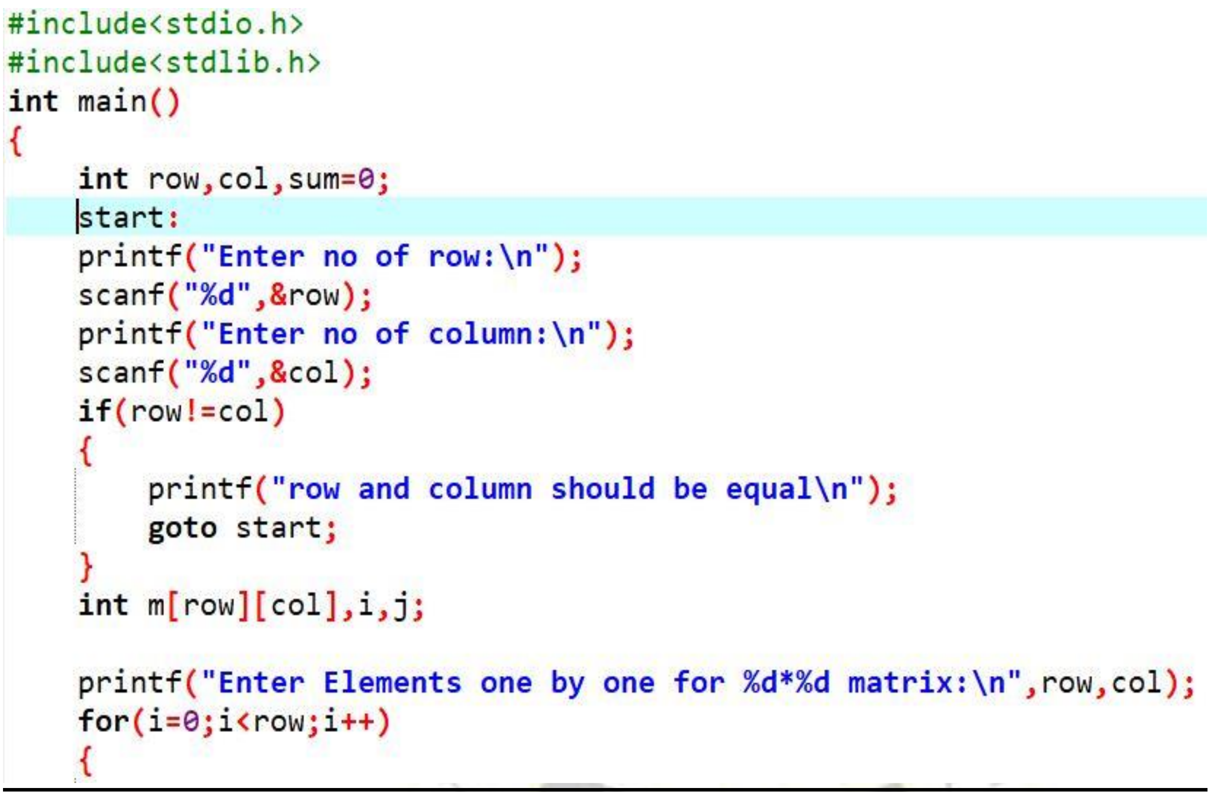
```
10 15 20
25 30 35
40 45 50
```

Trace=10+30+50=90

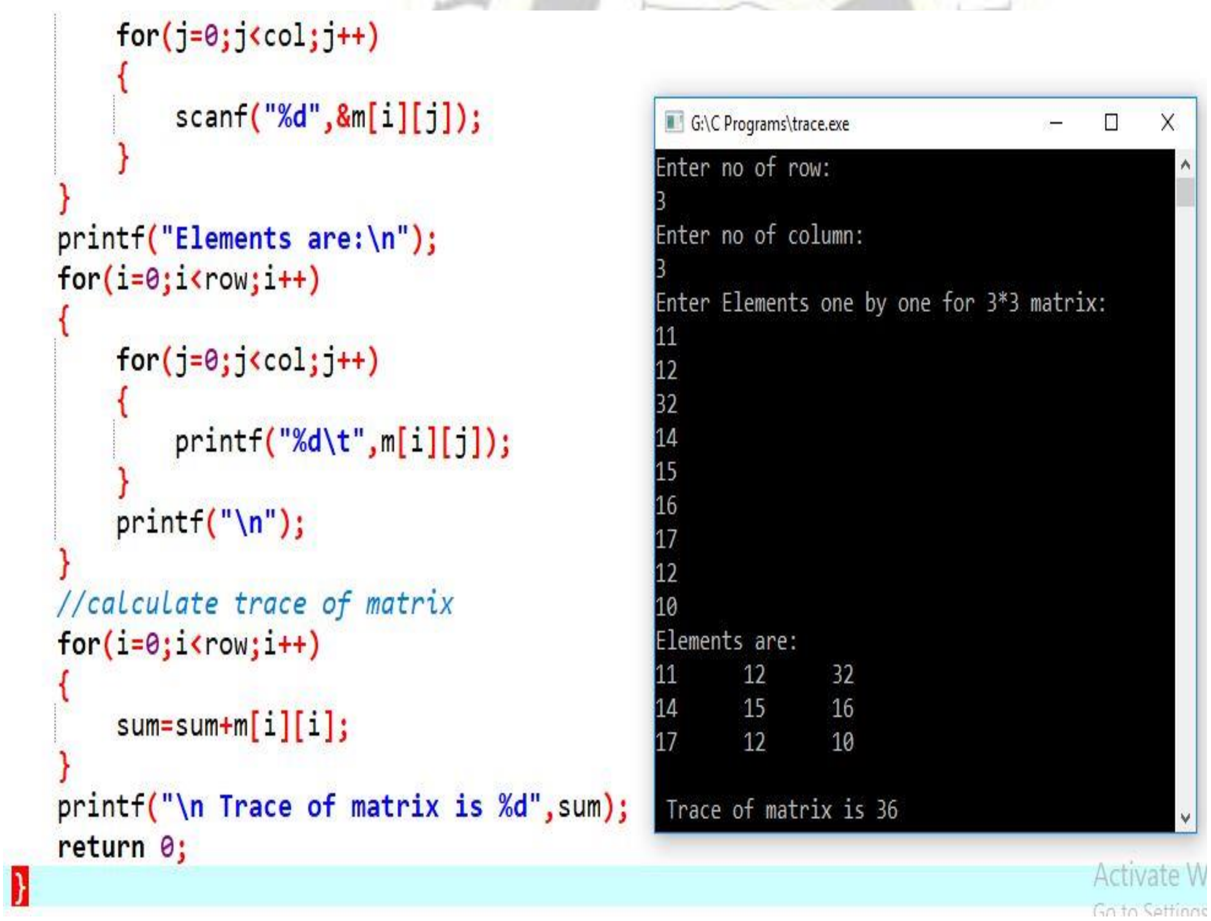
If we add the diagonal elements 10, 30 and 50, we get 90.So, the trace of given matrix is 90.

Note: Since diagonal elements exist only for a square matrix, the size of the matrix should be n x n i.e. size of row and column should be equal. For a rectangular matrix trace cannot be computed.

Program to computer trace of matrix:



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int row,col,sum=0;
    start:
    printf("Enter no of row:\n");
    scanf("%d",&row);
    printf("Enter no of column:\n");
    scanf("%d",&col);
    if(row!=col)
    {
        printf("row and column should be equal\n");
        goto start;
    }
    int m[row][col],i,j;

    printf("Enter Elements one by one for %d*%d matrix:\n",row,col);
    for(i=0;i<row;i++)
    {
```



```c
        for(j=0;j<col;j++)
        {
            scanf("%d",&m[i][j]);
        }
    }
    printf("Elements are:\n");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
    //calculate trace of matrix
    for(i=0;i<row;i++)
    {
        sum=sum+m[i][i];
    }
    printf("\n Trace of matrix is %d",sum);
    return 0;
}
```

G:\C Programs\trace.exe

```
Enter no of row:
3
Enter no of column:
3
Enter Elements one by one for 3*3 matrix:
11
12
32
14
15
16
17
12
10
Elements are:
11      12      32
14      15      16
17      12      10

 Trace of matrix is 36
```

Sparse Array:

- A **sparse array** is an **array** in which majority of the array elements are zero.
- If half of the array element or more than half element is zero then it would be a sparse array.

Dense Array:

- A **Dense array** is an **array** in which majority of the array elements are non-zero.

1	**0**	**3**
0	**0**	**4**
0	**6**	**0**

It's a sparse array

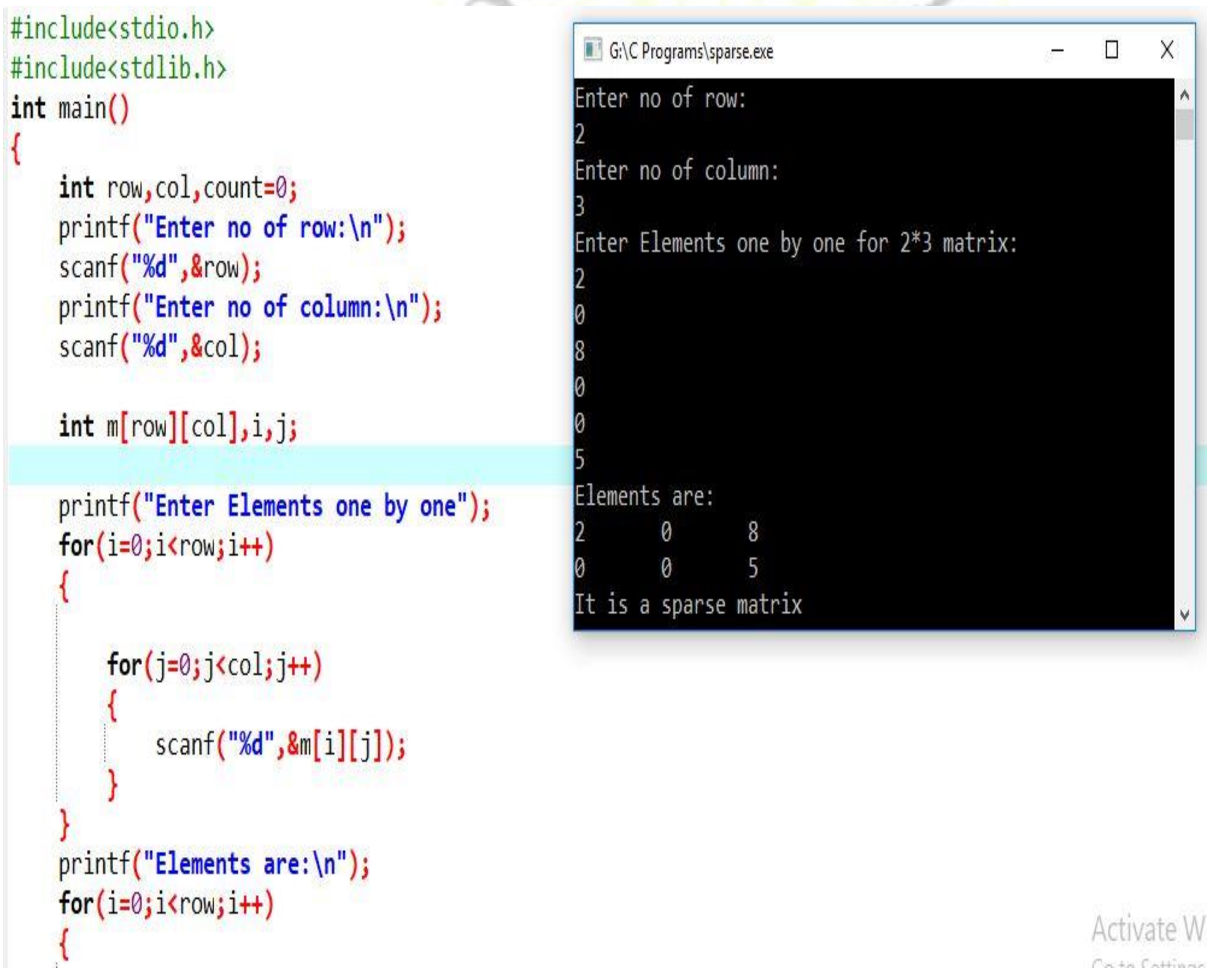
1	**2**	**0**
4	**11**	**5**
0	**15**	**0**

It's a Dense array

0	**2**	**0**
3	**0**	**5**
0	**0**	**1**

It's a sparse array

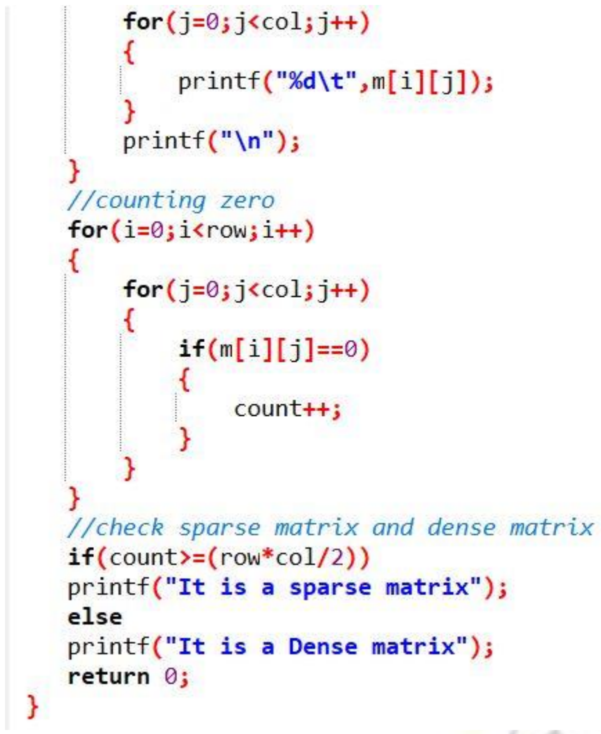
Program to check whether given array is sparse array or not.



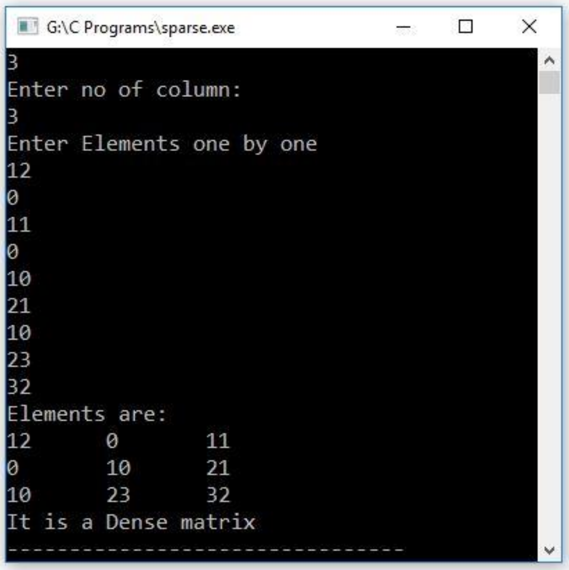
```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int row,col,count=0;
    printf("Enter no of row:\n");
    scanf("%d",&row);
    printf("Enter no of column:\n");
    scanf("%d",&col);

    int m[row][col],i,j;

    printf("Enter Elements one by one");
    for(i=0;i<row;i++)
    {

        for(j=0;j<col;j++)
        {
            scanf("%d",&m[i][j]);
        }
    }
    printf("Elements are:\n");
    for(i=0;i<row;i++)
    {
```

G:\C Programs\sparse.exe

```
Enter no of row:
2
Enter no of column:
3
Enter Elements one by one for 2*3 matrix:
2
0
8
0
0
5
Elements are:
2       0       8
0       0       5
It is a sparse matrix
```



```
        for(j=0;j<col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
    //counting zero
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            if(m[i][j]==0)
            {
                count++;
            }
        }
    }
    //check sparse matrix and dense matrix
    if(count>=(row*col/2))
    printf("It is a sparse matrix");
    else
    printf("It is a Dense matrix");
    return 0;
}
```



G:\C Programs\sparse.exe

```
3
Enter no of column:
3
Enter Elements one by one
12
0
11
0
10
21
10
23
32
Elements are:
12      0       11
0       10      21
10      23      32
It is a Dense matrix
--------------------------------
```

Triangular Matrix:

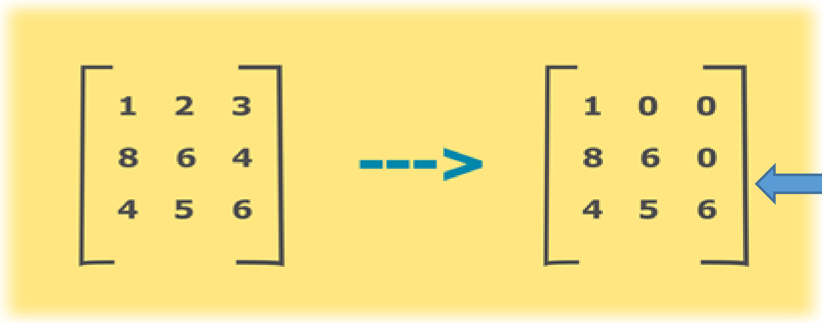
1. Lower Triangular Matrix:

Lower triangular matrix is a square matrix in which all the elements above the principle diagonal will be zero. To find the lower triangular matrix, a matrix needs to be a square matrix that is, number of rows and columns in the matrix needs to be equal.

	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

Note:

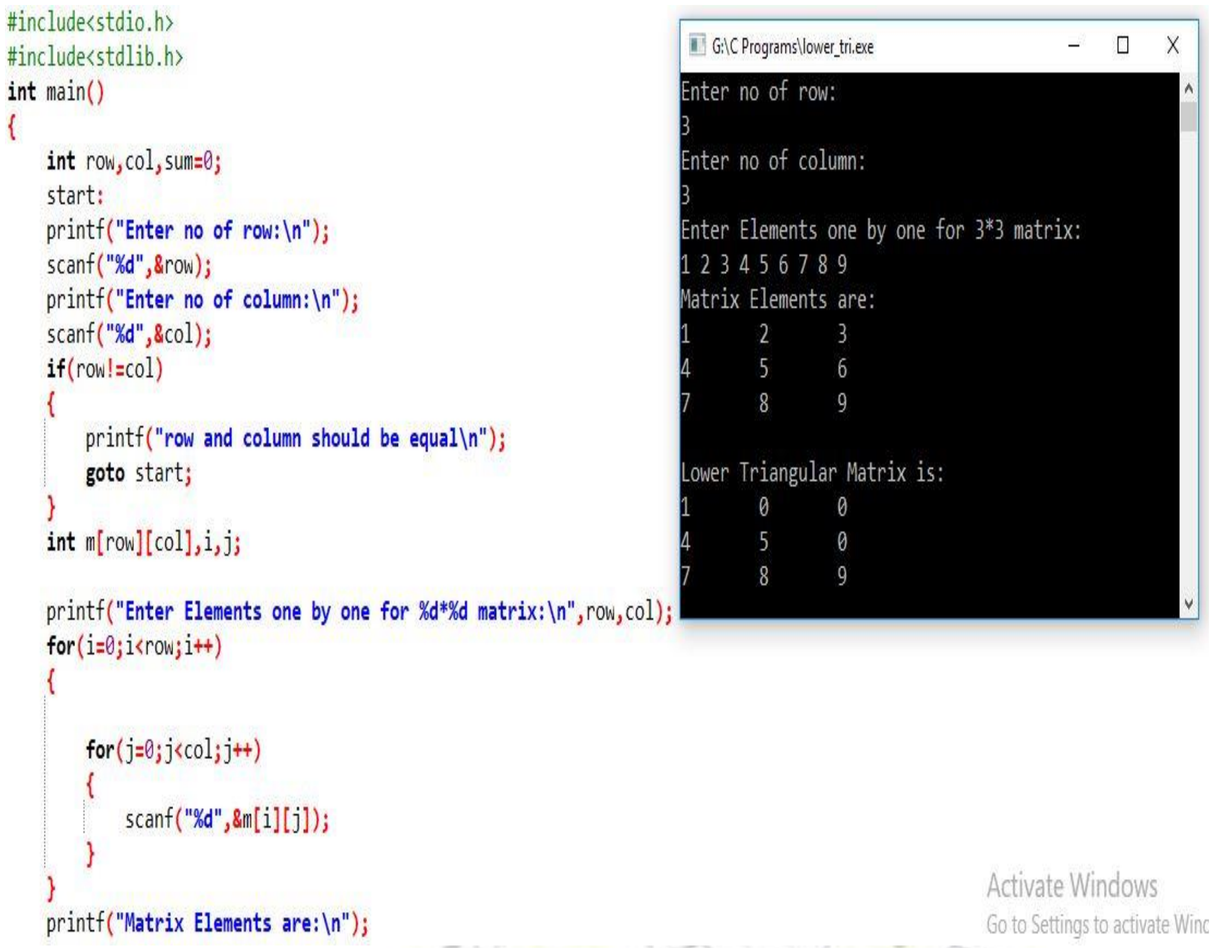
- In Lower triangular matrix value of row is lesser than the value of column.
- So to display Lower triangular matrix just check whether the row is less than column or not…if it is lower ,then print zero else print matrix element.



$$\begin{bmatrix} 1 & 2 & 3 \\ 8 & 6 & 4 \\ 4 & 5 & 6 \end{bmatrix} \dashrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 8 & 6 & 0 \\ 4 & 5 & 6 \end{bmatrix} \longleftarrow$$

Lower Triangular Matrix

Program to display a lower triangular matrix:

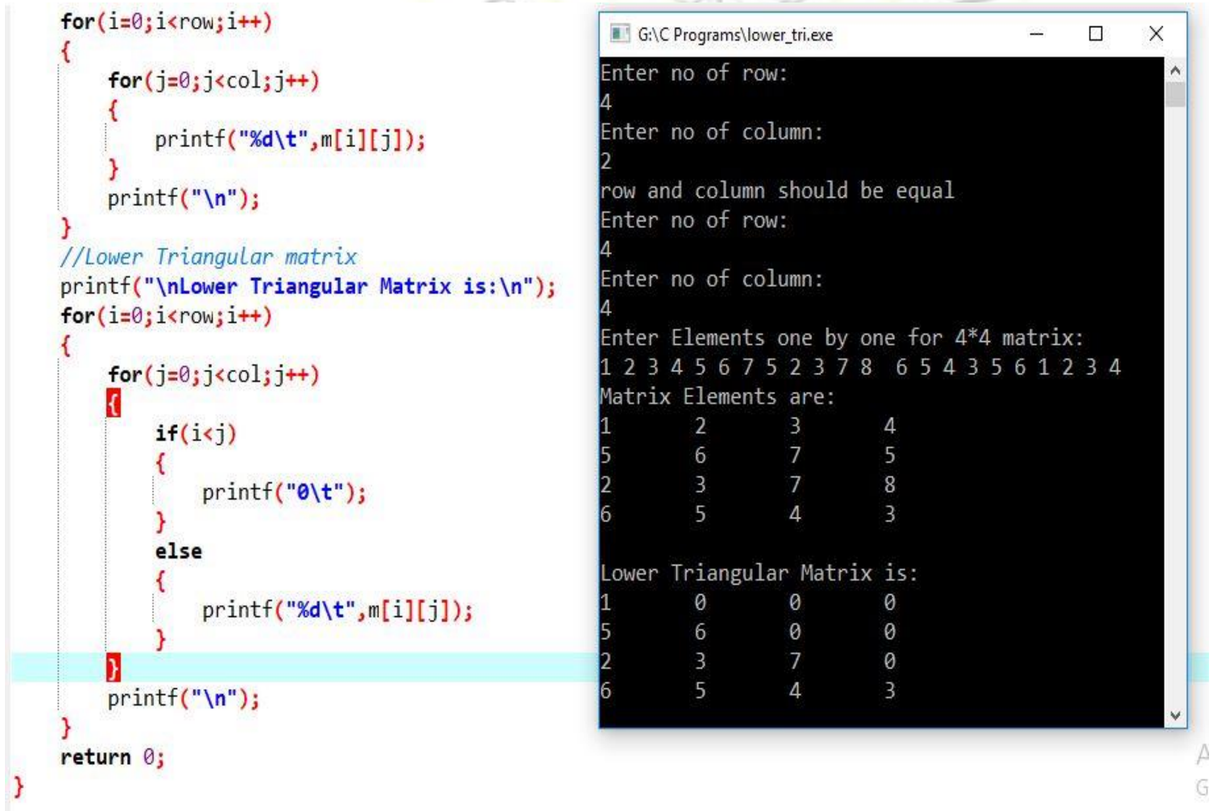


```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int row,col,sum=0;
    start:
    printf("Enter no of row:\n");
    scanf("%d",&row);
    printf("Enter no of column:\n");
    scanf("%d",&col);
    if(row!=col)
    {
        printf("row and column should be equal\n");
        goto start;
    }
    int m[row][col],i,j;

    printf("Enter Elements one by one for %d*%d matrix:\n",row,col);
    for(i=0;i<row;i++)
    {

        for(j=0;j<col;j++)
        {
            scanf("%d",&m[i][j]);
        }
    }
    printf("Matrix Elements are:\n");
```

G:\C Programs\lower_tri.exe

```
Enter no of row:
3
Enter no of column:
3
Enter Elements one by one for 3*3 matrix:
1 2 3 4 5 6 7 8 9
Matrix Elements are:
1       2       3
4       5       6
7       8       9

Lower Triangular Matrix is:
1       0       0
4       5       0
7       8       9
```

Activate Windows
Go to Settings to activate Winc



```c
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
    //Lower Triangular matrix
    printf("\nLower Triangular Matrix is:\n");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            if(i<j)
            {
                printf("0\t");
            }
            else
            {
                printf("%d\t",m[i][j]);
            }
        }
        printf("\n");
    }
    return 0;
}
```

G:\C Programs\lower_tri.exe

```
Enter no of row:
4
Enter no of column:
2
row and column should be equal
Enter no of row:
4
Enter no of column:
4
Enter Elements one by one for 4*4 matrix:
1 2 3 4 5 6 7 5 2 3 7 8  6 5 4 3 5 6 1 2 3 4
Matrix Elements are:
1       2       3       4
5       6       7       5
2       3       7       8
6       5       4       3

Lower Triangular Matrix is:
1       0       0       0
5       6       0       0
2       3       7       0
6       5       4       3
```

Upper Triangular Matrix:
Upper triangular matrix is a square matrix in which all the elements below the principle diagonal are zero. To find the upper triangular matrix, a matrix needs to be a square matrix that is, the number of rows and columns in the matrix needs to be equal.

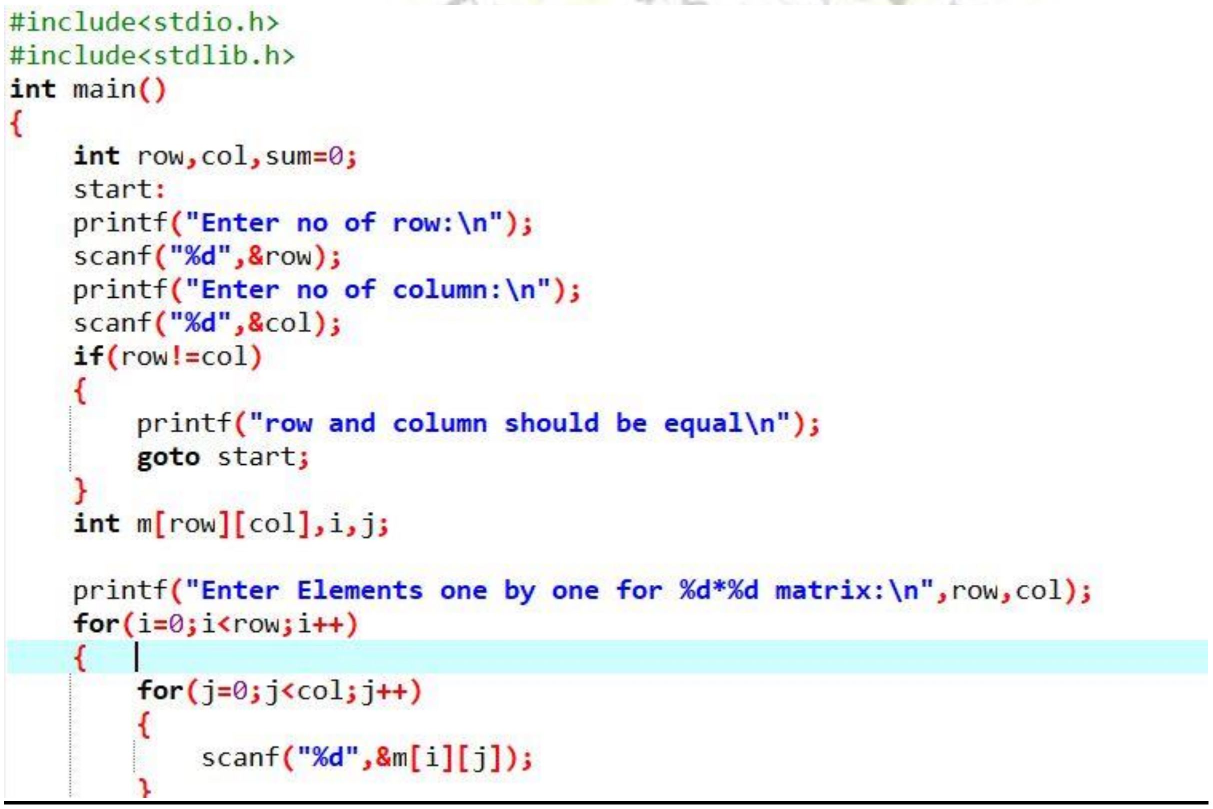
	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2

Note:

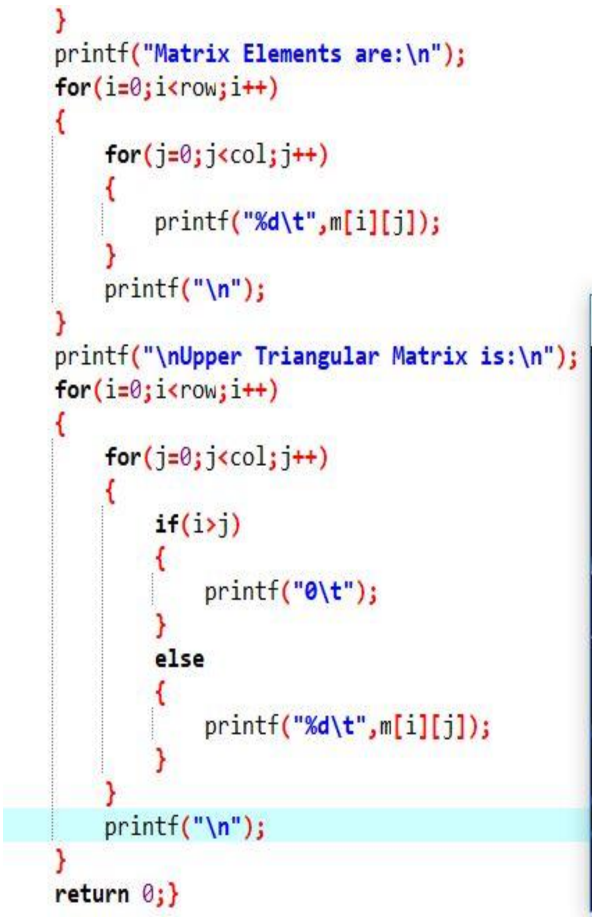
- In upper triangular matrix value of row is greater than value of column always.
- So to display upper triangular matrix just check whether the row is greater than column or not…if it is greater then print zero else print matrix element.

$$\begin{bmatrix}1&2&3\\8&6&4\\4&5&6\end{bmatrix} \dashrightarrow \begin{bmatrix}1&2&3\\0&6&4\\0&0&6\end{bmatrix} \longleftarrow \text{Upper Triangular Matrix}$$

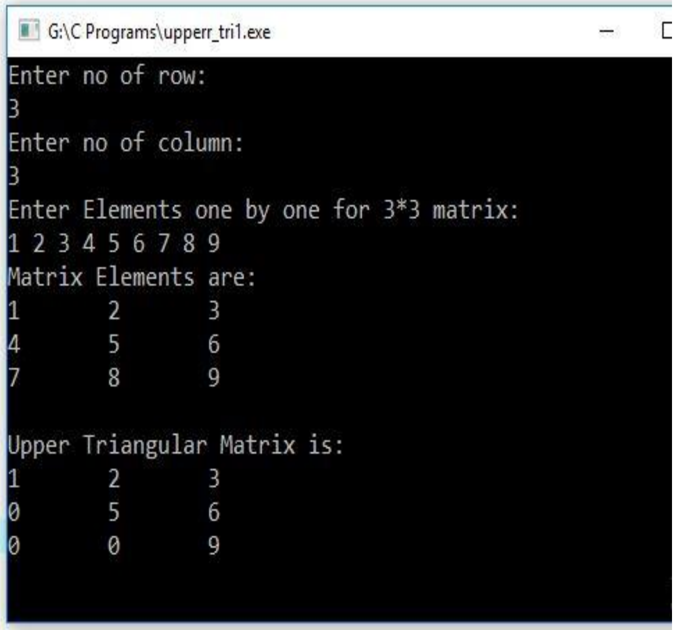
Program to display upper Triangular matrix:



```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int row,col,sum=0;
    start:
    printf("Enter no of row:\n");
    scanf("%d",&row);
    printf("Enter no of column:\n");
    scanf("%d",&col);
    if(row!=col)
    {
        printf("row and column should be equal\n");
        goto start;
    }
    int m[row][col],i,j;

    printf("Enter Elements one by one for %d*%d matrix:\n",row,col);
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            scanf("%d",&m[i][j]);
        }
```



```c
    }
    printf("Matrix Elements are:\n");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            printf("%d\t",m[i][j]);
        }
        printf("\n");
    }
    printf("\nUpper Triangular Matrix is:\n");
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            if(i>j)
            {
                printf("0\t");
            }
            else
            {
                printf("%d\t",m[i][j]);
            }
        }
        printf("\n");
    }
    return 0;}
```



G:\C Programs\upperr_tri1.exe

```
Enter no of row:
3
Enter no of column:
3
Enter Elements one by one for 3*3 matrix:
1 2 3 4 5 6 7 8 9
Matrix Elements are:
1       2       3
4       5       6
7       8       9

Upper Triangular Matrix is:
1       2       3
0       5       6
0       0       9
```