UNIT-1/ ARRAYS

Introduction to array:

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.

C array is beneficial if you have to store similar elements. For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variables for the marks in the different subject. Instead of that, we can define an array which can store the marks in each subject at the contiguous memory locations.

By using the array, we can access the elements easily. Only a few lines of code are required to access the elements of the array.

Properties of Array:

The array contains the following properties.

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.

- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Advantage of C Array

1) **Code Optimization**: Less code to the access the data.

2) **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.

3) **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.

4) **Random Access**: We can access any element randomly using the array.

Disadvantage of C Array

1) **Fixed Size**: Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

Declaration of C Array

We can declare an array in the c language in the following way.

- data_type array_name[array_size];

Now, let us see the example to declare the array.

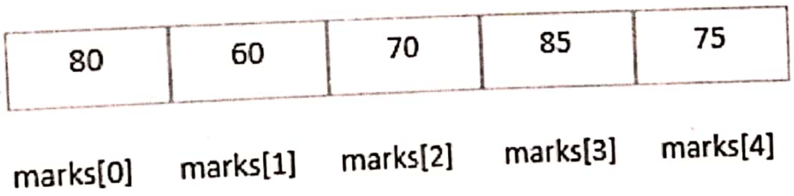
- int marks[5];

Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.

Initialization of C Array

The simplest way to initialize an array is by using the index of each element. We can initialize each element of the array by using the index. Consider the following example.

```
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
marks[4]=75;
```

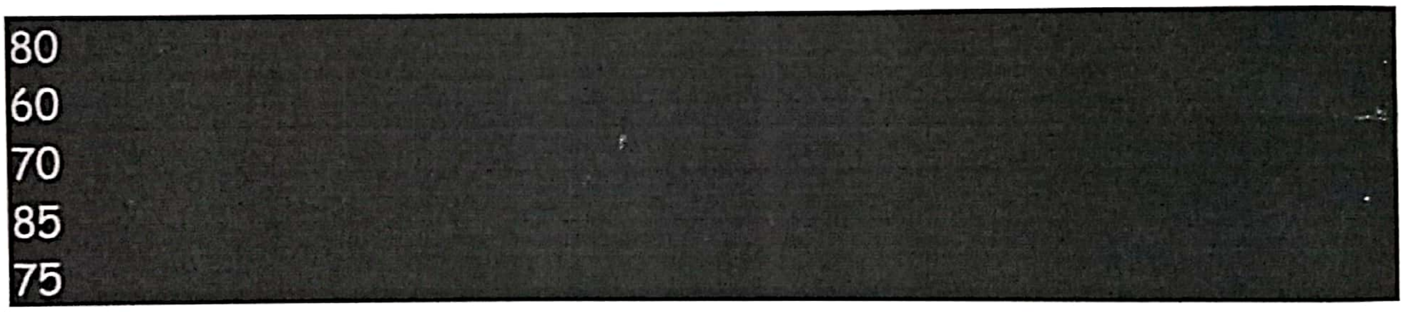


Initialization of Array

Array example

```
#include<stdio.h>
int main()
{
int i=0;
int marks[5];//declaration of array
marks[0]=80;//initialization of array
marks[1]=60;
marks[2]=70;
marks[3]=85;
      marks[4]=75;
//traversal of array
      for(i=0;i<5;i++)
      {
      printf("%d \n",marks[i]);
      }//end of for loop
      return 0;
      }
```

Output



```
80
60
70
85
75
```

C Array: Declaration with Initialization

We can initialize the c array at the time of declaration. Let's see the code.

- **int** marks[5]={20,30,40,50,60};

In such case, there is **no requirement to define the size**. So it may also be written as the following code.

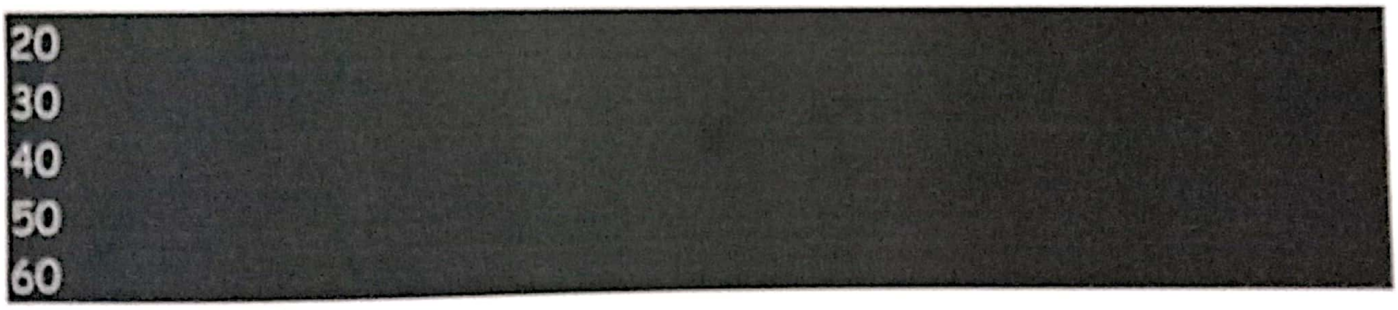
- int marks[]={20,30,40,50,60};

Let's see the C program to declare and initialize the array in C.

```
#include<stdio.h>
int main(){
int i=0;
int marks[5]={20,30,40,50,60};//declaration and initialization of array

//traversal of array
for(i=0;i<5;i++){
printf("%d \n",marks[i]);
}
return 0;
 }
```

Output



```
20
30
40
50
60
```

Program to print the largest and second largest element of the array.

```
#include<stdio.h>
void main ()
{
    int arr[100],i,n,largest,sec_largest;
    Printf ("Enter the size of the array?");
    scanf("%d",&n);
printf ("Enter the elements of the array?");
for(i = 0; i<n; i++)
{
        scanf("%d",&arr[i]);
}
    largest = arr[0];
    sec_largest = arr[1];
    for(i=0;i<n;i++)
    {
        if(arr[i]>largest)
        {
            sec_largest = largest;
            largest = arr[i];
        }
        else if (arr[i]>sec_largest && arr[i]!=largest)
        {
            sec_largest=arr[i];
        }
    }
    printf("largest = %d, second largest = %d",largest,sec_largest)
;

  }
```

Two Dimensional Array in C

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns. However, 2D arrays are created to implement a relational database lookalike data structure. It provides ease of holding the bulk of data at once which can be passed to any number of functions wherever required.

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

- data_type array_name[rows][columns];

Consider the following example.

- int twodimen[4][3];

Here, 4 is the number of rows, and 3 is the number of columns.

Initialization of 2D Array in C

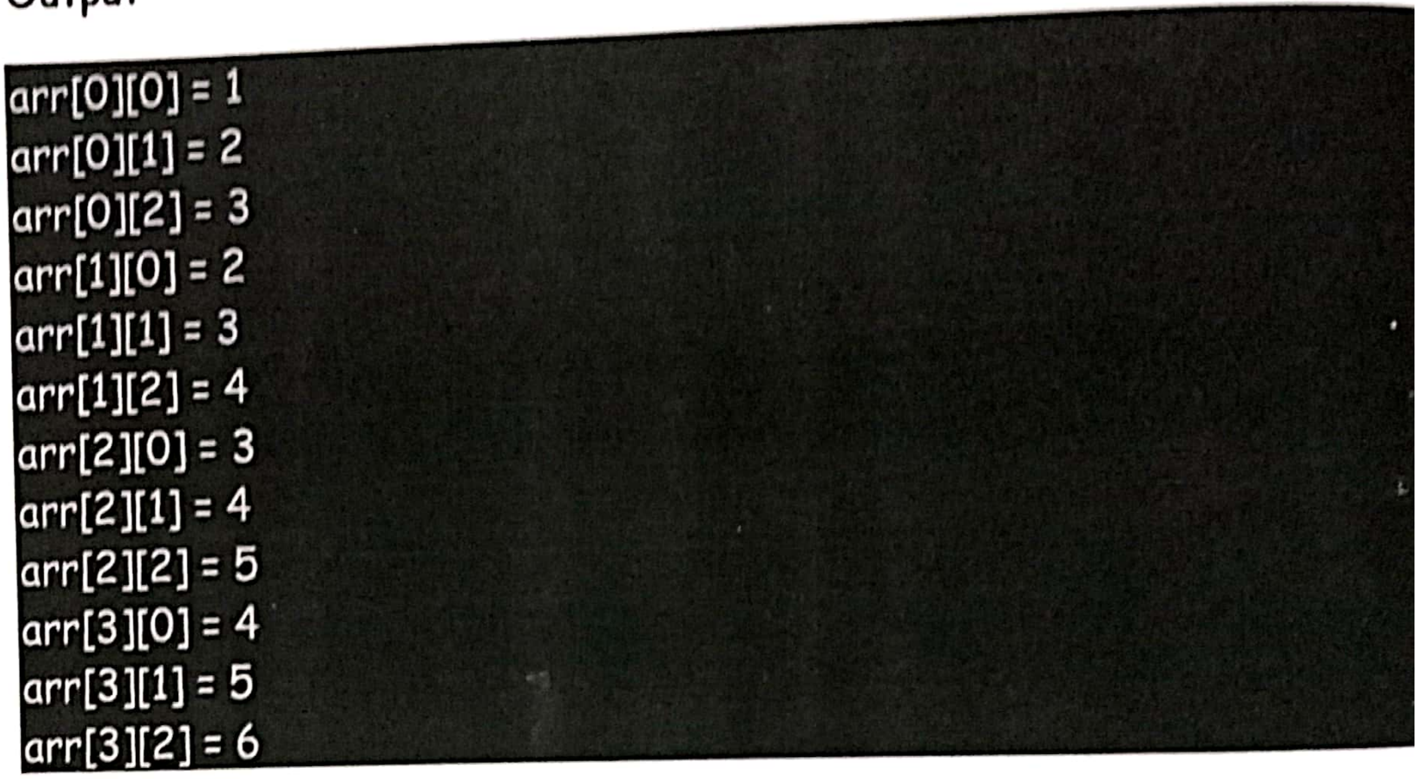
In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

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

Two-dimensional array example in C

```
#include<stdio.h>
int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++)
{
 for(j=0;j<3;j++)
{
        printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
}//end of j
    }//end of i
    return 0;
    }
```

Output

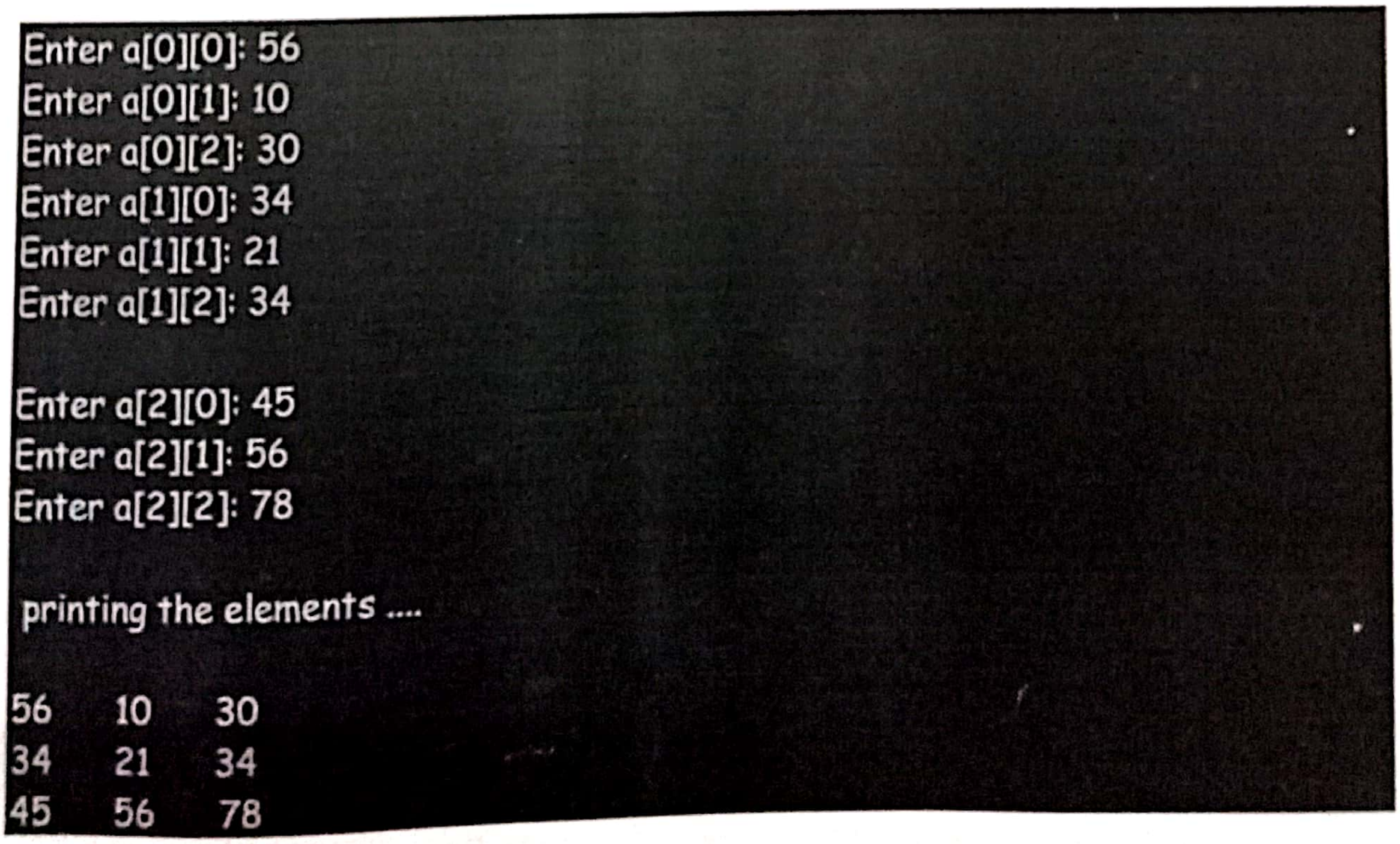


```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```

2D array example: Storing elements in a matrix anc printing it.

```
#include <stdio.h>
void main ()
{
    int arr[3][3],i,j;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
    {
        printf("Enter a[%d][%d]: ",i,j);
                scanf("%d",&arr[i][j]);
    }
        }
        printf("\n printing the elements ....\n");
        for(i=0;i<3;i++)
        {
            printf("\n");
            for (j=0;j<3;j++)
            {
                printf("%d\t",arr[i][j]);
            }
        }
    }
```

Output



```
Enter a[0][0]: 56
Enter a[0][1]: 10
Enter a[0][2]: 30
Enter a[1][0]: 34
Enter a[1][1]: 21
Enter a[1][2]: 34

Enter a[2][0]: 45
Enter a[2][1]: 56
Enter a[2][2]: 78

printing the elements ....

56      10      30
34      21      34
45      56      78
```

Accessing Elements in an Array

An array is a group of related data items that share a common name. A particular value in an array is identified by using its "index number" or "subscript".

The advantage of an array is as follows −

- The ability to use a single name to represent a collection of items and to refer to an item by specifying the item number enables the user to develop concise and efficient programs.

The syntax for declaring array is as follows −

datatype array_name [size];

For example,

float height [50]

This declares 'height' to be an array containing 50 float elements.

int group[10]

This declares the 'group' as an array to contain a maximum of 10 integer constants.

Individual elements are identified using "array subscripts". While complete set of values are referred to as an array, individual values are called "elements".

Accessing the array elements is easy by using an array index.

Example

Following is the C program for accessing an array −

```
#include<stdio.h>
int main(){
   int array[5],i ;
   array[3]=12;
   array[1]=35;
   array[0]=46;
   printf("Array elements are: ");
```

Scanned with CamScanner

```
  for(i=0;i<5;i++){
    printf("%d ",array[i]);
  }
  return 0;
}
```

Output

When the above program is executed, it produces the following result –

```
Array elements are: 46 35 38 12 9704368
```

Array[2] and array[4] prints garbage values because we didn't enter any values in that locations

Reading the elements of an array

For instance, let us take an example,

```
int i;

int marks [4];

for ( i = 0; i< = 3; i++ )

{

printf ( "/n Input marks of the given subject ");

scanf (" %d &marks [i];

}
```

Here, the 'for' loop will cause the process to ask for and receive a student's marks 4 times. The first time in the loop, 'i' has 0 value. So, scanf () function will cause the value type storing in the array element marks[0]. This loop will work until and unless the value of i is 3.

Accessing Data

We know that for reading data in the array, there is a use of loop for loading data.

Let us suppose, we need to find out the total marks of 5 subjects.

then the program for it will look like,

```
int marks [5] , i, sum = 0;

for ( i=0, i< 5; i++)

sum = sum + marks [i];

printf ( "/n Total marks  = %d", sum );
```

The above program accepts the marks of five subjects from the user and stores them in an array named marks. This is only known as the accessing of data. That is, the data input that the user provides is accessed using an array.

Memory Representation of Array

Arrays are represented with diagrams that represent their memory use on the computer. These are represented by the square boxes that represent every bit of the memory. We can write the value of the element inside the box. Below is the figure that represents the

memory representation in an array. Each box represents the amount the memory needed to hold one array of elements. The pointers in it hold the memory address of the other data. They are represented by a black disk with an arrow to the referring data.
Let us write code and represent the memory of it.

```
int a [5];

. . . . .

a[0] =1;

for (int i = 1; i< 5; i++)

{

a[i] = a[i-1]* 2;

}
```

Row-Major order or Column-Major order

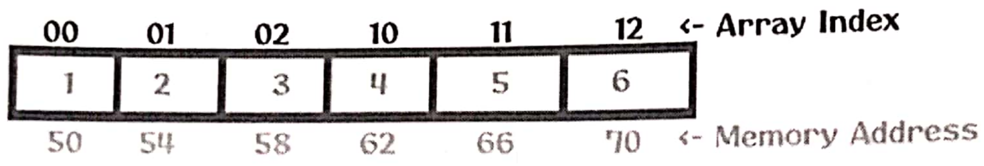
For example, consider the two-dimensional array declarations as shown in the image below.

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

Row Major Order and Column Major Order in C Initialization

Row-Major Order

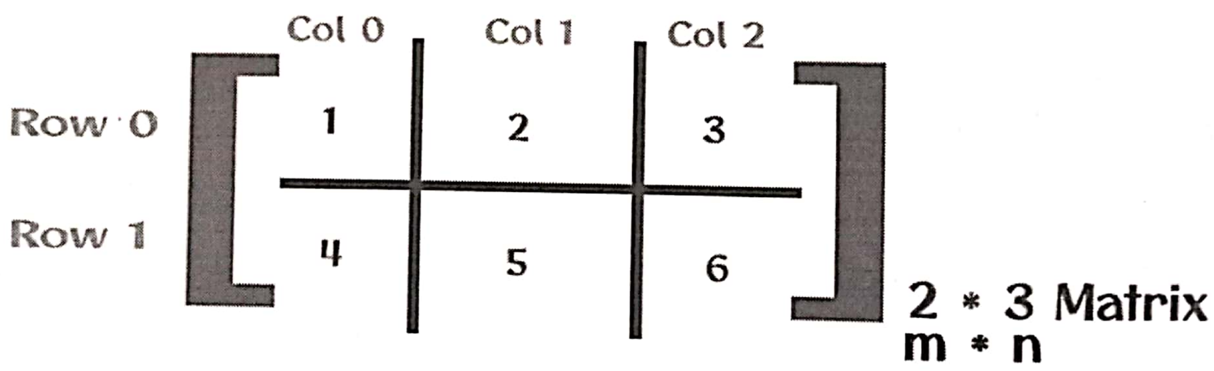
If the compiler is implemented using row-major order, then the values will get stored in the memory, as shown in the image below.



In the above image, the values 1, 2, 3, 4, 5, and 6 are stored at starting memory locations 50, 54, 58, 62, 66, and 70 (Assume int takes 4 bytes of space).

For example, if we want to access value two from memory, we must declare a[0][1].

This declaration is made under the visualization of a two-dimensional array as a matrix, as shown in the image below.



Now this declaration has to be converted into a memory address.

How will the compiler convert the [0][1] to a memory address?

The compiler uses the formula given below to convert it into a memory address.

((i * n) + j) * size of each element + Base Address

Row major order formula

In the above formula based upon our example, i and j are the array indexes; in this case, it is zero and one.

Also, m and n are the matrix indexes; in this case, it is two and three.

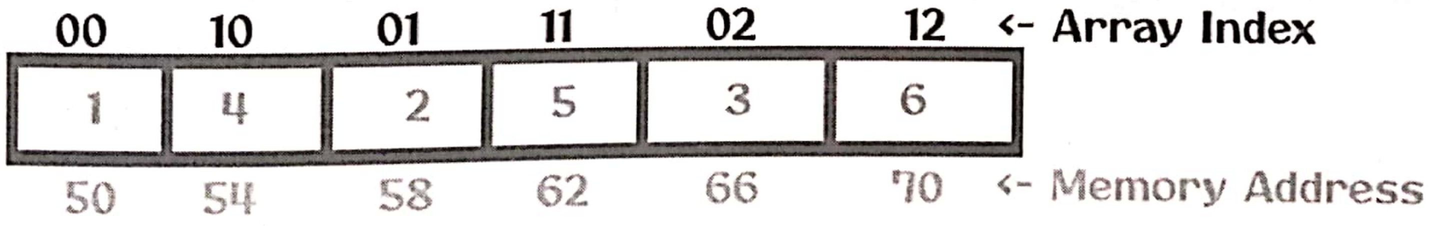
The base address is 50, and the size of the int is 4 bytes.

By substituting the corresponding values, we will get the memory address as 54.

((0*3) + 1) * 4 + 50 = 54.

Column Major Order

If the compiler is implemented by column-major order, the array elements will be stored in the memory, as shown in the image below.



Similar to the above discussion on row-major order, the formula for column-major order is shown below.

$((j * m) + i) *$ size of each element + Base Address

Multidimensional Array

C language also **allows multidimensional arrays**, i.e.` arrays that can hold elements in rows as well as columns.

Declaration:

While declaring the multidimensional array, one must specify the length of all dimensions except the left one because that is optional.

Example

Declaring array in the below manner will result in an error as dimensions other than left most is not specified.

```
Int a[][][2]={

{{1, 2}, {3, 4}},

{{5, 6}, {7, 8}}

}
```

Example

Below is one of the correct syntax for declaration of multidimensional array in C.

```
Int a[][3]={

{52,56,86},{44,6,21}

}
```

Displaying array elements:

C Program to Print an Array using do-while loop

```
#include <stdio.h>
int main()
{
  int arr[] = {10, 20, 30, 40, 50};

  // display array
  printf("Using do-while loop: \n");
  int i = 0;
  do {
    printf("%d ", arr[i]);
    i++;
  } while(i<5);

  return 0;
}
```
Output:-

```
10 20 30 40 50
```

C Program to Print an Array using for loop

```
#include <stdio.h>
int main()
{
  int arr[] = {10, 20, 30, 40, 50};

  // display array
  for(int i=0; i<5; i++) {
     printf("%d ", arr[i]);
  }

  return 0;
}
```
Output:-

```
10 20 30 40 50
```

Sorting array:

Program

Following is the C program to sort an array in an ascending order -

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

Output:

When the above program is executed, it produces the following result -enter number of elements in an array

```
5

Enter the elements

12

23

89

11

22

The numbers in ascending order is:

11

12

22

23

89
```