MODULE-3
ARRAYS AND STRINGS

Why Arrays?

Consider a situation, where we need to store 5 integer numbers. If we use simple variable and data type concepts, then we need 5 variables of int data type and program will be something as follows:

```
#include<stdio.h>
void main()
{
        int number1;
        int number2;
        int number3;
        int number4;
        int number5;
        number1 = 10;
        number2 = 20;
        number3 = 30;
        number4 = 40;
        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);
}
```

It was simple, because we had to store just 5 integer numbers. Now let's assume we have to store *5000 integer numbers*, so what is next???

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

Examples where arrays can be used are

- List of temperatures recorded every hour in a day, or a month, or a year
- List of employees in an organization
- List of products and their cost sold by a store
- Test scores of a class of students

Definition of an Array:-Array is a collection of elements of same data type.

The elements are stored sequentially one after the other in memory.

Any element can be accessed by using

→ name of the array

→ position of element in the array (index)

Types of array

- Single dimensional array or One dimensional array
- Two dimensional array
- Multi 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 of Declaring Single Dimensional Arrays

data_type array_name[array_size];

where

data_type: can be int, float or char

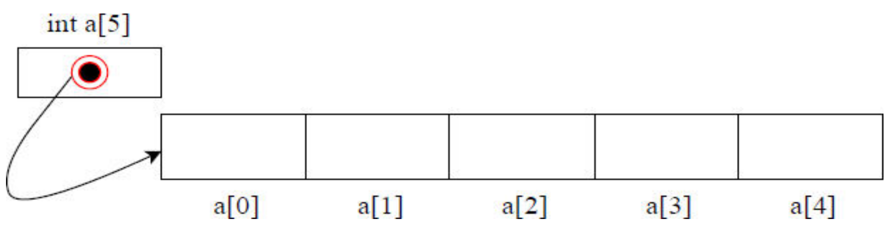
array_name: is name of the array

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

Example: int a[5];

Here a is an Integer Array that can hold up to 5 values in it.

Array Representation



Here

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

and so on..

Memory occupied by 1D array

Total memory=array size * size of datatype

For example :int a[5];

Total memory =5*sizeof (int)

= 5*2

=10 bytes.

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

1. Static initialization:- We can initialize the array in the same way as the ordinary values when they are declared.

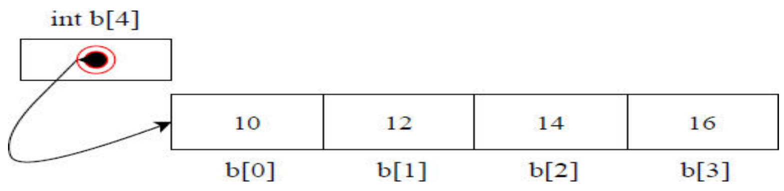
The general syntax of initialization of array is

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.



i.e in location b[0] we store the value 10, in location b[1] we store the value 12 and so on…

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**};

Here 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.

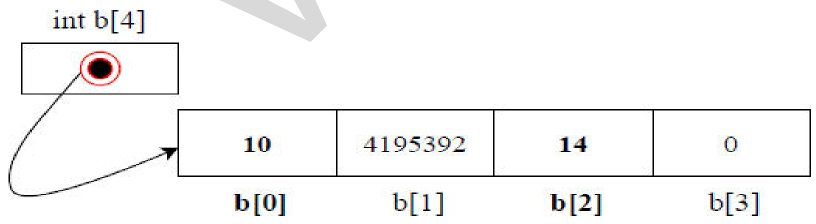
2. Initialization of array elements one by one:- Here the user has the liberty to select the locations and store values and the array. This type of initialization is not used much practically

Example

int b[4];

b[0]= 10;

b[2]=14;

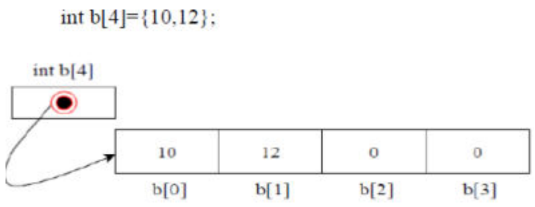


Only the array locations specified by the user will contain the values which the user wants the other locations of array will either be 0 or some garbage value.

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

Example:

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

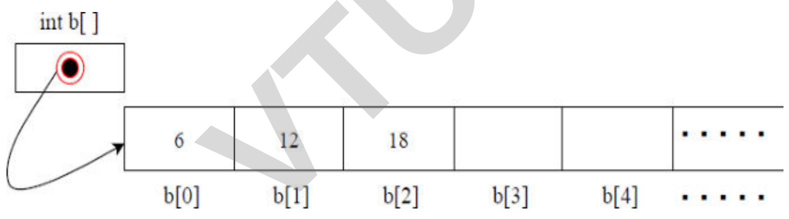


Here the remaining locations in the array will be initialized to zero.

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

5. 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 we make use of a looping construct to read the input values from the keyboard and store them sequentially in the array.

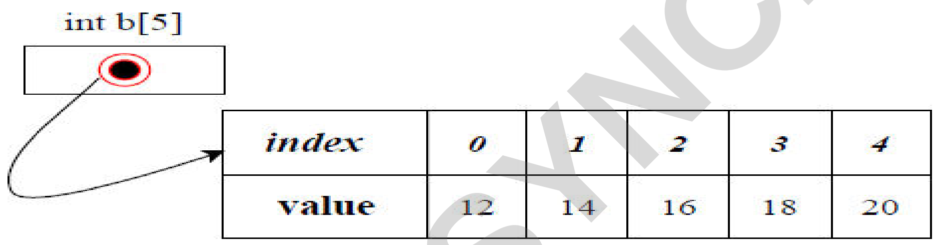
Example: **/* C program to demonstrate run time initialization*/**

```
#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:

Eg: int b[5]={12,14,16,18,20};

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]



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

To access value 16 we write b[2]=16 similarly if we wish to print the value 18 we write printf("%d",b[3]);

<u>Programming examples on one dimensional array:</u>

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

```
#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]);
          }
}
```

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]);
          }
}
```

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]);
}
```

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

```
include<stdio.h>
void main()
{
        int a[10 ],n,i,max,min
        printf("Enter the number of elements\n");
        scanf("%d",&n);
        printf("Enter the values \n");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        max=min=a[0];
        for(i=0;i<n;i++)
        {
                 if(a[i]>max)
                         max=a[i];
                if(a[i]<min)
                        min=a[i];
        }
        printf("Largest number=%d\n",max);
        printf("Smallest number=%d\n",min);
}
```

5. Write a 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]);
}
```

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

```
#include<stdio.h>
void main()
{
        int a[20 ],sum=0,esum=0,osum=0,n,i;
        float avg;
        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];
        }
        avg=sum/n;
        printf(“The sum of all numbers is %d”,sum);
        printf(“The sum of even numbers is %d”,esum);
        printf(“The sum of odd numbers is %d”,osum);
        printf(“The average of all numbers is %f”,avg);
}
```

7. Write a C Program to generate Fibonacci series using arrays

```
#include<stdio.h>
#include<conio.h>
void 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]);
        }
}
```

Sorting Techniques:-

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

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 is not suitable for large datasets as its average and worst case time complexity are of $O(n^2)$. where n is the number of items.

/*C program to sort n numbers using 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 %d elements\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;i++)
        {
                for(j=0;j<n-i;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]);
        }
}
```

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.

/*C program to sort numbers in ascending order using selection sort technique*/

```
#include<stdio.h>
void  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]);
        }
}
```

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.

/*C program to search an element in an array using linear search*/

```
#include<stdio.h>
void  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");
}
```

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

Advantages:

- Very efficicent searching technique.

Disadvantages:

- Array element should be sorted.

C program to search an element in an array using Binary search

```
#include<stdio.h>
void  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");
}
```

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];
```

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];

Represents **a** is a two dimensional integer array that holds two rows and three columns.

		0	1	2	
int a[2][3] =	**0**	a[0][0]	a[0][1]	a[0][2]	**Row 0**
	1	a[1][0]	a[1][1]	a[1][2]	**Row 1**
		Col 0	**Col 1**	**Col 2**	

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:

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

this initialization can also be written as int a[2][3] ={5,3,4,6,1,2}

		0	1	2
int a[2][3] =	**0**	5	3	4
	1	6	1	2

Accessing two dimensional array elements:- An element in a two dimensional array is accessed by using the subscripts i.e. row index and column index of the array.

The feasible way of accessing elements in a two dimensional array is by using nested loops.

Reading and printing 2 dimensional array:-

Reading 2D array

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

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

Printing 2D array

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

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

<u>Programming examples on Two dimensional array:</u>

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

```
#include<stdio.h>
void main()
{
        int a[20 ][20]m,n,i,j;
        printf("enter the row and column size\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("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");
        }
}
```

Output

```
Enter the row and column size
2 3
Enter the elements of matrix
3 4  5 9 10 12
The elements of matrix are

 3      4      5

 9      10     12
```

2. Write a C program to perform addition of two matrices

```
include<stdio.h>
void 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\n",c[i]);
                }
                printf("\n");
        }
}
```

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");
        }
}
```

4. Write a C Program to print Diagonal elements of the matrix

```
include<stdio.h>
void main()
{
        int a[20 ][20], m,n, i,j;
        printf("enter the no 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("Matrix A is \n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("%d\t",a[i][j]);
                }
                printf("\n");
        }
        printf("The diagonal Elements are\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        if(i==j)
                        {
```

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

5. Write a C Program to find sum of the the rows and columns of given matrix

```
#include<stdio.h>
void main()
{
        int a[20 ][20], m,n, i,j,sum;
        printf("enter the no 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("Martix A is Displayed as \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++)
        {
                sum=0;
                for(j=0;j<n;j++)
                {
                        sum=sum+a[i][j];
                }
                printf("sum of the elements of row %d in matrix=%d",i,sum);
        }
        for(i=0;i<m;i++)
        {
                sum=0;
                for(j=0;j<n;j++)
                {
                        sum=sum+a[j][i];
                }
                printf("sum of the elements of column %d in matrix=%d",i,sum);
        }

}
```

6. Write a C Program to find sum of the Diagonal elements of the matrix

```
#include<stdio.h>
void main()
{
        int a[20 ][20], m,n, i,j,sum=0;
        printf("enter the order of matrix");
        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("Martix A is Displayed as \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++)
        {
                sum=sum+a[i][i];
        }
        printf("The sum of the diagonal elements of matrix=%d\n",sum);
}
```

7. Write a C Program to find the largest element in given matrix

```
#include<stdio.h>
void main()
{
        int a[20 ][20], m,n, i,j,large;
        printf("enter the order 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(“Martix A is Displayed as \n”);
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf(“%d\t”,a[i][j]);
                }
                printf(“\n”);
        }

        large=0;
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                 {
                        if(a[i][j]>large)
                                large=a[i][j];
                }
        }
        printf(“The largest element in the matrix = %d\n”,large);
}
```

Multidimensional array

An array having 3 or more subscript or dimensions is known as multidimensional array.

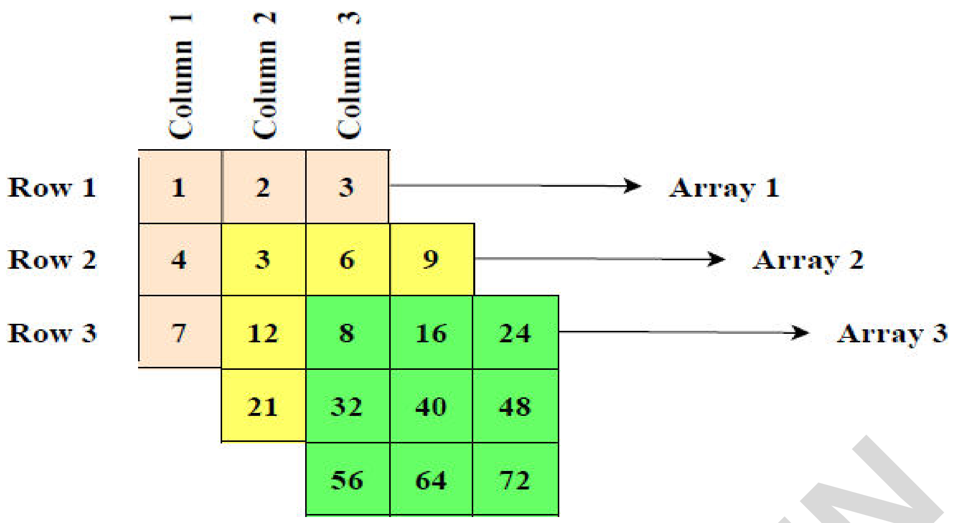
It is also known as arrays of arrays or matrix.

The general form of multidimensional array is

    data_type array_name[s1][s2][s3]….[sn];

Where s1 is the size of ith dimension.

Representation of Multidimensional Array



Strings:-

Definition:- A 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'**

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

Initialization of strings:

char str1[10]= "peter";

or

char str1[10]={'p','e','t','e','r'};

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 **{ }**

The string initialized previously will be stored in memory as

str1	0	1	2	3	4	5	6	7	8	9
	p	e	t	e	r	'\0'				

Formatted input and output:-

scanf() and printf():-

The scanf() and printf()statements are used when we are reading or displaying a single string (without space).

We do not use "&" symbol in scanf because string name itself represent the address where it is to be stored.

%s is the format specifier used for string.

/* C program to read and display string using formatted I/O function */

```
#include<stdio.h>
void main()
{
        char  str[20];
        printf("Enter a string\n");
        scanf("%s",str);
        printf("The entered string is =%s\n",str);
}
```

Unformatted input and output :-

gets() and puts():

if we want to read or display the set of strings (sentence or words with space) we make use of gets() and puts() function for single display at once

/* C program to read and display string using formatted I/O function */

```
#include<stdio.h>
void main()
{
        char  str[20];
        printf("Enter a string\n");
        gets(str);
        printf("The entered string is\n");
        puts(str);
}
```

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.

<u>Static Initialization of strings:-</u>

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 Array of strings:-

/* C program for reading and displaying Array 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]
        }
}
```

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 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);
```

/* C program to find length of the string using strlen() function */

```
#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);
}
```

```
***** OUTPUT *****

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,count;
        printf("Enter a string\n");
        scanf("%s",str1);
        count=0;
        for(i=0;str1[i]!='\0';i++)
        {
                count=count+1;
        }
        printf("Length of the String=%d\n",count);
}
```

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

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");
}
```

```
***** OUTPUT *****

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

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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void 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);
        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");
        }
}
```

String Copy:- The function **strcpy()** copies the content from one string to another string

SYNTAX

```
        strcpy(str2,str1);
```

/* C program to copy the string using strcpy function */

```
#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);
}
```

```
***** OUTPUT *****

Enter string1

mite mangalore

The copied string is = mite
```

/* C program to copy the string without using strcpy( ) function */

```
#include<stdio.h>
#include<string.h>
void 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);
}
```

```
***** OUTPUT *****

Enter string1

Mangalapuram

The Original String=Mangalapuram

The Copied String=Mangalapuram
```

String n Copy :-The **strncpy()** funtion copies the n characters of one string to another string

SYNTAX

```
                    strncpy(dest_string,source_string,n);
```

Where n is an integer value which specifies the number of characters to be copied

/* C program to illustrate strncpy( ) function */

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str1[30],str2[30];
        printf("Enter String1\n");
        gets(str1);
        printf("String 1= %s\n",str1);
        strncpy(str2,str1,10);
        printf("String 2= %s\n",str2);
}
```

```
***** OUTPUT *****


Enter String1

MANGALORE_INSTITUTE_OF_TECHNOLOGY

String 1= MANGALORE_INSTITUTE_OF_TECHNOLOGY

String 2= MANGALORE_
```

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 */

```
#include<stdio.h>
#include<string.h>
void 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);
}
```

```
***** OUTPUT *****
Harry
Enter String2
Potter
The concatenated string is=HarryPotter
```

/* C program to concatenate two strings without using strcat( ) function */

```
#include<stdio.h>
#include<string.h>
void 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[i]=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);
}
```

```
***** OUTPUT *****

Enter String1
Ronald
Enter String2
Weasley
The concatenated string is=RonaldWeasley
```

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

SYNTAX

```
strncat(string1,string2,n);
```

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

/* C program to illustrate strncat( ) function*/

```
#include<stdio.h>
#include<string.h>
void 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);
}
```

```
***** OUTPUT *****

Enter String1 and String2
Hermoine
Granger
The concatenated string is=HermoineGrang
```

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>
void main()
{
        char str1[30];
        printf("Enter String1\n");
        gets(str1);
        strrev(str1);
        printf("The Reversed string =%s\n",str1);
}
```

```
***** OUTPUT *****
Enter String1
wingardiamleviosa
The Reversed string =asoivelmaidragniw
```

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

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str1[50],str2[50];
        int i,j,len;
        printf("Enter String1\n");
        gets(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);
}
```

```
***** OUTPUT *****

Enter String1
Expectopetronum
The reversed string = munortepotcepxE
```

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

```
#include<stdio.h>
#include<string.h>
void main()
{
        char str1[50],str2[50];
        int i,j,len,x;
        printf("Enter String1\n");
        gets(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);
}
```

```
***** OUTPUT *****
malayalam
The original String =malayalam
The reversed String= malayalam
malayalam is a palindrome
```

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*/

```
#include<stdio.h>
#include<string.h>
void 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));
}
```

```
***** OUTPUT *****
Enter the string in lower case letters
dumbledore
The string in Upper case letter is= DUMBLEDORE
Enter the string in Upper case letters
HAGRID
The string in Lower case letter is= hagrid
```

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. We also make use of the function “isalpha()” which is used to check if the parsed character is an alphabet or not.

/* C program to count the number of Vowels and Consonants in a given Sentence */

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
main()
{
        char str1[50],ch;
        int i,vc=0,cc=0,ac=0,ec=0,ic=0,oc=0,uc=0;
        printf("\n Enter a sentence\n");
        gets(str1);
        printf("The entered sentence is\n");
        puts(str1);
        for (i=0;i<strlen(str1);i++)
        {
                if(isalpha(str1[i]))
                {
                        ch=str1[i];
                        if(ch=='a'||ch=='A')
                                ac++;
                        else if(ch=='e'||ch=='E')
                                ec++;
                        else if(ch=='i'||ch=='I')
                                ic++;
                        else if(ch=='o'||ch=='O')
                                oc++;
                        else if(ch=='u'||ch=='U')
                                uc++;
                        else
                                cc++;
                }
        }
        vc=ac+ec+ic+oc+uc;
        printf("\n The total number of vowels =%d\n",vc);
        printf("\n The frequency of vowel a is =%d\n",ac);
        printf("\n The frequency of vowel e is =%d\n",ec);
        printf("\n The frequency of vowel i is =%d\n",ic);
        printf("\n The frequency of vowel o is =%d\n",oc);
        printf("\n The frequency of vowel u is =%d\n",uc);
        printf("\n Total number of consonants =%d\n",cc);
}
```

/* C program to count the number of Vowels and Consonants in a given String using while loop */

```
#include<stdio.h>
void main()
{
        chat str[100];
        int i=0,vc=0,cc=0;
        printf("Enter any string\n");
        gets(str);
        while(str[i]!= '\0')
        {
                if(str[i]== 'a'|| str[i]== 'e'|| str[i]== 'i'|| str[i]== 'o'|| str[i]== 'u'|| str[i]==
                  'A'|| str[i]== 'E'|| str[i]== 'I'|| str[i]== 'O'|| str[i]== 'U')
                {
                        vc++;
                }
                else
                {
                        cc++;
                }
                i++;
        }
        printf("Number of Vowels in the string = %d\n",vc);
        printf("Number of consonants in the string=%d\n",cc);
}
```