Arrays: Declaration of arrays, accessing the elements of an array, storing values in arrays, Operations on arrays, Passing arrays to functions, two dimensional arrays, operations on two-dimensional arrays, twodimensional arrays to functions, multidimensional arrays, applications of arrays.

Introduction:

- An array is a collection of items of same data type stored at contiguous memory locations.
- Array of character is a string.
- Each data item of an array is called an element.
- And each element is unique and located in separated memory location.
- Each of elements of an array share a variable but each element having different index no. known as subscript.

Any element in an array can be accessed using

1. **Name of the array**
2. **Position of the element in an array.**

There are 2 types of array

1. **Single dimensional array**
2. **Multi-dimensional array**

Declaration of 1-Dimensional arrays

- Arrays are declared using following syntax:

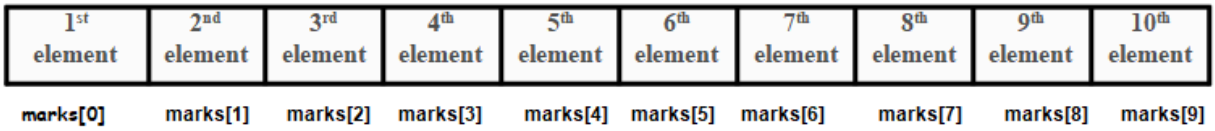
  data_type array_name[size];

  where, type can be int, float or char.

  name is the name of the array.

  size indicates number of elements in the array.

Example: int marks[10];



Calculating the address of Array

- Address of data element, **A[k] = BA(A) + w(k – lower_bound)**

Here,

A is the **array**

k is the **index of the element of which we have to calculate the address**

BA is the **base address of the array A.**

w is the **word size of one element in memory, for example, size of int is 2.**

Example 1:

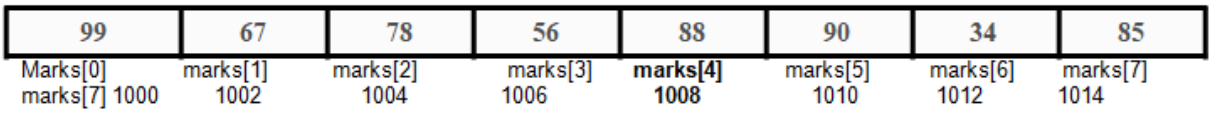
Given an array int marks[ ]={99,67,78,56,88,90,34,85}. Calculate the address of marks[4] if base address=1000.

Address(Marks[4])=1000+2(4-0)// size of int=2

=1000+2*4

=1000+8

=1008



Example 2:

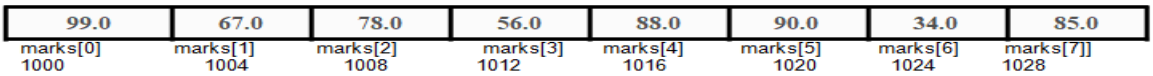
Given an array float avg[ ]={99.0,67.0,78.0,56.0,88.0,90.0,34.0,85.0}. Calculate the address of avg[4] if base address=1000.

Address(Avg[4]) =1000+4(4-0)// size of float=4

=1000+4*4

=1000+16

=1016



Calculating the length of Array

Length of the array is given by:

Length= upper_bound - lower_bound+1

where

Upper_bound=index of the last element

Lower_bound=index of the first element

Usually Lower_bound is zero but this is not a compulsion.

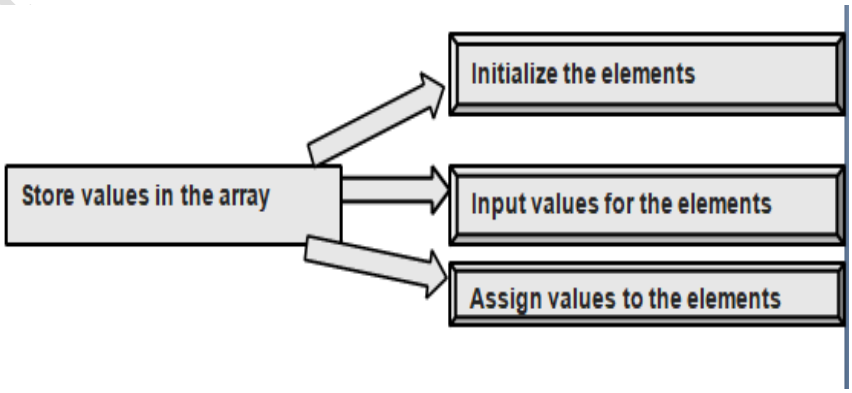
Example 1:

Let Age[5] be Age[0]=2, Age[1]=5, Age[2]=3, Age[3]=1, Age[4]=7.

Length=Upper_bound-Lower_bound+1

=4-0+1=5

Storing values in an Array

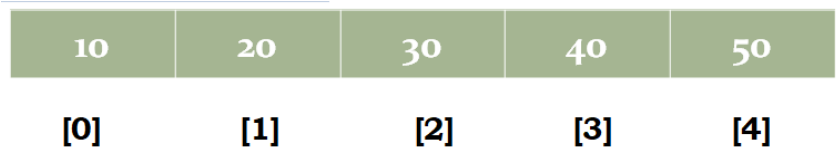


Initialization can be done using the following syntax:

type array_name[size]={list of values};

1. Initializing all specified memory location:

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

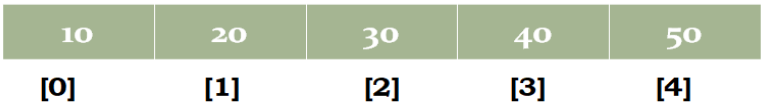


2. **Partial array initialization**

int a[5]={10,20}

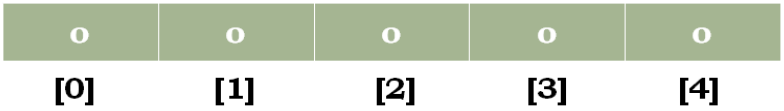
3. **Array initialization without size:**

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



4. **Array initialization without elements**

int a[5]={0}



Inputting values from keyboard

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

Assigning values to Individual Elements

```
int i, arr1[10], arr2[10];
arr1[i]={0,1,2,3,4,5,6,7,8,9};
for(i=0;i<10;i++)
        arr2[i] = arr1[i];
```

WAP to read and write one dimensional array.

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

Output:

```
Enter the elements: 1 2 3 4 5
Array a[0]=1
Array a[1]=2
```

Array a[2]=3

Array a[3]=4

Array a[4]=5

WAP to search an element in an array.

```
#include<stdio.h>
void main()
{
        int i,a[20],n,key;
        printf("Enter the number of elements: ");
        scanf("%d",&n);
        printf("Enter the elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("Enter the key element to be searched: ");
        scanf("%d",&key);
        for(i=0;i<n;i++)
        {
                if(key==a[i])
                printf("Element found at position %d\n",i+1);
        }
}
```

Output:

```
Enter the number of elements: 5
Enter the elements: 1 2 3 4 5
Enter the key element to be searched: 5
Element found at position 5
```

WAP to print the position of smallest of numbers using array.

```
#include<stdio.h>
void main()
{
    int i,a[20],n,small,pos;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    small=a[0];
    pos=0;
    for(i=0;i<n;i++)
    {
      if(a[i]<small)
        {
            small=a[i];
            pos=i;
```

```
        }
    }
    printf("The smallest element is %d and the position is %d",small,pos+1);
}
```

Output:

Enter the number of elements: 5

Enter the elements: 2 3 4 5 6

The smallest element is 2 and the position is 1

Operations on array

1. **Traversing an array**
2. **Inserting an element in an array**
3. **Deleting an element from an array**
4. **Merging 2 arrays**
5. **Searching an element in an array**
6. **Sorting an array in ascending or descending order**

1. Traversing an array

- Traversing an array means accessing each and every element of the array for a specific purpose.

Algorithm for Array Traversal:

Step 1: [Initialization] SET I=lower_bound
Step 2: Repeat steps 3 and 4 while I<=upper_bound
Step 3: Apply process to A[I]
Step 4: Set I=I+1;
Step 5: Exit

2. Inserting an element in an array

- Inserting an element in an array means adding a new data element to an already existing array.

Algorithm to insert a new element to the end of the array:

Step 1: Set upper_bound= upper_bound+1//Increment the value of Upper bound
Step 2: Set A[upper_bound]= VAL(Value that has to be inserted]//New value is stored
Step 3: EXIT

Algorithm to insert a new element in middle of the array:

Step 1: Set I=N//N=Number of elements
Step 2: Repeat 3 and 4 while I>=POS//POS=position at which element has to be inserted
Step 3: Set A[I+1]=A[I]
Set A[I+1]=A[I]
Step 4: Set N=N+1
Step 5: Set A[POS]=VAL
Step 6: EXIT

WAP to insert a number at a given location in an array.

```
#include<stdio.h>
void main()
{
   int i,a[20],n,num,pos;
   printf("Enter the number of elements: ");
   scanf("%d",&n);
   printf("Enter the elements: ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("Enter the number to be inserted: ");
   scanf("%d",&num);
   printf("Enter the postion at which number has to be inserted: ");
   scanf("%d",&pos);
   for(i=n-1;i>=pos;i--)
      a[i+1]=a[i];
   a[pos]=num;
   n++;
   printf("The array after insertion of %d is :",num);
```

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

Output:

Enter the number of elements: 5

Enter the elements: 1 2 4 5 6

Enter the number to be inserted: 3

Enter the postion at which number has to be inserted: 2

The array after insertion of 3 is : 1 2 3 4 5 6

WAP to insert a number in an array that is already sorted in ascending order.

```
#include<stdio.h>
void main()
{
    int i,n,j,num,a[10];
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("Enter the number to be inserted: ");
    scanf("%d",&num);
    for(i=0;i<n;i++)
    {
        if(a[i]>num)
        {
            for(j=n-1;j>=i;j--)
            a[j+1]=a[j];
            a[i]=num;
            break;
        }
    }
    n++;
```

```
        printf("The array after insertion of %d is: ",num);
        for(i=0;i<n;i++)
            printf("\t%d",a[i]);
}
```

Output:

Enter the number of elements: 5

Enter the elements: 1 2 3 4 5

Enter the number to be inserted: 0

The array after insertion of 0 is: 0 1 2 3 4 5

Deleting an element in an array

- Deleting an element from an array means removing a data element from an already existing array.

Algorithm to delete a new element to the end of the array:

Step 1: Set upper_bound= upper_bound-1

Step 2: EXIT

Algorithm to delete element from middle of the array:

Step 1: Set I=POS// POS=position at which element has to be deleted

Step 2: Repeat 3 and 4 while I<=N-1//N=Number of elements in the array

Step 3: Set A[I] = A[I+1]

Step 4: Set I=I+1

Step 5: Set N=N-1

Step 6: EXIT

WAP to delete a number from a given location in an array.

```
#include<stdio.h>

void main()

{
```

```
    int i,a[20],n,pos;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
    for(i=0;i<n;i++)
    {   scanf("%d",&a[i]);    }
    printf("Enter the postion from which number has to be deleted: ");
    scanf("%d",&pos);
    for(i=pos;i<n-1;i++)
    a[i]=a[i+1];
    n--;
    printf("The array after deletion is :");
    for(i=0;i<n;i++)
        printf("\nA[%d]=%d",i,a[i]);
}
```

Output:

Enter the number of elements: 5

Enter the elements: 1 2 3 4 5

Enter the postion from which number has to be deleted: 4

The array after deletion is :

A[0]=1

A[1]=2

A[2]=3

A[3]=4

WAP to delete a number from an array that is already sorted in ascending order

```
#include<stdio.h>
void main()
{
        int i,n,j,num,a[10];
        printf("Enter the number of elements: ");
        scanf("%d",&n);
        printf("Enter the elements: ");
        for(i=0;i<n;i++)
        {    scanf("%d",&a[i]);    }
        printf("Enter the number to be deleted: ");
        scanf("%d",&num);
   for(i=0;i<n;i++)
        {
                if(a[i]==num)
                        {
                                for(j=i;j<n-1;j++)
                                a[j]=a[j+1];
                        }
        }
                printf("The array after deletion is");
                for(i=0;i<n-1;i++)
                printf("\t%d",a[i]);
}
```

Output:

Enter the number of elements: 5

Enter the elements: 1 2 3 4 5

Enter the number to be deleted: 3

The array after deletion is 1 2 4 5

Merging 2 arrays

- Merging of 2 arrays in a third array means first copying the contents of the first array into the third array and then copying the contents of second array into the third array.
- Hence, the merged array contains contents of the second array.

WAP to merge 2 unsorted arrays.

```
#include <stdio.h>

void main()

{

    int a1[10],a2[10],a3[10],i,n1,n2,m,index=0;

    printf("Enter the number of elements in array1:");

    scanf("%d",&n1);

    printf("Enter the elements in array1:");

    for(i=0;i<n1;i++)

       scanf("%d",&a1[i]);

    printf("Enter the number of elements in array2:");

    scanf("%d",&n2);

    printf("Enter the elements in array2:");

    for(i=0;i<n2;i++)

       scanf("%d",&a2[i]);
```

```
    m=n1+n2;
        for(i=0;i<n1;i++)
        {
                a3[index]=a1[i];
                index++;
        }
        for(i=0;i<n2;i++)
        {
                a3[index]=a2[i];
                index++;
        }
        printf("\n\nThe merged array is\n");
        for(i=0;i<m;i++)
                printf("\t Arr3[%d]=%d\n",i,a3[i]);
}
```

Output:

```
Enter the number of elements in array1:3
Enter the elements in array1:1 2 3
Enter the number of elements in array2:3
Enter the elements in array2:4 5 6
The merged array is
        Arr3[0]=1
        Arr3[1]=2
        Arr3[2]=3
```

Arr3[3]=4

Arr3[4]=5

Arr3[5]=6

WAP to merge 2 sorted arrays.

```
#include <stdio.h>
void main()
{
    int a1[10],a2[10],a3[10],i,n1,n2,m,index=0,index_1=0,index_2=0;
    printf("Enter the number of elements in array1:");
    scanf("%d",&n1);
    printf("Enter the elements in array1:");
    for(i=0;i<n1;i++)
       scanf("%d",&a1[i]);
    printf("Enter the number of elements in array2:");
    scanf("%d",&n2);
    printf("Enter the elements in array2:");
    for(i=0;i<n2;i++)
       scanf("%d",&a2[i]);
    m=n1+n2;
    while(index_1<n1&&index_2<n2)
    {
       if(a1[index_1]<a2[index_2])
       {
          a3[index]=a1[index_1];
```

```
            index_1++;
        }
        else
        {
            a3[index]=a2[index_2];
            index_2++;
        }
        index++;
    }
```

if(index_1==n1)//if elements of the first array are over and the second array has some elements

```
    {
        while(index_2<n2)
        {
            a3[index]=a2[index_2];
            index_2++;
            index++;
        }
    }
```

else if(index_2==n2) //if elements of the second array are over and the first array has some elements

```
    {
        while(index_1<n1)
        {
            a3[index]=a1[index_1];
```

```
            index_1++;
            index++;
        }
    }
    printf("\n\nThe contenets of merged array are");
    for(i=0;i<m;i++)
        printf("\n Arr[%d] = %d",i,a3[i]);
}
```

Output:

Enter the number of elements in array1:3

Enter the elements in array1:4 5 6

Enter the number of elements in array2:3

Enter the elements in array2:1 2 3

The contenets of merged array are

Arr[0] = 1

Arr[1] = 2

Arr[2] = 3

Arr[3] = 4

Arr[4] = 5

Arr[5] = 6

Searching for a value in an array

- Searching means to find whether a particular value is present in the array or not.
- If the value is present in the array then search is said to be successful and the search process gives the location of that array.

- If the value is not present, the search process displays the appropriate message.

Linear search: ALGORITHM

Step1: [Initialization] Set pos=-1

Step2: [Initialization] Set I=0

Step3: Repeat Step 4 while I<=N

Step4: IF A[I]= val

SET POS=I

PRINT POS

Go to Step 6

[END OF IF]

SET I=I+1

[END OF LOOP]

Step5: IF POS= -1,

PRINT "VALUE IS NOT PRESENT IN THE ARRAY"

[END OF IF]

Step6: EXIT

Program:

```
#include<stdio.h>
void main()
{
    int a[10],num,i,n,found=0,pos=-1;
    printf("Enter the number of elements in an array: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
```

```
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
        printf("Enter the number that has to be searched: ");
        scanf("%d",&num);
        for(i=0;i<n;i++)
    {
        if(a[i]==num)
        {
                found=1;
                pos=i;
                printf("\n%d is found in the array at position %d",num,i+1);
                break;
        }
}
        if(found==0)
        printf("Element not found in the array");
}
```

Output:

Enter the number of elements in an array: 5

Enter the elements: 50 9 6 7 1

Enter the number that has to be searched: 6

6 is found in the array at position 3

Binary Search:

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

```
void main()
{
        int i,low,high,mid,n,key,a[20];
        printf("Enter the number of elements in an array: ");
        scanf("%d",&n);
        printf("Enter the elements: ");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        printf("Enter the value to find: ");
        scanf("%d",&key);
        low=0;
        high=n-1;
    while(low<=high)
        {
                mid=(low+high)/2;
                if(a[mid]==key)
                        {
                                printf("%d found at location %d",key,mid+1);
                                break;
                        }
                else if(a[mid]<key)
                low=mid+1;
                else
                high=mid-1;
```

```
        }
        if(low>high)
        printf("%d not found in the array",key);
}
```

Output:

Enter the number of elements in an array: 5

Enter the elements: 1 2 3 4 5

Enter the value to find: 3

3 found at location 3

WAP to sort n numbers in ascending order using bubble sort technique:

```
#include<stdio.h>
void main()
{
        int i,j,n,temp,a[20];
        printf("Enter the number of elements in an array: ");
        scanf("%d",&n);
        printf("Enter the elements: ");
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
    for(i=0;i<n-1;i++)
        {
                for(j=0;j<n-1-i;j++)
                {
                        if(a[j]>a[j+1])
```

```
                    {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                    }
                }
        }
    printf("Array after implementing bubble sort:");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}
```

Output:

Enter the number of elements in an array: 5

Enter the elements: 60 9 8 5 100

Array after implementing bubble sort: 5 8 9 60 100

WAP to sort n numbers in decending order using bubble sort technique:

```
#include<stdio.h>
void main()
{
    int i,j,n,temp,a[20];
    printf("Enter the number of elements in an array: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
    for(i=0;i<n;i++)
```

```
        scanf("%d",&a[i]);
    for(i=0;i<n-1;i++)
        {
            for(j=0;j<n-1-i;j++)
            {
                if(a[j]<a[j+1])
                    {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                    }
            }
        }
    printf("Array after implememting bubble sort:");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}
```

Output:

Enter the number of elements in an array: 5

Enter the elements: 90 7 6 100 99

Array after implememting bubble sort: 100 99 90 7 6

2-Dimensional Array

Arrays with 2 dimensions are called 2 –Dimensional array or 2-D array.

Declaration of 2-D array:

data_type array_name[row_size][column_size];

data_type can be any primitive data type.

array_name is a variable name

row_size is the maximum number of rows in the array.

column_size is the maximum number of column in the array.

Example: int a[2][3];

This can be read as

R/C	Column 0	Column 1	Column 2
Row 0	a[0][0]	a[0][1]	a[0][2]
Row 1	a[1][0]	a[1][1]	a[1][2]

Initialization of 2-D array:

1. **Initialize with total number of elements:**

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

2. **Initialize with sets**

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

3. **Partial initialization**

   int a[2][3]={{1,1},{2}}

4. **Initialize without size**

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

Initialization of 2-D array:

```
for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
```

```
        {
                scanf("%d",&a[i][j]);
        }
}
```

WAP to read and display elements from 2-D array.

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

```
                printf("%d\t",a[i][j]);
            }
        printf("\n");
    }
}
```

Output:

Enter the number of rows and columns: 3,3

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

The array elements are:

1 2 3

4 5 6

7 8 9

WAP to generate Pascal's triangle.

```
#include<stdio.h>
void main()
{
    int a[5][5]={0},row=2,col,i,j;
    a[0][0]=a[1][0]=a[1][1]=1;
    while(row<5)
        {
            a[row][0]=1;
            for(col=1;col<=row;col++)
                a[row][col]=a[row-1][col-1]+a[row-1][col];
            row++;
```

```
        }
    for(i=0;i<5;i++)
        {
            printf("\n");
            for(j=0;j<=i;j++)
            printf("%d\t",a[i][j]);
        }
}
```

Output:

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
```

Operations on 2-Dimensional Array

1. Transpose
2. Sum
3. Difference
4. Product

WAP to transpose 3 X 3 matrix.

```
#include<stdio.h>
void main()
{
    int a[20][20],m,n,i,j,b[20][20];
```

```
printf("Enter the number of rows and columns: ");
scanf("%d,%d",&m,&n);
printf("Enter the elements of the array:");
for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                        {
                                scanf("%d",&a[i][j]);
                        }
        }
printf("The array elements are:\n");
for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                        {
                                printf("%d\t",a[i][j]);
                }
              printf("\n");
        }
for(i=0;i<m;i++)
    {
                for(j=0;j<n;j++)
                        {
                                b[i][j]=a[j][i];
```

```
                    }
            }
            printf("The elemnts of transposed matrix are:\n");
            for(i=0;i<m;i++)
                {
                    for(j=0;j<n;j++)
                        {
                            printf("%d\t",b[i][j]);
                }
            printf("\n");
           }
}
```

Output:

Enter the number of rows and columns: 3,3

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

The array elements are:

1 2 3

4 5 6

7 8 9

The elemnts of transposed matrix are:

1 4 7

2 5 8

3 6 9

WAP to input 2 m x n matrices and then calculate the sum of their corresponding elements and store it in third m x n matrix.

```
#include<stdio.h>
void main()
{
        int a[20][20],b[20][20],c[20][20],m,n,p,q,r,t,i,j;
        printf("Enter the number of rows and columns in first matrix: ");
        scanf("%d,%d",&m,&n);
        printf("Enter the number of rows and columns in second matrix: ");
        scanf("%d,%d",&p,&q);
        if(m!=p||n!=q)
        {
           printf("Number of rows and columns of both the matrix should be
equal");
        }
        r=m;
        t=n;
        printf("Enter the elements of the array 1:");
        for(i=0;i<m;i++)
                {
                        for(j=0;j<n;j++)
                                {
                                        scanf("%d",&a[i][j]);
                                }
                }
```

```
        printf("Enter the elements of the array 2:");
for(i=0;i<p;i++)
        {
                for(j=0;j<q;j++)
                        {
                                scanf("%d",&b[i][j]);
                        }
        }
for(i=0;i<r;i++)
        {
                for(j=0;j<t;j++)
                        {
                                c[i][j]=a[i][j]+b[i][j];
                       }
        }
printf("The elements of the resultant matrix are:\n");
for(i=0;i<r;i++)
        {
                for(j=0;j<t;j++)
                        {
                                 printf("%d\t",c[i][j]);
                        }
                printf("\n");
        }
```

```
}
```

Output:

```
Enter the number of rows and columns in first matrix: 2,2
Enter the number of rows and columns in second matrix: 2,2
Enter the elements of the array 1:2 2 2 2
Enter the elements of the array 2:2 2 2 2
The elements of the resultant matrix are:
4       4
4       4
```

WAP to input 2 m x n matrices and then calculate the product of their corresponding elements and store it in third m x n matrix.

```
#include<stdio.h>
void main()
{
        int a[20][20],b[20][20],c[20][20],m,n,p,q,k,i,j;
        printf("Enter the number of rows and columns in first matrix: ");
        scanf("%d,%d",&m,&n);
        printf("Enter the number of rows and columns in second matrix: ");
        scanf("%d,%d",&p,&q);
        if(n!=p)
        {
           printf("Matrix multiplication is not possible");
        }
        printf("Enter the elements of the array 1:");
        for(i=0;i<m;i++)
```

```
        {
            for(j=0;j<n;j++)
                {
                    scanf("%d",&a[i][j]);
                }
        }
printf("Enter the elements of the array 2:");
    for(i=0;i<p;i++)
        {
            for(j=0;j<q;j++)
                {
                    scanf("%d",&b[i][j]);
                }
        }
    for(i=0;i<m;i++)
        {
            for(j=0;j<q;j++)
                {
                   c[i][j]=0;
                   for(k=0;k<n;k++)
                       c[i][j]=a[i][k]*b[k][j]+c[i][j];
                }
        }
printf("The elements of the resultant matrix are:\n");
```

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

Output:

Enter the number of rows and columns in first matrix: 2,2

Enter the number of rows and columns in second matrix: 2,2

Enter the elements of the array 1:2 2 2 2

Enter the elements of the array 2:2 2 2 2

The elements of the resultant matrix are:

8 8

8 8

Using arrays with functions

- Putting individual elements of the array
- Passing the whole array

Passing individual elements of the array

```
#include<stdio.h>
void square(int x);
```

```
void main()
{
    int n,a[10],i;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter the elements: ");
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);
    printf("The square of given elements are: ");
    for(i=0;i<n;i++)
    square(a[i]);
}
void square(int x)
{
    printf("%d\t",x*x);
    return;
}
```

Output:

```
Enter the number of elements: 5
Enter the elements: 1 2 3 4 5
The square of given elements are: 1   4       9       16      25
```

Passing whole array

```
#include<stdio.h>
void avg(int a[]);
```

```
void main()
{
    int b[6]={1,2,3,4,5,6};
    avg(b);
}
void avg(int a[])
{
    int i,Average,sum=0;
    for(i=0;i<6;i++)
    {
        sum=sum+a[i];
    }
    Average=sum/6;
    printf("Average=%d",Average);
}
```

Output:

Average=3

Multi-Dimensional array

- A Multi-Dimensional array is an array of arrays.
- Like we have 1 index in 1-D array, 2 index in 2-D array, we have n index in n-dimensional array.

WAP to read and display 2 x 2 x 2 array.

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

```
int a[2][2][2],i,j,k;
printf("Enter the elements of the matrix: ");
for(i=0;i<2;i++)
{
    for(j=0;j<2;j++)
    {
        for(k=0;k<2;k++)
        {
            scanf("%d",&a[i][j][k]);
        }
    }
}
printf("The matrix is: \n");
for(i=0;i<2;i++)
{
    for(j=0;j<2;j++)
    {
        for(k=0;k<2;k++)
        {
            printf("a[%d][%d][%d]=%d\t",i,j,k,a[i][j][k]);
        }
        printf("\n");
    }
}
```

}

Output:

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

The matrix is:

a[0][0][0]=1 a[0][0][1]=2

a[0][1][0]=3 a[0][1][1]=4

a[1][0][0]=5 a[1][0][1]=6

a[1][1][0]=7 a[1][1][1]=8

Applications of array

- **Storing and accessing data:** Arrays are used to store and retrieve data in a specific order. For example, an array can be used to store the scores of a group of students, or the temperatures recorded by a weather station.
- **Sorting:** Arrays can be used to sort data in ascending or descending order. Sorting algorithms such as bubble sort, merge sort, and quick sort rely heavily on arrays.
- **Searching:** Arrays can be searched for specific elements using algorithms such as linear search and binary search.
- **Matrices:** Arrays are used to represent matrices in mathematical computations such as matrix multiplication, linear algebra, and image processing.
- **Stacks and queues:** Arrays are used as the underlying data structure for implementing stacks and queues, which are commonly used in algorithms and data structures.
- **Graphs:** Arrays can be used to represent graphs in computer science. Each element in the array represents a node in the graph, and the relationships between the nodes are represented by the values stored in the array.

*****End*****