Unit 5: Arrays

- It is group of logically related data, stored in contiguous blocks of memory under common name.
- An Array is homogeneous or similar type of data items under common name.
- Data items or elements of arrays are separated from each other by subscript or index.
- Array is indirect pointer.
- C Supports following arrays.
    1. One Dimensional Arrays
    2. Two or Multi-Dimensional Arrays.
- One dimensional arrays are represented as set of values in one row.
- Multi-dimensional arrays are views as table containing data ie row & column.

One Dimensional or Single Dimensional Array:-

- It is list of data values of same data type & stored in one row only.

1. Declaring One Dimensional Array:

We must declare array before use, and following is the syntax:

Syntax:-

```
data_type  arr_name[size];
```

In above syntax,

- data_type-can be'char','int','float'or'double'.
- arr_name-is similar to normal variables name.
- size- is is the maximum size of array & size should be integer constant.

Example:-

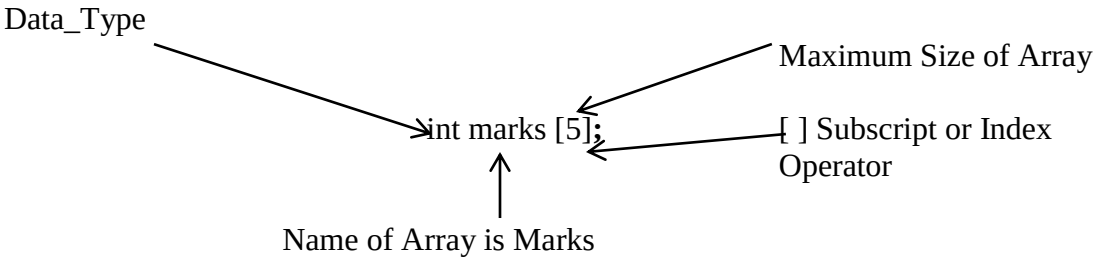


Fig 1:-Array Declaration.

- In primary memory array will get contiguous block of memory as shown in following declaration of integer array "a" of size 5.

int a[5] ;

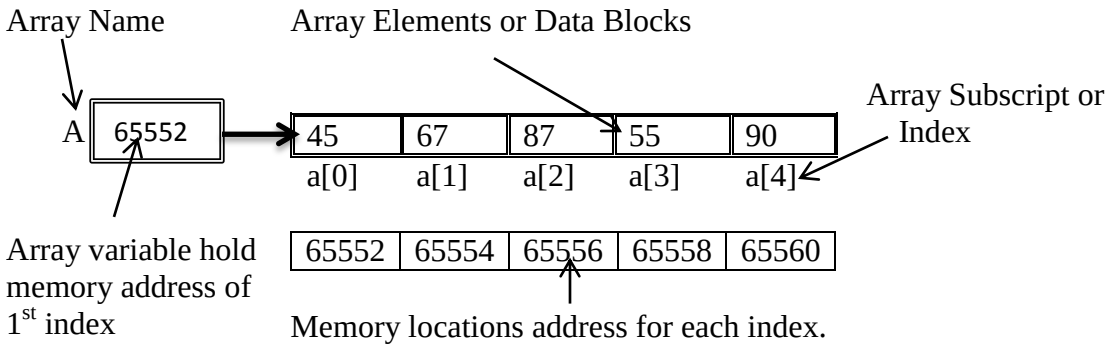


Fig 2:- Memory allocation for array variable.

- In array first element is indexed by "zero" and last indexed by "size-1". That is Lower Bound(LB) of array is zero & Upper Bound (UB) is size-1.

  Some example of arrays are:

  - int arr[10];
  - char name[20];
  - float height[60];
  - double salary[50];

2. Accessing One Dimensional Array Elements:

- One's array variables are declared or define, its elements can be accessed by using its index.
- Syntax
  arr_name[index];
  Example:-
  If we want to display or access number at position 4 from array 'a' (see fig2) .
  printf("%d",a[3]); → it will display 55.(see fig 2)
- To access array elements, specify the array name followed by index in square bracket eg a[3]. It will access number available at index 3.

3. Initialization of One Dimensional Array:

- We can initialize array elements when they are declared.
- Syntax:
  data_type arr_name[size]={value_list};
  In above syntax,
  We can also omit size of array & value_list is list of constant separated by comma.
- Example:
  i. int a[10]={3,6,8,9,12,5,9,7,18,25} ;
  ii. int a[ ]= {3,6,8,9,12,5,9,7,18,25} ;
  iii. int a[ ]={3,6,8,9,12};
  iv. int a[10]= {3,6,8,9,12,5} ;
  In example ii & iii we have omitted the size of array as it is initialized. And in example iv we have array size 10 & only six elements are initialized that's why remaining array elements will be zero.

4. Reading & Displaying (Processing) One Dimensional Array.

- We can use scanf() to read & printf() to display elements of array.
- As array is list of elements or numbers that's why we have multiple numbers to read and display then using loop is good practice to reduce code length & improve efficiency.
- We can use for loop.
- Suppose we have array
  int a[10]; then
- Reading Array Elements.

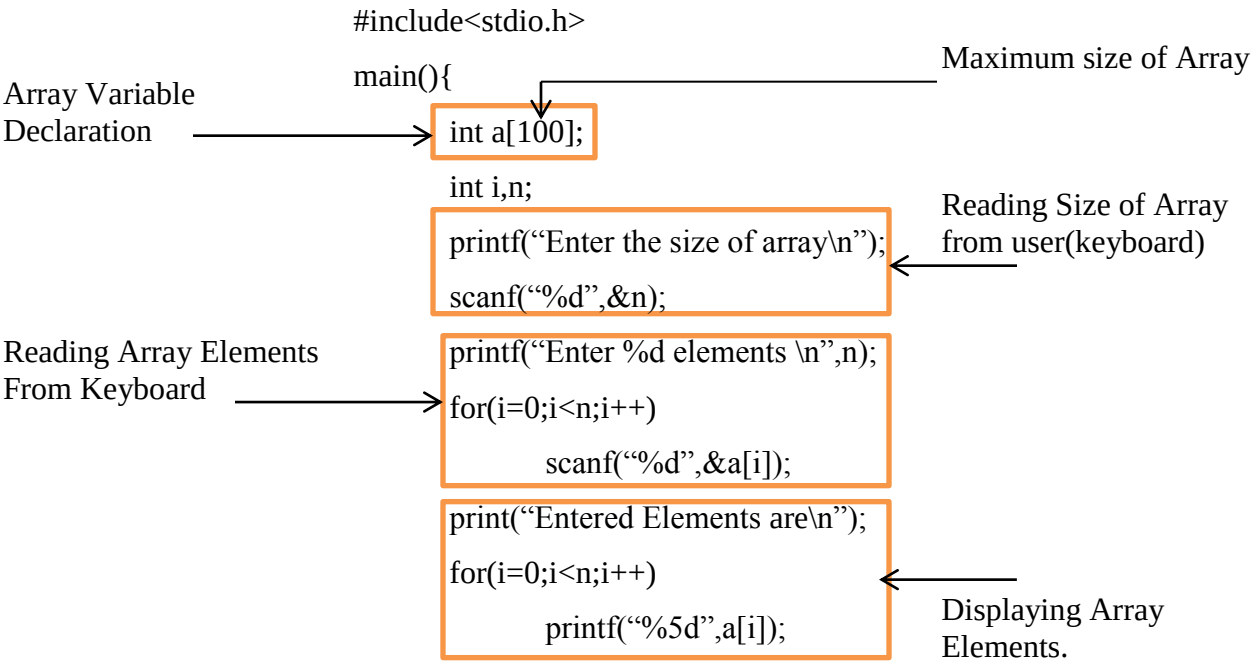
```
for(i=0;i<10;i++)
        scanf("%d",&a[i]);
```

- Displaying Array Elements.

```
for(i=0;i<10;i++)
        printf("%5d",a[i]);
```

Example:

/* Programming Example to read and Display Array elements */



Output:-

Enter the size of array

5

Enter 5 elements

5 8 7 9 3

Entered Elements are

5 8 7 9 3

Two Dimensional Array:-

- It is used to store data in matrix or tabular form..

1. Declaring Two Dimensional Array:

We must declare two dimensional array before use, and following is the syntax:

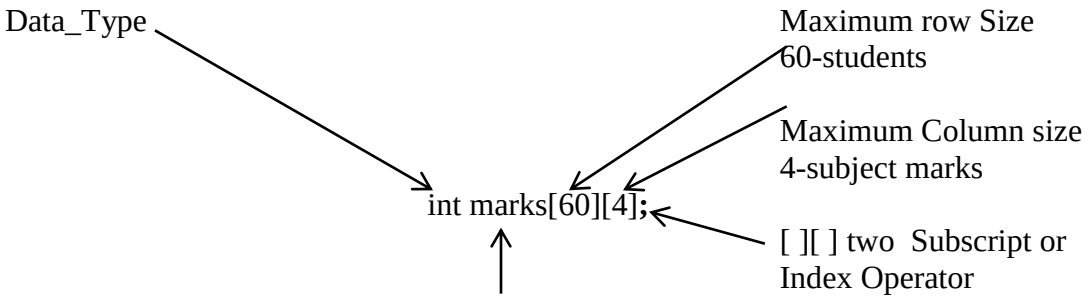
Syntax:-

```
data_type  arr_name[r_size][c_size];
```

In above syntax,

- data_type-can be'char','int','float'or'double'.
- arr_name-is similar to normal variables name.
- r_size- is the row size. & c_size is the column size.

Example:-



Name of 2-D Array is marks

Fig 3:- 2-DArray Declaration.

- In primary memory array will get contiguous block of memory as shown in following declaration of integer array "a" of size 2X3.

int a[2][3] ;

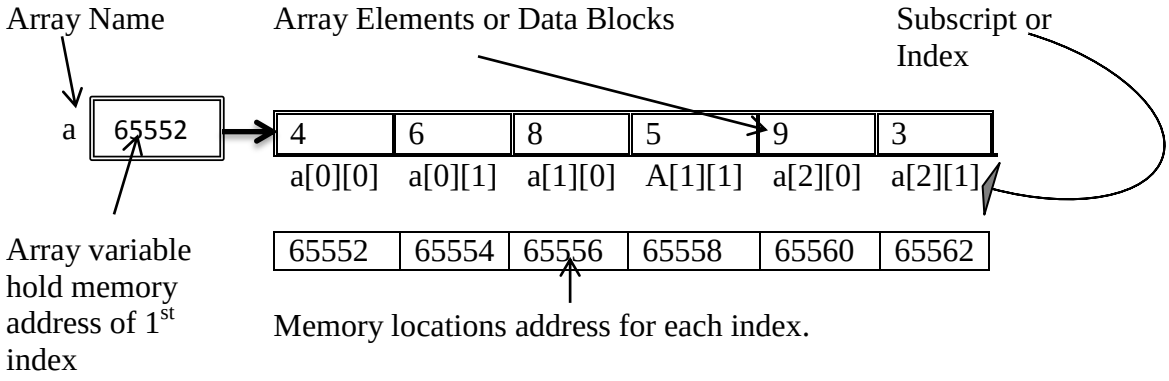


Fig 4:- Memory allocation for 2 D Array variable.

- Some example of two dimensional arrays are:

int arr[5][5];	char month [12][10];
char stud_name[60][20];	double salary[100][12];

2. Accessing Two Dimensional Array Elements:

- One's array variables are declared or define, its elements can be accessed by using its row & column index.
- Syntax
  arr_name[r_index][c_index];
  Example:-
  If we want to display or access number at position first row second column from array 'a' (see fig3) .
  printf("%d",a[0][1]); → it will display 6.(see fig 3)
- To access array elements, specify the array name followed by row index & column in separate square bracket eg a[0][1]. It will access number available at index [0][1].

3. Initialization of Two Dimensional Array:

- We can initialize array elements when they are declared.
- Syntax:
  data_type arr_name[r_size][c_size]={value_list};
  In above syntax,
  We can omit r_size of array but c_size is compulsory, value_list is list of constant separated by comma.
- Example:
  i. int a[3][3]={3,6,8,9,2,5,7,1,4} ;
  ii. int a[3][3]= {{3,6,8},{9,2,5},{7,1,4}}
  iii. int a[ ][3]={3,6,8,9,2,5,7,1,4};
  iv. int a[ ][3]= {{3,6,8},{9,2,5},{7,1,4}
  In example iii & iv we have omitted the r_size as it is initialized but column size is mandatory.
- For all above example 3X3 matrix will created & stored in memory, shown in following diagram.

3	6	8
a[0][0]	a[0][1]	a[0][2]
9	2	5
a[1][0]	a[1][1]	a[1][2]
7	1	4
a[2][0]	a[2][1]	a[2][2]

  v. int a[3][3]={3,6,8,9,2,5}
  Elements of above array will be as follows:

3	6	8
a[0][0]	a[0][1]	a[0][2]
9	2	5
a[1][0]	a[1][1]	a[1][2]
0	0	0
a[2][0]	a[2][1]	a[2][2]

vi. int a[ ][3]={3,6,8,9}

Elements of above array will be as follows:

3	6	8
a[0][0]	a[0][1]	a[0][2]
9	0	0
a[1][0]	a[1][1]	a[1][2]

vii. int a[3][3]={{3,9},{9,2,5},{7}}

Elements of above matrix will be as follows

3	9	0
a[0][0]	a[0][1]	a[0][2]
9	2	5
a[1][0]	a[1][1]	a[1][2]
7	0	0
a[2][0]	a[2][1]	a[2][2]

iv. Reading & Displaying (Processing) two Dimensional Array.

- We can use scanf() to read & printf() to display elements of array.
- As array is list of elements or numbers arranged in matrix or tabular format that's why we have multiple numbers to read and display then using nesting of loop is good practice to reduce code length & improve efficiency.
- We can use scanf() & printf() in nested for loop.
- Suppose we have array
  int a[3][3]; then
- Reading 2-D Array Elements.

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

- Displaying 2-D Array Elements.

```
for(i=0;i<3;i++){
        for(j=0;j<3;j++)
                printf("%5d",a[i][j]);
        printf("\n");
}
```

Example:→ Program to read and display matrix of order mXn.

```
#include<stdio.h>

main(){

int a[10][10];

int i,j,m,n;

printf("Enter row & column size\n");

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

printf("Enter  %d  elements",m*n);

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

        for(j=0;j<n;j++)

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

printf("Entered matrix is\n");

for(i=0;i<m;i++){

        for(j=0;j<n;j++)

            printf("%5d",a[i][j]);

  printf("\n");

}

}
```

2-D array Declaration
Maximum row &
column size is 10.

Reading row & column
size from user

Reading 2D Array
(Matrix)Elements

Displaying 2D array
elements in matrix form

Passing Array to Functions (Functions & Arrays)

- Arrays can be passed as a argument to the function by specifying its name without index.
- **Syntax:**

  fun_name(arr_name_without_index);
- **Example:**

  modify(a);
- **Explanation :-**

  Suppose we have function **modify()** & array **a[ ]** then we can pass array a[] as argument to function as follows.

```
main(){
        int a[ ]={10,20,30,40,50};
        modify(a);
        :
        :

}
```

- If we are passing an array variable as argument(actual parameter) then <u>the formal parameter of the called function may be one of the following.</u>

  i. **Sized Array.**

  Example:

```
 void modify(int a[5]){
 }
```

  ii. **Unsized Array**

  Example:

```
void modify(int a[ ]){
}
```

  iii. **Pointer**.

  Example:

```
void modify(int *a){
}
```

- Similarly, we may have function prototype or declaration as follows

  i. **With array name ie parameter name:**

  Example:-

```
void modify(int a[10]);
```

  ii. **Without array nam ie without parameter name:**

  Example:-

```
void modify(int [ ]);
```

Example:- Passing array as argument to function

```
/* Programming Example to double the elements of array*/
#include<stdio.h>
void modify(int [ ]);
main(){
        int i;
        int a[ ]={10,20,30,40,50};
        modify(a);
        printf("After modify array is");
        for(i=0;i<5;i++)
        printf("%5d",a[i]);
}
void modify(int a[5]){
        int i;
        for(i=0;i<5;i++)
        a[i]=a[i]*2;
}
```

Output:-

After modify array is 20 40 60 80 100

/* Write a program to find smallest element in an array of given 10 elements of integer using function and passing this array as argument to the function */

```
#include<stdio.h>
int small(int [ ]);
main(){
int a[ ]={26,45,67,89,21,94,5,78,81,91};
int i,min;
/* passing array as argument to function*/
min=small(a);
printf("Smallest number is %d",min);
}
```

```
int small(int a[10]){
int i,min;
min=a[0];
for(i=0;i<10;i++)
    if(min>a[i])
        min=a[i];
return min;
}
```

Output:-

Smallest number is 5.

/*Write a program to arrange the elements in an array of given numbers in ascending order using function and passing this array as argument to function. Given numbers are 54,36,83,46,94,47,58,91,11,44*/

```
#include<stdio.h>
void bubble(int [ ],int n);
main(){
int a[ ]={54,36,83,46,94,47,58,91,11,44};
int i,n=10;
/* passing array as argument to function*/
bubble(a,n);
printf("Numbers in ascending order are\n");
for(i=0;i<10;i++)
printf("%5d",a[i]);
}
int bubble(int a[10], int n){
int i,j;
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;
         }
      }
  }
}
```

Output:-

Numbers in ascending order are

11 36 44 46 47 45 54 58 83 94

Storage Classes

- In C storage classes are used to indicate following things.
    1. Where variable should be stored [**Storage**].
    2. How long variable will exists [**Life Time**]
    3. What will be their area of existence [**Scope**]
    4. What will be default value [**Default Initial Value**]
- At the time of program execution, Storing variable for some period of time in memory is called Extent of variable.
- Extent of variable is categorized under four storage classes.
- C provides four storage classes & can be used along with data types specifiers in variable declaration statement as follows:
    i. auto
    ii. static
    iii. register
    iv. extern
- Storage class specifier precedes the declaration statement of variable.
- **Syntax:**

    storage_class data_type variable_name;
- **Example:**

    auto int i=0;
    static int i;
    register int i=0;
    extern int i;