ARRAYS

In C programming, one of the frequently arising problems is to handle similar types of data. For example: If the user wants to store marks of 100 students. This can be done by creating 100 variables individually but, this process is rather tedious and impracticable. This type of problem can be handled in C programming using arrays

An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.

It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Where arrays are used

- ❖ to store list of Employee or Student names,
- ❖ to store marks of a students,
- ❖ or to store list of numbers or characters etc.

Each element of an array is of same data type and carries the same size, i.e., int = 4 or 2 bytes.

Array position starts with 0 and end with n-1

Advantage of C Array

1) **Code Optimization**: Less code to the access the data.
2) **Easy to traverse data**: By using the for loop, we can retrieve the elements of an array easily.
3) **Easy to sort data**: To sort the elements of array, we need a few lines of code only.
4) **Random Access**: We can access any element randomly using the array

Disadvantage of Array

Prasant Dash
7008191907

Fixed Size: Whatever size, we define at the time of declaration of array, we can't exceed the limit. So, it doesn't grow the size dynamically like Linked List.

Arrays are of two types:

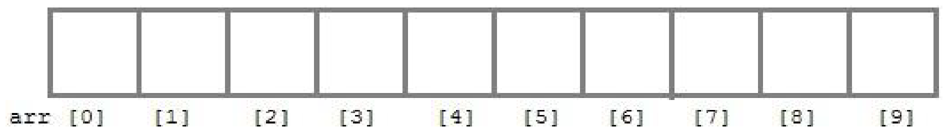
One-dimensional arrays
Multidimensional arrays

Declaration of One-dimensional Array

data-type variable-name[size/length of array];

For example:

int arr[10];



Here **int** is the data type, **arr** is the name which is constant of the array and 10 is the size of array. It means array **arr** can only contain 10 elements of **int** type. **Index** of an array starts from 0 to size-1 i.e first element of **arr** array will be stored at arr[0] address and last element will occupy arr[9].
Array stored first index (arr[0]) address by default

Initialization of an Array

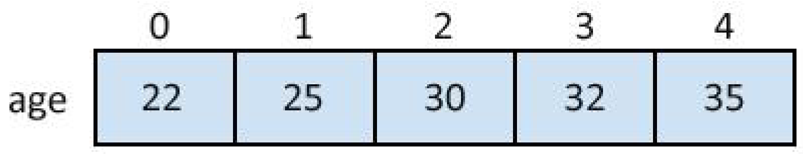
After an array is declared it must be initialized. Otherwise, it will contain garbage value (any random value). An array can be initialized at either compile time or at runtime.

Compile time Array initialization

Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,

Syntax: data_type array_name[size]={v1,v2,…vn/list of values} ;

int age[5]={22,25,30,32,35};



int marks[4]={ 67, 87, 56, 77 }; //integer array initialization

float area[5]={ 23.4, 6.8, 5.5 }; //float array initialization

int marks[4]={ 67, 87, 56, 77, 59 }; //Compile time error

One important thing to remember is that when you will give more initialize than declared array size than the **compiler** will give an error.

Different ways of initializing arrays:

1: Initializing all specified memory locations

2: Partial array initialization.

3: Initialization without size.

4: String initialization.

1 : Initilizing all specified memory locations : If the number of values to be initialized is equal to size of array. Arrays can be initialized at the time of declaration. Array elements can be initialized with data items of type int, float,char, etc.

Ex : consider integer initialization

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

During compilation, 5 contiguous memory locations are reserved by the compiler for the variable a and all these locations are initialized.

An array a is initialized as

a[0]	a[1]	a[2]	a[3]	a[4]
10	**20**	**30**	**40**	**50**
1000	**1002**	**1004**	**1006**	**1008**

If the size of integer is 2 bytes, 10 bytes will be allocated for the variable a.

Ex : consider character initialization

char b[9] = {‘E’,’D’,’U’,’C’,’A’,’T’,’I’,’O’,’N’}

Hear b is a char array store nine characters and the size of character is 1 byte, 9 bytes will be memory allocation for the b variable.

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	
E	D	U	C	A	T	I	O	N	\0

Other Example

int A[5]={1,4,6,7,8,4,5,9}

// Compilation Error: Number of initial values are more than the size of array.

Partial Array Initialization:

Partial array initialization is possible in C language. 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 locations will be initialized to zero automatically.

Ex : Consider the partial initialization

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

Hear a is an array integer of five elements and 10 bytes memory allocation but hear 3 elements are stored in given array, the next set of memory locations are automatically initialized to zero.

How to input data in array time of execution

As you can see, in above example that I have used “for loop” and “ scanf statement” to enter data into array. You can use any loop for data input.

```
int A[10],i;
for (i=0; i<10;i++)
{
printf("enter the integer number %d\n", i);
scanf("%d", &A[i]);
        }
```

Reading out data from an array

For example you want to read and display array elements, you can do it just by using any loop. Suppose array is A[10].

```
for (int i=0; i<10; i++)
{
```

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

Experiment #01 input elements in an array at time of compilation

```
#include<stdio.h>
void main()
{
int i,a[5]={20,10,30,40,50}; //declaration and initialization of array at compilation

for(i=0;i<5;i++)
printf("%d\t",a[i]);
}
Output: 20      10      30      40      50
```

Experiment #02 Runtime Array initialization

An array can also be initialized at runtime using scanf() function. This approach is usually used for initializing large array, or to initialize array with user specified values.

```
#include<stdio.h>
void main()
{
int i,a[5];

  printf("enter array elements :");
  for(i=0;i<5;i++)
  scanf("%d",&a[i]);

 printf("An Array elements are:\n");
for(i=0;i<5;i++)
printf("%d\t",a[i]);
}
Input 5 6 8 7 9
Output: 5 6 8 7 9
```

Experiment #03 W.A.P input data in an array sum all elements.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,a[50],sum,n;
 printf("\n Enter no of elements :");
 scanf("%d",&n);

                /* Reading values into Array */
                printf("\n Enter the values :");
                for(i=0;i < n;i++)
                scanf("%d",&a[i]);
 /* computation of total */
sum=0;
 for(i=0;i < n;i++)
 sum=sum+a[i];
                        /* printing of all elements of array */
                        for(i=0;i < n;i++)
                        printf("\n a[%d]=%d",i,a[i]);

 /* printing of total */
 printf("\n\nSum=%d",sum);
}
```

Experiment #04 W.A.P input data in an array find out biggest and smallest in array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[30],i,n,smallest,biggest;
                          printf("\n Enter no of elements :");
                          scanf("%d",&n);

 /* read n elements in an array */
 for(i=0 ; i < n ; i++)
   scanf("%d",&a[i]);

  biggest=smallest = a[0];

  for(i = 0 ;i < n ; i++)
  {
              if ( a[i]>biggest )
                                     biggest = a[i];

 else if ( a[i] < smallest )
   smallest = a[i];
  }

  /* Print out the Result */
   printf("\n Biggest Element : %d",biggest);
            printf("\n\n \n Smallest Element : %d",smallest);
}
```

Experiment # 05 Program for insert of an element from the specified location from Array.

```
#include<stdio.h>
int main()
{
 int arr[30],element,num,i,location;
 printf("\n\n Enter no of elements :");
 scanf("%d",&num);

 for(i=0 ; i < num ; i++)
scanf("%d",&arr[i]);

 printf("\n\n Enter the element to be inserted :");
 scanf("%d",&element);
 printf("\n Enter the location");
 scanf("%d",&location);

 /* create space at the specified location */
 for(i = num ;i >= location ; i--)
            arr[i] = arr[i-1];

num++;
 arr[location-1] = element;
 /* Print out the Result of Insertion */
 for(i = 0 ;i < num ;i++)
        printf("\n %d",arr[i]);
return(0);
}
```

Experiment # 06Program for deletion of an element from the specified location from Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[30],n,i,j;
                        printf("\n Enter no of elements :");
                        scanf("%d",&n);
 /* read n elements in an array  */
 printf("\n\n Enter %d elements :",n);
 for(i=0;i < n;i++)
 scanf("%d",&a[i]);
                                    /* read the location of the element to be deleted */
                                    printf("\n location of the element to be deleted :");
                                    scanf("%d",&j);
/* loop for the deletion  */
 while(j < n)
 {
  a[j-1]=a[j];
  j++;
 }
 n--;    /* no of elements reduced by 1 */

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

Experiment # 07 C Program to delete duplicate elements in an array (1, 2, 3, 2, 5, 4, 4).

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

```
main()
{
int a[20],i,j,k,n;

printf("\nEnter array size : ");
scanf("%d",&n);

printf("\nAccept Numbers : ",n);
for(i=0;i<n;i++)
 scanf("%d",&a[i]);

printf("\nOriginal array is : ");
for(i=0;i<n;i++)
 printf(" %d",a[i]);

printf("\nUpdated array is  : ");
for(i=0;i<n;i++)
{
   for(j=i+1;j<n;)
   {
      if(a[j]==a[i])
      {
         for(k=j;k<n;k++)
             a[k]=a[k+1];
          n--;
      }
      else
         j++;
   }
}
for(i=0;i<n;i++)
    printf("%d ",a[i]);
getch();
}
```

Experiment # 08 C Program to Merge Two arrays in C Programming

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[30],b[30],c[30],i,j,k,n1,n2;

 printf("\n Enter no of elements in 1'st array :");
 scanf("%d",&n1);

             printf("\n Enter elements in 1'st array :");
             for(i=0;i < n1;i++)
                         scanf("%d",&a[i]);

 printf("\n Enter no of elements in 2'nd array :");
 scanf("%d",&n2);

                         printf("\n Enter elements in 2nd array :");
                         for(i=0;i < n2;i++)
                          scanf("%d",&b[i]);

 i=0;j=0;k=0;  /* merging starts */

 while(i < n1 && j < n2)
```

```
{
 if(a[i] <= b[j])
  {
  c[k]=a[i];
  i++;k++;
  }
 else
  {
  c[k]=b[j];
  k++;j++;
  }
}
```

/* some elements in array 'a' are still remaining where as the array 'b' is exhausted */

```
while(i < n1)
 {
 c[k]=a[i];
 i++;k++;
 }
```

/* some elements in array b are still remaining whereas the array 'a' is exhausted */

```
while(j < n2)
 {
 c[k]=b[j];
 k++;j++;
 }
```

```
/* Displaying elements of array 'c' */

    printf("\nMerged array is :");

    for(i=0;i < n1+n2;i++)
      printf("\t %d",c[i]);

}
```

Selection sort

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of O(n2), where n is the number of items.

Algorithm

Step 1 − Set MIN to location 0
Step 2 − Search the minimum element in the list
Step 3 − Swap with value at location MIN
Step 4 − Increment MIN to point to next element
Step 5 − Repeat until list is sorted

How it works

Let the elements of array are -

12	29	25	8	32	17	40

At present, **12** is stored at the first position, after searching the entire array, it is found that **8** is the smallest value.

12	29	25	8	32	17	40

So, swap 12 with 8. After the first iteration, 8 will appear at the first position in the sorted array.

8	29	25	12	32	17	40

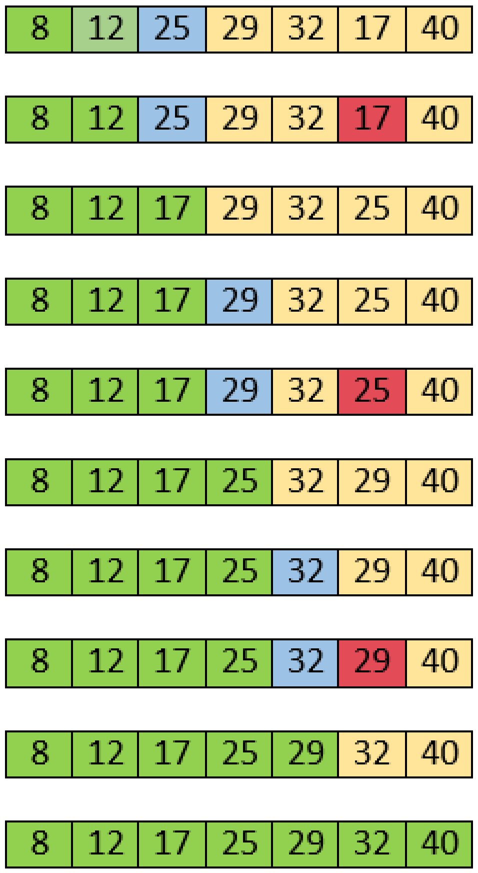
For the second position, where 29 is stored presently, we again sequentially scan the rest of the items of unsorted array. After scanning, we find that 12 is the second lowest element in the array that should be appeared at second position.

8	29	25	12	32	17	40

Now, swap 29 with 12. After the second iteration, 12 will appear at the second position in the sorted array. So, after two iterations, the two smallest values are placed at the beginning in a sorted way.

8	12	25	29	32	17	40

The same process is applied to the rest of the array elements. Now, we are showing a pictorial representation of the entire sorting process.



Now, the array is completely sorted.

Selection sort complexity

Now, let's see the time complexity of selection sort in best case, average case, and in worst case. We will also see the space complexity of the selection sort.

1. Time Complexity

Best Case	$O(n^2)$
Average Case	$O(n^2)$
Worst Case	$O(n^2)$

- **Best Case Complexity -** It occurs when there is no sorting required, i.e. the array is already sorted. The best-case time complexity of selection sort is **$O(n^2)$**.

- **Average Case Complexity -** It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of selection sort is **O(n^2)**.
- **Worst Case Complexity -** It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of selection sort is **O(n^2)**.

2. Space Complexity

Space Complexity O(1)

Experiment # 08 W.A.P input data in an array display ascending order using selection sort

```
        #include<stdio.h>
        #include<conio.h>
        void main()
        {
        int a[100],i,no,j,temp;
                printf("enter the rang:");
                scanf("%d",&no);
                        for(i=0;i<no;i++)
                        {
                        printf("enter data in a array:");
                        scanf("%d",&a[i]);
                        }
                for(i=0;i<no;i++)
                {
                        for(j=i+1;j<no;j++)
                        {
```

```
                    if(a[i]>=a[j])
                    {
                    temp=a[i];
                    a[i]=a[j];
                    a[j]=temp;
                    }//end if
               }//end inner loop
          }//end outer loop
                    printf("sorting data \n");
                    for(i=0;i<no;i++)
                    printf(" %d",a[i]);

     }//end main
```

Experiment # 09 W.A.P input data in an array display ascending order using bubble sort

```
          #include<stdio.h>
          #include<conio.h>
          int main()
          {
          int a[100],i,no,j, temp;

          printf("enter the rang:");
          scanf("%d",&no);
               for(i=0;i<no;i++)
               {
               printf("enter data in a array:");
               scanf("%d",&a[i]);
               }

               for(i=0;i<no;i++)
```

```
                {
                        for(j=0;j<no-i;j++)
                        {
                                        if(a[j]>a[j+1])
                                        {
                                        temp=a[j];
                                        a[j]=a[j+1];
                                        a[j+1]=temp;
                                        }
                        }
                }
        for(i=0;i<no;i++)
        printf(" %d",a[i]);
        return 0;
        }
```

Experiment # 10 W.A.P input data in an array display ascending order using insertion sort

```
        #include<stdio.h>
        #include<conio.h>
        int main()
        {
        int a[100],i,no,j,temp;

        printf("enter the rang:");
        scanf("%d",&no);
                for(i=0;i<no;i++)
                {
                printf("enter data in a array:");
                scanf("%d",&a[i]);
                }
                for(i=1;i<no;i++)
                {
                temp=a[i];
                for(j=i-1;(j>=0)&&temp<a[j];j--)
                a[j+1]=a[j];
                a[j+1]=temp;
                }
        for(i=0;i<no;i++)
        printf(" %d ",a[i]);
        return 0 ;
        }
```

Experiment # 11 Searching element in array print present or not with which location (linear search)

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

        int  main()
        {
         int a[30],x,n,i,p=0;

         printf("\n Enter no of elements :");
        scanf("%d",&n);

        printf("\n Enter the values :");
        for(i=0;i < n;i++)
        scanf("%d",&a[i]);
```

```
printf("\n\nEnter the elements to be searched:");/* read the element to be searched
*/
scanf("%d",&x);

for(i=1;i<=n;i++)
        {
                if(a[i]==x)
                {

                                printf("found at the location =%d",i+1);
                                p=1;
                break;
                }
        }

if(p==1)
        printf("\n found at the values =%d",x);
else
        printf("\n  not found");

}
```

Experiment # 12 Searching element in array print present or not with which location (Binary search)

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int i, arr[10], search, first, last, middle;
   printf("Enter 10 elements (in ascending order): ");
   for(i=0; i<10; i++)
    scanf("%d", &arr[i]);


           printf("\nEnter element to be search: ");
   scanf("%d", &search);
   first = 0;
   last = 9;
   middle = (first+last)/2;
   while(first <= last)
   {
       if(arr[middle]<search)
           first = middle+1;
       else if(arr[middle]==search)
       {
           printf("\nThe number, %d found at Position %d", search, middle+1);
           break;
       }
       else
           last = middle-1;
       middle = (first+last)/2;
   }
   if(first>last)
       printf("\nThe number, %d is not found in given Array", search);

   return 0;
}
```

Assignment

Q1. WAP to take the input size of array and then take the input for those many numbers and print the Armstrong numbers?

Q2.WAP to take the input size of two arrays and then take the input and print the common elements?

Two-Dimensional Arrays

The two dimensional array in C language is represented in the form of rows and columns, also known as matrix. It is also known as array of arrays or list of arrays.

The two dimensional, three dimensional or other dimensional arrays are also known as multidimensional arrays.

2D arrays are created to implement a relational database lookalike data structure

Declaration of two dimensional Array

data_type array_name[size1][size2];

Example

int twodimen[4][3];

Example :

int A[3][4];

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

Memory allocation

Data type size*(size of row*columns)

```
#include<stdio.h>
void main()
{
int a[3][4];
printf("%d",sizeof(a));
}
Output:24
```

Initialization of 2D Array

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

Experment:01

//Accessing 2-D Array Elements In C Programming

```
#include<stdio.h>
#include<conio.h>
int  main()
{
int i,j,a[3][3];

for(i=0;i<3;i++)
  for(j=0;j<3;j++)
  {
  printf("\nEnter the a[%d][%d] = ",i,j);
  scanf("%d",&a[i][j]);
  }
for(i=0;i<3;i++)
{
  for(j=0;j<3;j++)
    printf("%d\t",a[i][j]);
printf("\n");
}
return 0;

}
```

Experment:02

//C program for addition of two matrices in C

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

void main()
{
int i,j,a[10][10],b[10][10],c[10][10],m1,n1,m2,n2;

printf("\nEnter the number of Rows of Mat1 : ");
scanf ("%d",&m1);

printf("\nEnter the number of Columns of Mat1 : ");
scanf ("%d",&n1);

/* Accept the Elements in m x n Matrix */
```

```
for(i=0;i<m1;i++)
    for(j=0;j<n1;j++)
    {
    printf("Enter the Element a[%d][%d] : ",i,j);
    scanf("%d",&a[i][j]);
    }

printf("\nEnter the number of Rows of Mat2 : ");
scanf ("%d",&m2);

printf("\nEnter the number of Columns of Mat2 : ");
scanf ("%d",&n2);

/* Before accepting the Elements Check if no of
   rows and columns of both matrices is equal */

if ( m1 != m2 || n1 != n2 )
 {
 printf("\n Order of two matrices is not same ");
 exit(0);
 }

for(i=0;i<m2;i++)
    for(j=0;j<n2;j++)
    {
    printf("Enter the Element b[%d][%d] : ",i,j);
    scanf("%d",&b[i][j]);
    }



/* Addition of two matrices */

for(i=0;i<m1;i++)
    for(j=0;j<n1;j++)
    {
    c[i][j] = a[i][j] + b[i][j] ;
    }

/* Print out the Resultant Matrix */

printf("\nThe Addition of two Matrices is : \n");

for(i=0;i<m1;i++)
 {
    for(j=0;j<n1;j++ )
    {
    printf("%d\t",c[i][j]);
    }
  printf("\n");
 }
}
```

Experiments #03//Addition of All Elements in Matrix

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,a[10][10],sum,m,n;

        printf("\nEnter the number of Rows : ");
        scanf ("%d",&m);
printf("\nEnter the number of Columns : ");
scanf ("%d",&n);
/* Accept the Elements in m x n Matrix */
for(i=0;i<m;i++)
     for(j=0;j<n;j++ )
     {
     printf("Enter the Element a[%d][%d] : ", i , j);
     scanf("%d",&a[i][j]);
     }
/* Addition of all Elements */
sum = 0;
for(i=0;i<m;i++ )
     for(j=0;j<n;j++ )
     {
     sum = sum + a[i][j];
     }
printf("\nThe Addition of All Elements in the Matrix : %d",sum);
}
```

Experiments #04//Multiplication Of Two Matrices

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

void main()
{
int a[10][10],b[10][10],c[10][10],i,j,k;
int sum=0;

printf("\nEnter First Matrix : \n");

for(i=0;i<3;i++)
  {
  for(j=0;j<3;j++)
  scanf("%d",&a[i][j]);
  }

printf("\nEnter Second Matrix:\n");
for(i=0;i<3;i++)
  {
  for(j=0;j<3;j++)
  scanf("%d",&b[i][j]);
  }

printf("The First Matrix Is:\n");

for(i=0;i<3;i++)//print the first matrix
```

```
  {
  for(j=0;j<3;j++)
  printf(" %d ",a[i][j]);
  printf("\n");
  }
printf("The Second Matrix Is:\n");

for(i=0;i<3;i++) // print the second matrix
  {
  for(j=0;j<3;j++)
  printf(" %d ",b[i][j]);
  printf("\n");
  }

for(i=0;i<=2;i++)
  for(j=0;j<=2;j++)
  {
  sum = 0;
   for(k=0;k<=2;k++)
    sum = sum + a[i][k] * b[k][j];
  c[i][j]=sum;
  }

printf("\nMultiplication Of Two Matrices :\n\n");

for(i=0;i<3;i++)
{
   for(j=0;j<3;j++)
   printf(" %d ",c[i][j]);
printf("\n");
}
}
```

Experiments:05//Addition of Diagonal Elements in Matrix

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

void main()
{
int i,j,a[10][10],sum,m,n;

printf("\nEnter the number of Rows : ");
scanf ("%d",&m);

        printf("\nEnter the number of Columns : ");
        scanf ("%d",&n);

        for(i=0;i<m;i++ )
     for(j=0;j<n;j++)
     {
     printf("Enter the Element a[%d][%d] : ", i , j);
     scanf("%d",&a[i][j]);
     }

for(i=0;i<3;i++)//print the first matrix
```

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

sum = 0;
for(i=0;i<m;i++ )
     for(j=0;j<n;j++)
     {
      if ( i == j )
      sum = sum + a[i][j];
     }

printf("\nSum of All Diagonal Elements in Matrix : %d",sum);
}
```

Experiments #06//C Program to find addition of Lower Triangular Elements in C ProgrammingS

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

void main()
{
int i,j,a[10][10],sum,m,n;

printf("\nnEnter the number of Rows : ");
```

Prasant Dash
7008191907

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

printf("\n\nEnter the number of Columns : ");
scanf ("%d",&n);

for( i = 0 ; i < m ; i++ )
    for( j = 0 ; j < n ; j++ )
    {
    printf("Enter the Element a[%d][%d] : ", i , j);
    scanf("%d",&a[i][j]);
    }

        printf("\nThe elements in the matrix are: \n");
  for(int i=0;i<m;i++)     //Print the matrix
  {
     for(int j=0;j<n;j++)
     {
        printf("%d ",a[i][j]);
     }
     printf("\n");
  }

if(m==n)             //If number of rows and columns equal
  {
    int upsum=0;
    for(int i=0;i<m;i++)
    {
```

```
                for(int j=0;j<n;j++)
                {
                    if(i>=j)        //Traverse only in the upper triangle
                    upsum=upsum+a[i][j];   //Add the elements
                }
              }
              //Print the sum of upper triangular elements
              printf("\nThe sum of upper triangular matrix is %d",upsum);
          }
          else
          {                //Not possible to declare upper triangular elements
              printf("Not Possible to display lower triangular elements sum");
          }

       }
```

Experiments :07//Print the sum of upper triangular elements

```
       #include <stdio.h>

       int main()
       {
          int m,n;                //Matrix Size Declaration
          printf("Enter the number of rows and column: \n");
          scanf("%d %d",&m,&n);   //Matrix Size Initialization
          int arr[10][10];        //Matrix Size Declaration
          printf("\nEnter the elements of the matrix: \n");
          for(int i=0;i<m;i++)    //Matrix Initialization
          {
```

```
    for(int j=0;j<n;j++)
    {
        scanf("%d",&arr[i][j]);
    }
}
printf("\nThe elements in the matrix are: \n");
for(int i=0;i<m;i++)     //Print the matrix
{
    for(int j=0;j<n;j++)
    {
        printf("%d ",arr[i][j]);
    }
    printf("\n");
}
if(m==n)               //If number of rows and columns equal
{
   int upsum=0;
   for(int i=0;i<m;i++)
   {
      for(int j=0;j<n;j++)
      {
           if(i<=j)          //Traverse only in the upper triangle
           upsum=upsum+arr[i][j];    //Add the elements
      }
    }
    //Print the sum of upper triangular elements
    printf("\nThe sum of upper triangular matrix is %d",upsum);
```

```
        }
        else
        {                 //Not possible to declare upper triangular elements
           printf("Not Possible to display lower triangular elements sum");
        }
        return 0;
     }
```

Multidimensional Arrays

How to initialize a multidimensional array?

Initialization of a three dimensional array.

You can initialize a three dimensional array in a similar way like a two dimensional array. Here's an **example**

```
int test[2][3][4] = {
{ {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },
{ {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }
        };

#include <stdio.h>
int main()
{
// this array can store 12 elements
int i, j, k, test[2][3][2];
printf("Enter 12 values: \n");
for(i = 0; i < 2; ++i) {
for (j = 0; j < 3; ++j) {
for(k = 0; k < 2; ++k ) {
scanf("%d", &test[i][j][k]);
}
}
}
// Displaying values with proper index.
printf("\nDisplaying values:\n");
for(i = 0; i < 2; ++i) {
for (j = 0; j < 3; ++j) {
for(k = 0; k < 2; ++k ) {
printf("test[%d][%d][%d] = %d\n", i, j, k, test[i][j][k]);
}
}
}
return 0;
}
```

Output

```
Enter 12 values:
1    2    3    4    5    6    7    8    9    10    11    12
Displaying Values:
test[0][0][0] = 1
test[0][0][1] = 2
test[0][1][0] = 3
test[0][1][1] = 4
test[0][2][0] = 5
test[0][2][1] = 6
test[1][0][0] = 7
test[1][0][1] = 8
test[1][1][0] = 9
test[1][1][1] = 10
test[1][2][0] = 11
test[1][2][1] = 12
```