CHAPTER 4
Arrays & Structures

I. Arrays:-

a) Introduction

- **What is an array?**

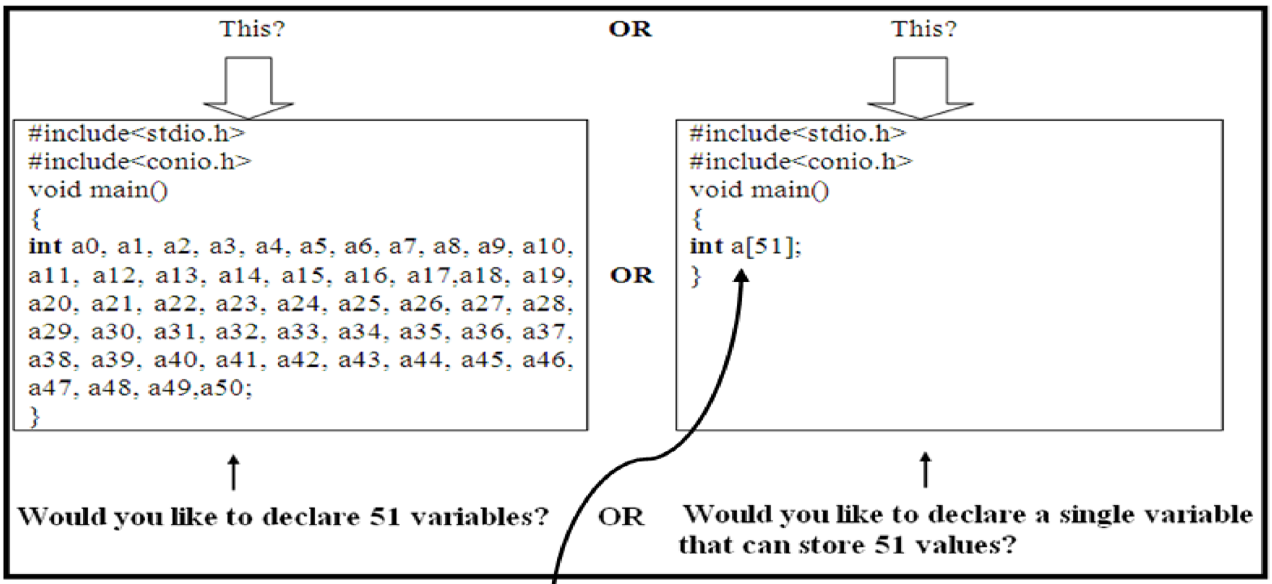
"An array is a collection of elements of same data type."
OR
"An array is a variable that can store multiple elements of same data type."

- For understanding array properly, let's consider following program.

1.	

#include<stdio.h>
#include<conio.h>
void main()
{
int a;
clrscr();
a=5;
a=10;
printf("%d",a);
getch();
}
	OUTPUT
	10
	
	This program will print 10 on the screen. Why 10? Why not 5? Because when we assign value 10 to variable 'a' the previous value (5) hold by variable 'a' is replaced by value 10.
	
	What is conclusion?
	Variable **a** can hold only one value at a time.

Now suppose if you want to store marks of 51 students, then would you like to declare 51 variables or would you like to declare a single variable that can store 51 values.



This? **OR** This?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
a11, a12, a13, a14, a15, a16, a17,a18, a19,
a20, a21, a22, a23, a24, a25, a26, a27, a28,
a29, a30, a31, a32, a33, a34, a35, a36, a37,
a38, a39, a40, a41, a42, a43, a44, a45, a46,
a47, a48, a49,a50;
}
```

OR

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[51];
}
```

Would you like to declare 51 variables? OR Would you like to declare a single variable that can store 51 values?

Obliviously you would like this.
(Here 'a' is called as an array that can store 51 elements of type int.)

b) Array Fundamentals (1 dimensional OR 1D Array)

➢ A simple program using arrays

2.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int age[4];
int i;
clrscr();

  for(i=0;i<4;i++)
    {
     printf("\nEnter an age: ");
     scanf("%d",&age[i]);
    }

  for(i=0;i<4;i++)
    {
     printf("\n\nYour entered: %d",age[i]);
    }

getch();
}
```

OUTPUT

```
Enter an age: 44

Enter an age: 16

Enter an age: 23

Enter an age: 68


Your entered: 44

Your entered: 16

Your entered: 23

Your entered: 68
```

In the above program first for loop gets the value from the user and places them in the array **age**, while the second for loop reads them from the array and displays them.

➢ Array Declaration

Like other variables in C, an array must be declared before it can be used to store data items (elements). Like other variable declaration, array declaration specifies array type and a name. But it includes another feature: a size. The size specifies how many elements the array will contain. The size is surrounded by square brackets. Following figure shows the syntax of an array declaration.

In the above program the array declaration **int age[4];** specifies that the name of array is **age** that can store **4** elements of type **int**. That means array **age** can store **4** integer elements.

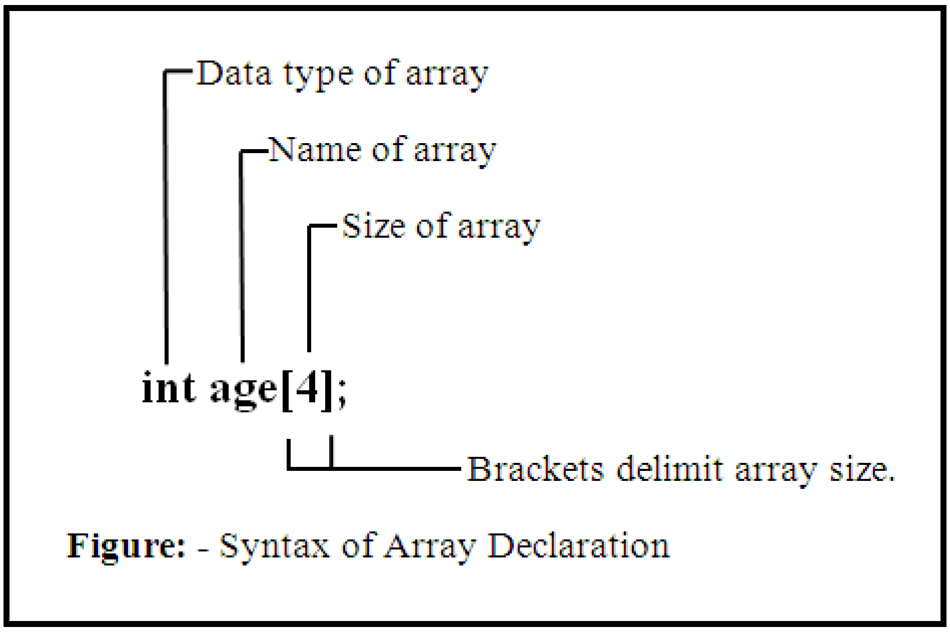


Figure: - Syntax of Array Declaration

➢ Array Elements

When compiler see this **int age[4];** declaration, immediately it will reserve 8 (4x2=8) bytes of memory for array age. Why 8 bytes? Because each element in the **age** is of type int, and the size of age is 4. And we know that int requires 2 bytes.

Following figure2 shows the elements of the array **age** in memory.

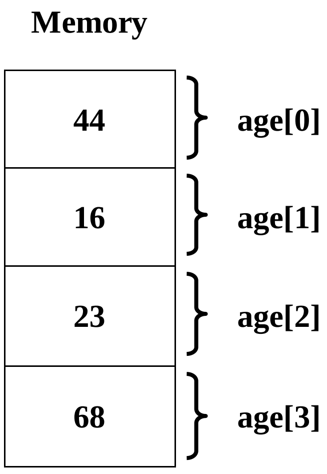


Figure2:- Array Elements in Memory

Notice that first element in the array age is age[0], second element is age[1], third element is age[2] and fourth element is age[3]. And this constant in the brackets is called as an array index. That means, in array all the elements are numbered, starting with 0.

➢ Accessing Array Elements

You can access elements in the array by number; this number is called as an array index.

Example, printf("%d", age[0]); this statement will print 44.

printf("%d", age[1]); this statement will print 16.

printf("%d", age[2]); this statement will print 23.

printf("%d", age[3]); this statement will print 68.

Above four statements show that you can access elements of age using index.

o *Entering Data into an Array*

(See above program2)

You can use for loop for entering data into array age as shown below,

```
for(i=0;i<4;i++)
  {
   printf("\nEnter an age: ");
   scanf("%d",&age[i]);
  }
```

Notice that **i=0;** and **i<4;** The for loop will start at i=0 and scanf function will assign value entered by the user to 1st element age[0], then control goes to increment section, now the value of i becomes 1. So the condition i<4 is true, therefore scanf function will assign value entered by the user to 2nd element age[1]. And this process is repeated until condition becomes false. That means when i becomes 4 loop will be terminated and loop body will not be executed.

You can enter data into array without using for loop also, but you need to write more statements.

```
printf("\nEnter an age: ");
scanf("%d",&age[0]);
printf("\nEnter an age: ");
scanf("%d",&age[1]);
printf("\nEnter an age: ");
scanf("%d",&age[2]);
printf("\nEnter an age: ");
scanf("%d",&age[3]);
```

Reading and Printing Data from an Array

(See above program2)

You can read and print data from the array using for loop, as shown below,

```
for(i=0;i<4;i++)
  {
   printf("\n\nYour entered: %d",age[i]);
  }
```

You can read and print data from the array without using for loop but you need more statements, as shown below,

```
printf("\n\nYour entered: %d",age[0]);
printf("\n\nYour entered: %d",age[1]);
printf("\n\nYour entered: %d",age[2]);
printf("\n\nYour entered: %d",age[3]);
```

Array Initialization

You can give values to each element of array when the array is declared. And this is called as array initialization.

For example,

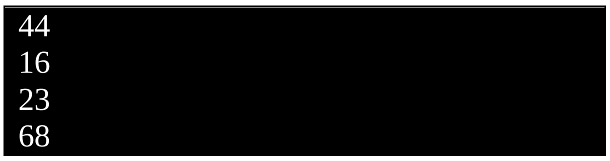
3.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int age[4]={44,16,23,68};
int i;
clrscr();


  for(i=0;i<4;i++)
    {
    printf("\n\n%d",age[i]);
    }

getch();
}
```

OUTPUT



```
44
16
23
68
```

In the above program3 the values to which age is initialized are surrounded by braces and separated by commas. They are connected to array expression by equal sign. The syntax of array initialization is shown in figure3 below,

Actually, we don't need to use the array size when we initialize all the array elements. Therefore size is optional during array initialization, as shown in the program4 below. See the syntax for array initialization shown in figure3 below.

4.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int age[ ]={44,16,23,68};
int i;
clrscr();


   for(i=0;i<4;i++)
     {
     printf("\n\n%d",age[i]);
     }

getch();
}
```

OUTPUT

```
44
16
23
68
```

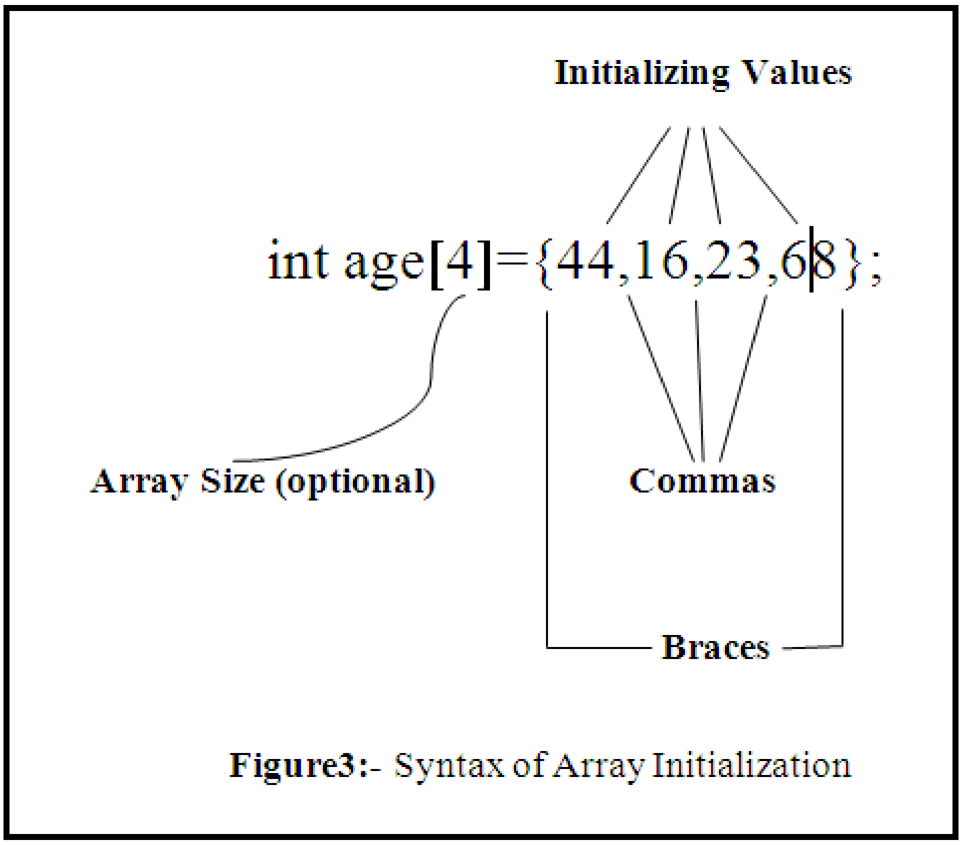


Figure3:- Syntax of Array Initialization

5. Write a program to find smallest and largest number from an array?

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

for(i=0;i<5;i++)
  {
   printf("\nEnter the number:");
   scanf("%d",&a[i]);
  }

small=a[0];
large=a[0];

for(i=0;i<5;i++)
  {
   if(a[i]<small)
       small=a[i];
   if(a[i]>large)
      large=a[i];
}
printf("\nThe smallest number in array is %d",small);
printf("\n\nThe largest  number in array is %d",large);

getch();
}
```

OUTPUT

```
Enter the number:13

Enter the number:15

Enter the number:99

Enter the number:10

Enter the number:5

The smallest number in array is 5

The largest  number in array is 99
```

6. Write a program to sort array elements in ascending order? Take array elements from the user?

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

for(i=0;i<5;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }

for(i=0;i<5;i++)
  {
   for(j=i+1;j<5;j++)
           {
            if(a[i]>a[j])
             {
                  temp=a[i];
                  a[i]=a[j];
                  a[j]=temp;
             }
           }
   }
printf("\nArray Elements in Ascending Order: ");
for(i=0;i<5;i++)
  {
   printf("\n%d",a[i]);
  }
getch();
}
```

OUTPUT

```

Enter number: 5

Enter number: 4

Enter number: 3

Enter number: 2

Enter number: 1

Array Elements in Ascending Order:
1
2
3
4
5
```

7. Write a program to sort array elements in descending order?

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

for(i=0;i<5;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }

for(i=0;i<5;i++)
  {
   for(j=i+1;j<5;j++)
         {
          if(a[i]<a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
         }
    }
printf("\nArray Elements in Descending Order: ");
for(i=0;i<5;i++)
  {
   printf("\n%d",a[i]);
  }
getch();
}
```

OUTPUT

```

Enter number: 1

Enter number: 2

Enter number: 3

Enter number: 4

Enter number: 5

Array Elements in Descending Order:
5
4
3
2
1
```

8. Write a program to sort an array of 10 elements entered by user in descending order?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10];
int b[10];
int i,j,temp;
clrscr();

for(i=0;i<10;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }
for(i=0;i<10;i++)
  {
   for(j=i+1;j<10;j++)
          {
           if(a[i]<a[j])
            {
                  temp=a[i];
                  a[i]=a[j];
                  a[j]=temp;
            }
          }
  }
printf("\nArray Elements in Descending Order: ");
for(i=0;i<10;i++)
  {
   printf("\n%d",a[i]);
  }
getch();
}
```

OUTPUT

```

Enter number: 1

Enter number: 2

Enter number: 3

Enter number: 4

Enter number: 5

Enter number: 6

Enter number: 7

Enter number: 8

Enter number: 9

Enter number: 1111

Array Elements in Descending Order:
1111
9
8
7
6
5
4
3
2
1
```

9. Any number is entered from keyboard. WAP to check whether that no. is present in the array of 10 elements. If it is display its position?

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10];
int i,num,flag=0;
clrscr();

for(i=0;i<10;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }

printf("\n\nEnter no. to check whether it is present in the array: ");
scanf("%d",&num);

for(i=0;i<10;i++)
  {
            if(num==a[i])
            {
              flag=1;
              break;
            }
  }

  if(flag==1)
  printf("\nNumber is present in the array at position %d",i);
  else
  printf("\nNumber is not present");
getch();
}
```

OUTPUT

```
Enter number: 1

Enter number: 2

Enter number: 3

Enter number: 4

Enter number: 5

Enter number: 6

Enter number: 7

Enter number: 8

Enter number: 9

Enter number: 12


Enter no. to check whether it is present in the array: 1

Number is present in the array at position 0
```

10. Any number is entered from keyboard. WAP to check whether that no. is present in the array of 10 elements. If it is display how many times it appears in the array?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10];
int i,num,count=0;
clrscr();

for(i=0;i<10;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }
printf("\n\nEnter number to check: ");
scanf("%d",&num);

for(i=0;i<10;i++)
  {
         if(num==a[i])
         {
           count++;
         }
  }

  printf("\nNumber appears %d times",count);
  getch();
}
```

OUTPUT

```

Enter number: 1

Enter number: 2

Enter number: 3

Enter number: 4

Enter number: 55

Enter number: 6

Enter number: 55

Enter number: 7

Enter number: 55

Enter number: 8


Enter number to check: 55

Number appears 3 times
```

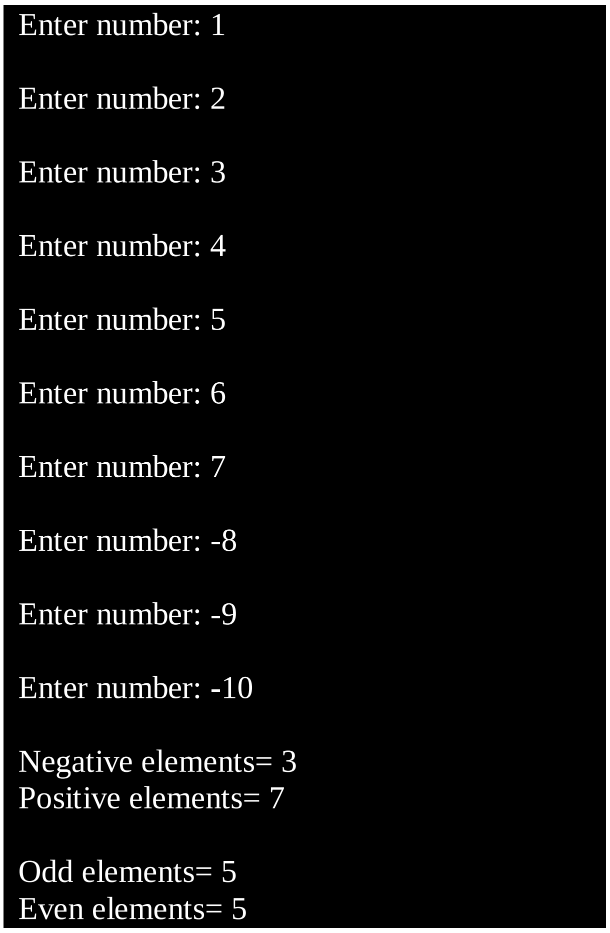
11. Ten numbers are entered from the keyboard into an array. Write a program to find out how many of them are positive, how many are negative, how many are odd and how many are even?

```
#include<stdio.h>
void main()
{
int a[10];
int b[10];
int i,pos=0,neg=0,eve=0,odd=0;

for(i=0;i<10;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }
for(i=0;i<10;i++)
  {
     if(a[i]>0)
            pos++;
     else neg++;

     if(a[i]%2==0)
            eve++;
     else odd++;
  }
printf("\nNegative elements=%d",neg);
printf("\nPositive elements=%d",pos);
printf("\n\nOdd elements=%d",odd);
printf("\nEven elements=%d",eve);
}
```

OUTPUT



```
Enter number: 1

Enter number: 2

Enter number: 3

Enter number: 4

Enter number: 5

Enter number: 6

Enter number: 7

Enter number: -8

Enter number: -9

Enter number: -10

Negative elements= 3
Positive elements= 7

Odd elements= 5
Even elements= 5
```

12. Write a program to calculate average marks obtained by a class of 30 students?

```
#include<stdio.h>
void main()
{
int marks[30];
int i,sum=0,avg;

for(i=0;i<30;i++)
  {
    printf("\nEnter marks: ");
    scanf("%d",&marks[i]);
   }

for(i=0;i<30;i++)
   {
    sum=sum+marks[i];
   }
avg=sum/30;
  printf("\n\nAverage marks= %d",avg);
  }
```

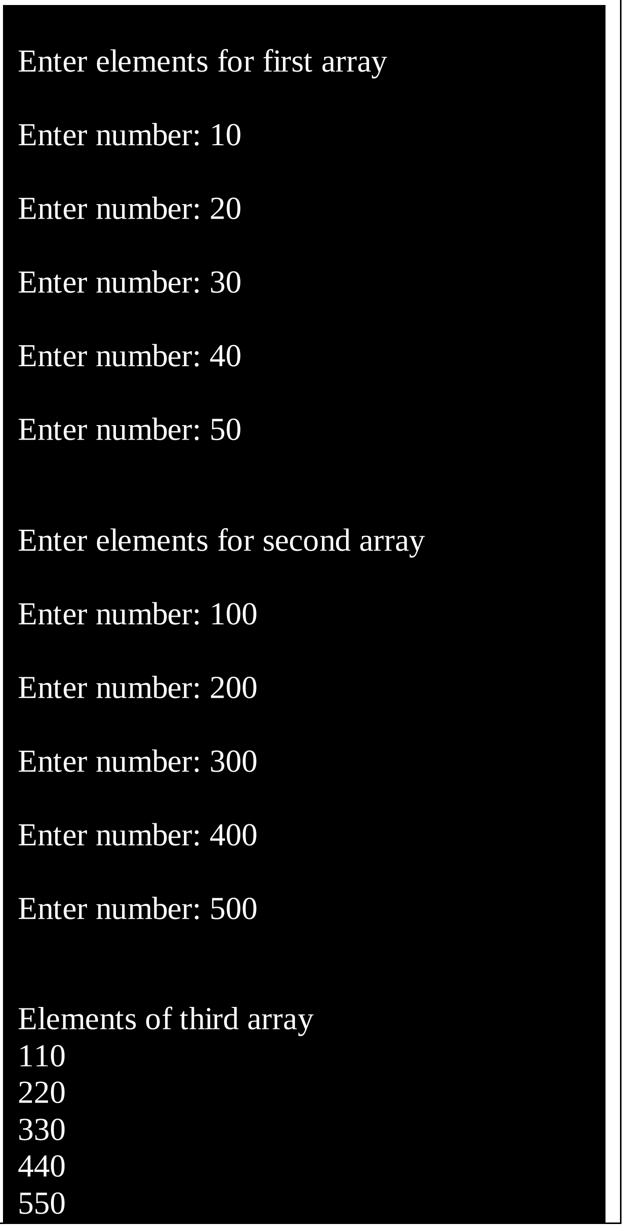
13. Write a program to add two 1D array and store into third array?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],b[5],c[5];
int i;
clrscr();
printf("\nEnter elements for first array\n");
for(i=0;i<5;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }
printf("\n\nEnter elements for second array\n");
for(i=0;i<5;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&b[i]);
  }

for(i=0;i<5;i++)
  {
   c[i]=a[i]+b[i];
  }

printf("\n\nElements of third array");
for(i=0;i<5;i++)
  {
   printf("\n%d",c[i]);
  }
getch();
}
```

OUTPUT



```

Enter elements for first array

Enter number: 10

Enter number: 20

Enter number: 30

Enter number: 40

Enter number: 50


Enter elements for second array

Enter number: 100

Enter number: 200

Enter number: 300

Enter number: 400

Enter number: 500


Elements of third array
110
220
330
440
550
```

14. Write a program to copy contents of one array into another in the reverse order?

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

for(i=0;i<5;i++)
  {
   printf("\nEnter number: ");
   scanf("%d",&a[i]);
  }
printf("\nElements of array a:");
for(i=0;i<5;i++)
  {
   printf("\n%d",a[i]);
  }

for(i=0;i<5;i++)
  {
   b[4-i]=a[i];
     }
printf("\n\nElements of array b:");
for(i=0;i<5;i++)
  {
   printf("\n%d",b[i]);
  }
getch();
}
```

OUTPUT

```

Enter number: 9

Enter number: 69

Enter number: 332

Enter number: 25

Enter number: 55

Elements of array a:
9
69
332
25
55

Elements of array b:
55
25
332
69
9
```

c) Two Dimensional (2D) Arrays:-

So far we have seen arrays of one dimension. Arrays can have two or more dimensions. Array having two dimensions is called as two dimensional array or 2D array. The two dimensional array is also called as a **matrix**.

A 2D array has two size specifiers, each enclosed in brackets. First specifier denotes number of rows and second specifier denotes number of columns.

For example,

1) Consider, **stud[4][2]** array. Here stud is a 2D array, having 4 rows and 2 columns.
2) Consider, **score[2][5]** array. Here score is a 2D array, having 2 rows and 5 columns.
3) Consider, **a[2][2]** array. Here **a** is a 2D array, having 2 rows and 2 columns.
4) Consider, **b[3][3]** array. Here **b** is a 2D array, having 3 rows and 3 columns.
5) Consider, c**[4][4]** array. Here **c** is a 2D array, having 4 rows and 4 columns.

➢ Declaration of 2D array

The declaration of two dimension arrays is as follows:

data_type array_name[rows][columns];

A 2D array has two size specifiers, each enclosed in brackets. First specifier denotes number of rows and second specifier denotes number of columns in the array.

For example, **int stud[4][2];** is a 2D array having 4 rows and 2 columns.

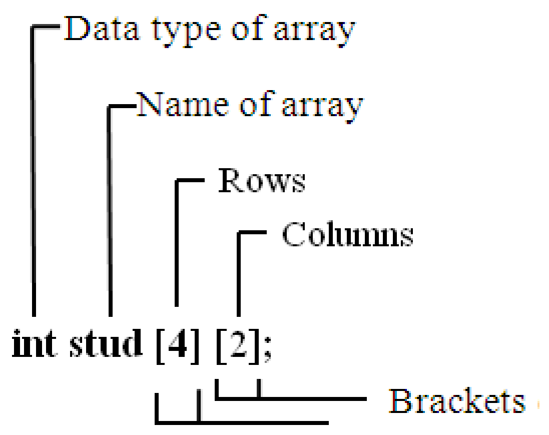


Figure4:- Syntax of 2D Array Declaration

➢ Initialization of 2D Arrays:-

We can initialize 2D array as shown below,

```
int stud[4][2]={
                        {1234,56},
                        {1212,33},
                        {1434,80},
                        {1312,78}
                    };
```

OR even this will work,

```
int stud[4][2]={1234,56,1212,33,1434,80,1312,78};
```

Following figure5 shows arrangement of **int stud[4][2];**

Array stud is a 2D array having 4 rows (numbered from 0 to3) and 2 columns (numbered from 0 to 1).

Thus 1234 is stored in stud[0][0], 56 is stored in stud[0][1], 1212 is stored in stud[1][0] and so on…..and last element 78 is stored in stud[3][1].

	column no. 0	**column no. 1**
Row no. 0	**1234**	**56**
Row no. 1	**1212**	**33**
Row no. 2	**1434**	**80**
Row no. 3	**1312**	**78**

Figure5:- Arrangement of stud[4][2] array.

stud[0][0]	**stud[0][1]**	**stud[1][0]**	**stud[1][1]**	**stud[2][0]**	**stud[2][1]**	**stud[3][0]**	**stud[3][1]**
1234	**56**	**1212**	**33**	**1434**	**80**	**1312**	**78**

Figure6:- Elements of stud [4] [2] are stored in contiguous memory locations.

➢ Accessing 2D Array Elements

You can access elements in the 2D array by subscripts. The first subscript represents row number and second subscript represents column number.

Example, printf("%d", **stud[3][0]);** this statement will print 1312.

printf("%d", **stud[3][1]);** this statement will print 78.

printf("%d", **stud[0][1]);** this statement will print 56..

o *Reading and Printing Data from an Array*

(See program 15 or 16 given below)

You can read and print data from the array using nested for loop, outer for loop is used to count number of rows and inner for loop is used to count number of columns, as shown below,

```
for(i=0;i<4;i++)
  {
   for(j=0;j<2;j++)
     {
       printf("%d ",stud[i][j]);
     }
   printf("\n");
  }
```

o *Entering Data into an Array*

(See program17 given below)

You can use nested for loop for entering data into 2D array stud, outer for loop is used to count number of rows and inner for loop is used to count number of columns, as shown below,

```
for(i=0;i<4;i++)
  {
    printf("Enter roll number then marks:");
    for(j=0;j<2;j++)
     {
       scanf("%d",&stud[i][j]);
     }
    printf("\n");
  }
```

15. Write a program to store roll number and marks obtained by a student side by side in a matrix?

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int stud[4][2]={{1234,56},{1212,33},{1434,80},{1312,78}};
int i,j;
clrscr();

printf("\nArray Elements:\n");

for(i=0;i<4;i++)
 {
   for(j=0;j<2;j++)
      {
        printf("%d ",stud[i][j]);
      }
   printf("\n");
  }

getch();
}
```

OUTPUT

```

Array Elements:
1234 56
1212 33
1434 80
1312 78
```

16. Write a program to store roll number and marks obtained by a student side by side in a matrix?

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int stud[4][2]={{1234,56},{1212,33},{1434,80},{1312,78}};
int i,j;
clrscr();
printf("\nArray Elements:\n");

for(i=0;i<4;i++)
 {
   for(j=0;j<2;j++)
      {
           printf("%d ",stud[i][j]);
      }
  }

getch();
}
```

OUTPUT

```

Array Elements:
1234 56 1212 33 1434 80 1312 78
```

17. Write a program to store roll number and marks obtained by a student side by side in a matrix? Get the values from the user.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int stud[4][2];
int i,j;
clrscr();

for(i=0;i<4;i++)
  {
    printf("Enter roll number then marks:");
    for(j=0;j<2;j++)
     {
       scanf("%d",&stud[i][j]);
     }
    printf("\n");
  }

printf("\nArray Elements:\n");

for(i=0;i<4;i++)
  {
   for(j=0;j<2;j++)
     {
       printf("%d ",stud[i][j]);
     }
   printf("\n");
  }

getch();
}
```

OUTPUT

```
Enter roll number then marks:1234 56

Enter roll number then marks:1212 33

Enter roll number then marks:1434 80

Enter roll number then marks:1312 78


Array Elements:
1234 56
1212 33
1434 80
1312 78
```

18. Write a program to display contents of 2X2 matrix? (2D array having 2 rows 2 columns)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][2]={
                   {1,2},
                   {3,4}
                 };
int i,j;
clrscr();

printf("\nArray Elements:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     printf("%d ",a[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Array Elements:
1 2
3 4
```

19. Write a program to display contents of 2X3 matrix? (i.e. 2D array having 2 rows and 3 columns)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][3]={
                  {1,2,3},
                  {4,5,6}
                };
int i,j;
clrscr();

printf("\nArray Elements:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<3;j++)
    {
     printf("%d ",a[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Array Elements:
1 2 3
4 5 6
```

20. Write a program to display contents of 3X2 matrix? (2D array having 3 rows and 2 columns)

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][2]={
                {1,2},
                {3,4},
                {5,6}
              };
int i,j;
clrscr();

printf("\nArray Elements:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<2;j++)
    {
     printf("%d ",a[i][j]);
    }
    printf("\n");
  }
getch();
}
```

OUTPUT

```

Array Elements:
1 2
3 4
5 6
```

21. Write a program to display contents of 3X3 matrix? (2D array having 3 rows 3 columns)

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3]={
                {1,2,3},
                {4,5,6},
                {7,8,9}
              };
int i,j;
clrscr();

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

OUTPUT

```
Array Elements:
1 2 3
4 5 6
7 8 9
```

22. Write a program for addition of two 3X3 matrices?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],b[3][3],c[3][3];
int i,j;
clrscr();
printf("\nEnter elements of first array:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }
printf("\nEnter elements of second array:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     scanf("%d",&b[i][j]);
    }
  }

for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
    c[i][j]=a[i][j]+b[i][j];
    }
  }

printf("\nAddition of two matrices:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     printf("%d ",c[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Enter elements of first array:
10 20 30
40 50 60
70 80 90

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

Addition of two matrices:
11 22 33
44 55 66
77 88 99
```

23. Write a program for subtractions of two 3X3 matrices?

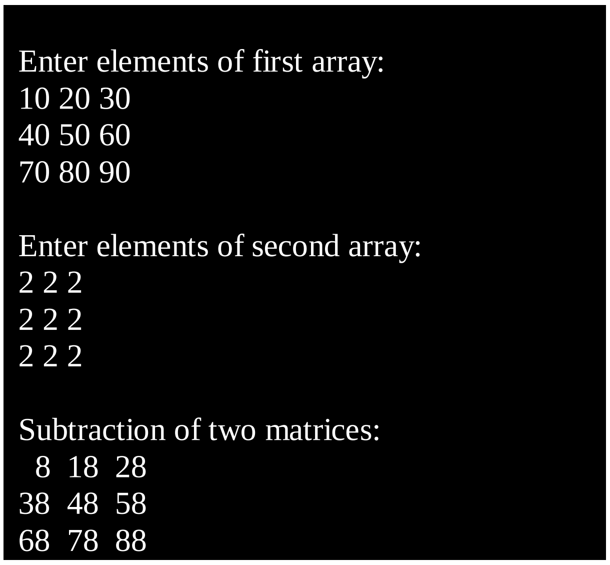
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],b[3][3],c[3][3];
int i,j;
clrscr();
printf("\nEnter elements of first array:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }
printf("\nEnter elements of second array:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     scanf("%d",&b[i][j]);
    }
  }

for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
    c[i][j]=a[i][j]-b[i][j];
    }
  }

printf("\nSubtraction of two matrices:\n");
for(i=0;i<3;i++)
  {
    for(j=0;j<3;j++)
    {
     printf("%d ",c[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT



```

Enter elements of first array:
10 20 30
40 50 60
70 80 90

Enter elements of second array:
2 2 2
2 2 2
2 2 2

Subtraction of two matrices:
  8  18  28
38  48  58
68  78  88
```

24. Write a program for multiplication of two 2X2 matrices?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][2],b[2][2],c[2][2];
int i,j;
clrscr();
printf("\nEnter elements of first array:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }
printf("\nEnter elements of second array:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     scanf("%d",&b[i][j]);
    }
  }

for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
    c[i][j]=a[i][0]*b[0][j]+a[i][1]*b[1][j];
    }
  }

printf("\Multiplication of two matrices:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     printf("%d ",c[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Enter elements of first array:
1 2
3 4

Enter elements of second array:
1 2
3 4

Multiplication of two matrices:
7   10
15 22
```

25. Write a program for transpose of 2x2 matrix?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[2][2],b[2][2];
int i,j;
clrscr();
printf("\nEnter elements 2x2 matrix:\n");
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }

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

printf("\nTranspose of matrix is\n");

for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
    {
     printf("%d ",a[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Enter elements of 2x2 matrix:
10 20
30 40

Transpose of matrix is
10 30
20 40
```

26. Write a program to obtain transpose of 4x4 matrix?

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[4][4],b[4][4];
int i,j;
clrscr();
printf("\nEnter elements of 4X4 Matrix:\n");
for(i=0;i<4;i++)
  {
    for(j=0;j<4;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }

for(i=0;i<4;i++)
  {
    for(j=0;j<4;j++)
    {
    b[i][j]=a[i][j];
    }
  }
for(i=0;i<4;i++)
  {
    for(j=0;j<4;j++)
    {
    a[i][j]=b[j][i];
    }
  }

printf("\nTranspose of matrix is\n");

for(i=0;i<4;i++)
  {
    for(j=0;j<4;j++)
    {
     printf("%d ",a[i][j]);
    }
    printf("\n");
  }

getch();
}
```

OUTPUT

```

Enter elements of 4X4 Matrix:
10 20 30 40
50 60 70 80
11 12 13 14
15 16 17 18

Transpose of matrix is
10 50 11 15
20 60 12 16
30 70 13 17
40 80 14 18
```

27. Write a program to pick up the smallest and largest number from any 5 row by 5 column matrix?

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5][5];
int i,j,small=0,large=0;
clrscr();
printf("\nEnter elements of 5X5 Matrix:\n");
for(i=0;i<5;i++)
  {
    for(j=0;j<5;j++)
    {
     scanf("%d",&a[i][j]);
    }
  }
small=a[0][0];
large=a[0][0];
for(i=0;i<5;i++)
  {
    for(j=0;j<5;j++)
    {
      if(a[i][j]<small)
         small=a[i][j];
      if(a[i][j]>large)
         large=a[i][j];
    }
  }
printf("\nSmallest element is %d\n",small);
printf("\nLargest element is %d\n",large);
getch();
}
```

OUTPUT

```

Enter elements of 5X5 Matrix:
10 20 30 40 50
60 70 80 90 100
11 12 34 67 89
1 -99 55 67 666
-9 98 98 98 -999

Smallest element is -999

Largest element is 666
```

d) Passing 1D Array to a Function

Array elements can be passed to a function by calling the function by value or by reference. In the call by value, we pass values of array elements to the function, whereas in the call by reference, we pass address of array elements to the function. See examples 28 and 29 below,

28. Write a program to demonstrate how to pass array elements to function using call by value?

```
/*Passing Array Elements to
Function using Call By Value*/
#include<stdio.h>
#include<conio.h>
void display(int);
void main()
{
int a[5]={10,20,30,40,50};
int i;
clrscr();
for(i=0;i<5;i++)
  {
    display(a[i]);
  }
getch();
}
void display(int n)
{
printf("\n%d",n);
}
```

OUTPUT

```
10
20
30
40
50
```

29. Write a program to demonstrate how to pass array elements to function using call by reference?

```
/*Passing Array Elements to
Function using Call By Reference*/
#include<stdio.h>
#include<conio.h>
void display(int*);
void main()
{
int a[5]={10,20,30,40,50};
int i;
clrscr();
for(i=0;i<5;i++)
  {
    display(&a[i]);
  }
getch();
}
void display(int *n)
{
printf("\n%d",*n);
}
```

OUTPUT

```
10
20
30
40
50
```

30. Write a program to demonstrate how to pass an entire array to function?

```
/*Passing An Entire Array to Function*/
#include<stdio.h>
#include<conio.h>
void display(int *b);
void main()
{
int a[5]={10,20,30,40,50};
clrscr();
display(a);    /*display(&a[0])*/

/*Above both function calls are same*/
getch();
}

void display(int *b)
{
int i;
 for(i=0;i<5;i++)
 {
   printf("\n%d",*b);
   b++;     /*increment pointer to point to next element */
 }
}
```

OUTPUT

```
10
20
30
40
50
```

31. Write a program to demonstrate how to pass an entire array to function?

```
/*Passing An Entire Array to Function*/
#include<stdio.h>
#include<conio.h>
void display(int b[5]);
void main()
{
int a[5]={10,20,30,40,50};
clrscr();

    display(a);

getch();
}

void display(int b[5])
{
int i;

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

OUTPUT

```
10
20
30
40
50
```

e) Passing 2D Array to a Function

32. Write a program to demonstrate how pass to an entire 2D array to function?

```
#include<stdio.h>
#include<conio.h>
void display(int b[2][2]);

void main()
{
int a[2][2]={
                    {10,20},
                    {30,40}
                   };
clrscr();

    display(a);

getch();
}

void display(int b[2][2])
{
int i,j;

 for(i=0;i<2;i++)
 {
  for(j=0;j<2;j++)
   printf("%d ",b[i][j]);
   printf("\n");
 }

}
```

OUTPUT

```
10  20
30  40
```

33. Write a program to demonstrate how to pass an entire 2D array to function?

```
#include<stdio.h>
#include<conio.h>
void display(int b[3][2]);
void main()
{
int a[3][2]={
                    {10,20},
                    {30,40},
                    {50,60}
                  };
clrscr();

    display(a);

getch();
}

void display(int b[3][2])
{
int i,j;

 for(i=0;i<3;i++)
 {
  for(j=0;j<2;j++)
   printf("%d ",b[i][j]);
   printf("\n");
 }

}
```

OUTPUT

```
10  20
30  40
50  60
```

II. Structures:-

A structure is a collection of simple variables. The variables in a structure can be of different type: some can be int, some can be float, some can be char and so on. The data items in a structure are called structure members or structure elements.

OR

A structure contains a number of data types grouped together. These data types may or may not be of the same type. The data items in a structure are called structure members.

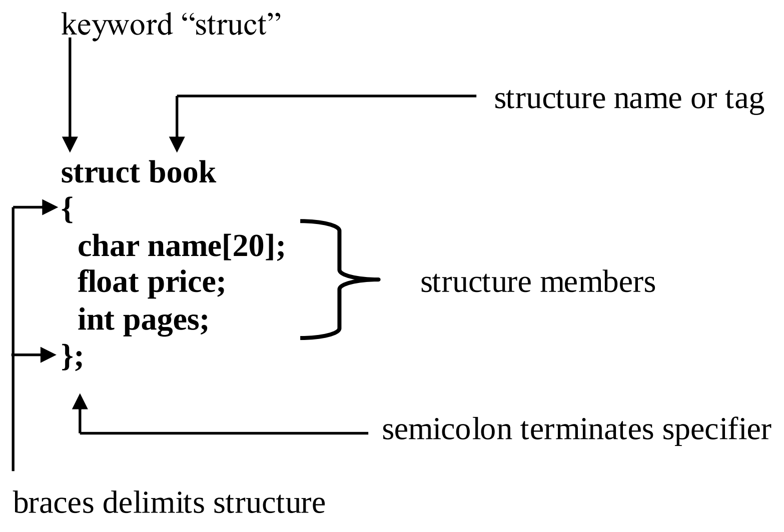
➢ **Specifying (or declaring) the structure**

The structure specifier tells what members the structure will have.

For example, suppose you want to store data about a book. Then you might want to store name of book, its price and number of pages.

- Name of book is a char type data(i.e. string),
- price is a float type data and
- number of pages is an int type data.

Then structure specifier to store data about a book could be,
(See program34 below given on page no.35)



```
struct book
{
  char name[20];
  float price;
  int pages;
};
```

Fig:- syntax of structure specifier

The structure specifier starts with the keyword **struct.** Next come structure name or tag. Then structure members are declared within braces. Finally a semicolon follows the closing brace, it terminates the structure specifier.

The structure specifier is used as blueprint for creating variables of type book. The specifier does not set aside any space in memory.

- **Defining a structure variable:-**

Defining a structure variable is same as that for defining a variable of built-in data type.
In the program34 the first statement in main ()

- **struct book b1;**

defines variable called b1. Now, this definition reserve space in memory for variable b1.
How much space?

- 20 bytes for array name
- 4 bytes for price variable and
- 2 bytes for pages variable

- **Accessing structure members:-**

You can access structure members through a structure variable using a dot (**.**) operator.
See program34, where we have accessed three structure members of **b1** using dot (**.**) operator.

- **b1.name**
- **b1.price**
- **b1.pages**

Figure below that shows how structure variable b1 looks in memory.

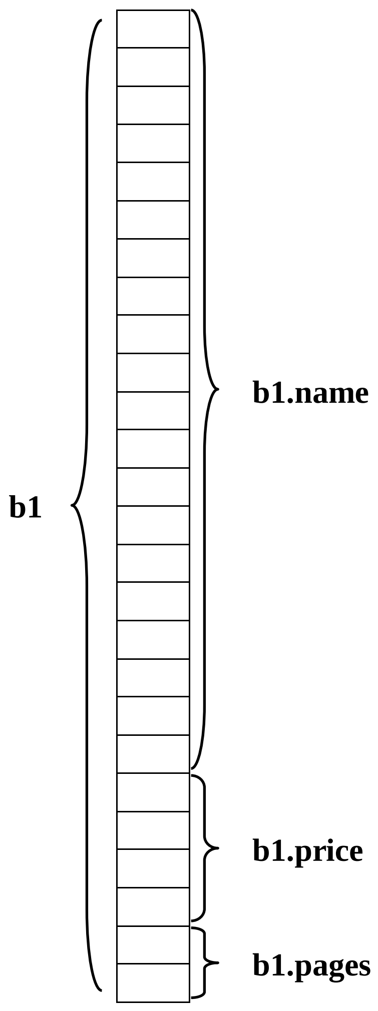


Figure: - Structure members in the memory.

- **Combining specifier and definition:-**

Specifying (or declaring) the structure and defining structure variable can be combined as shown in program35 below,

- **Initializing structure members:-**

The structure members can be initialized when they are defined as shown in program36.

- **One structure variable can be assigned to other using assignment operator (=) as shown in program37.**

- **Why we need structure?**

We know that array is a collection of elements of same data type. And when we want to store together multiple values of the same data type then obviously we use array. But suppose you want to store together multiple values of the different data type then we need different data type that can store values of different data type. And 'C' provides a special data type-the structure.

34. Program that demonstrates how to use structure to store dissimilar data of book.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b1;
clrscr();
printf("\nEnter name of book:");
scanf("%s",b1.name);
printf("\nEnnter book price:");
scanf("%f",&b1.price);
printf("\nEnter book pages:");
scanf("%d",&b1.pages);
printf("\n\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);
getch();
}
```

OUTPUT

```

Enter name of book:Let_Us_C

Ennter book price:198

Enter book pages:738


Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

35. Program that demonstrates how to combine structure specifier and definition.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
}b1;

void main()
{
clrscr();
printf("\nEnter name of book:");
scanf("%s",b1.name);
printf("\nEnnter book price:");
scanf("%f",&b1.price);
printf("\nEnter book pages:");
scanf("%d",&b1.pages);
printf("\n\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);
getch();
}
```

OUTPUT

```

Enter name of book:Let_Us_C

Ennter book price:198

Enter book pages:738


Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

36. Program that demonstrates how to initialize structure members.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b1={"Let_Us_C",198,738};
clrscr();
printf("\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);
getch();
}
```

OUTPUT

```

Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

37. Program that demonstrates how to assign one structure variable to other.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b1={"Let_Us_C",198,738};
struct book b2;
b2=b1;
clrscr();

printf("\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);

printf("\n\n\nDetails of book b2:\n");
printf("\nBook name=%s",b2.name);
printf("\nBook price=%f",b2.price);
printf("\nBook pages=%d",b2.pages);
getch();
}
```

OUTPUT

```

Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738


Details of book b2:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

Note: - You can assign one structure variable to another.

38. Program with multiple structure variables.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b1={"Let_Us_C",198,738};
struct book b2={"Basic_Electronics",135.25,300};
struct book b3={"Engg_Math",350.50,800};

clrscr();

printf("\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);

printf("\n\n\nDetails of book b2:\n");
printf("\nBook name=%s",b2.name);
printf("\nBook price=%f",b2.price);
printf("\nBook pages=%d",b2.pages);

printf("\n\n\nDetails of book b3:\n");
printf("\nBook name=%s",b3.name);
printf("\nBook price=%f",b3.price);
printf("\nBook pages=%d",b3.pages);

getch();
}
```

OUTPUT

```

Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738


Details of book b2:

Book name=Basic_Electronics
Book price=135.250000
Book pages=300


Details of book b3:

Book name=Engg_Math
Book price=350.500000
Book pages=800
```

- **Array of structure:**

We know that array is a collection of elements of same data type. And structure is a collection of variables, where variables can be of different data types.

Suppose you want to store data of 100 books then definitely you would not like to define 100 structure variables from b1 to b100. Rather you would like to define only one variable that can store data of 100 books. And you can achieve this using array of structure as shown below,

struct book b[100];

See program39 where we have defined array of structure that can store data of 3 books.

39. Program that demonstrates array of structure.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b[3];
int i;
clrscr();
for(i=0;i<3;i++)
{
printf("\n\nEnter details of book:\n");
printf("Enter name of book:");
scanf("%s",b[i].name);
printf("Enter book price:");
scanf("%f",&b[i].price);
printf("Enter book pages:");
scanf("%d",&b[i].pages);
}

for(i=0;i<3;i++)
{
printf("\n\n\nDetails of book:\n");
printf("\nBook name=%s",b[i].name);
printf("\nBook price=%f",b[i].price);
printf("\nBook pages=%d",b[i].pages);
}

getch();
}
```

OUTPUT

```

Enter details of book:
Enter name of book:Let_Us_C
Enter book price:198
Enter book pages:738


Enter details of book:
Enter name of book:Basic_Electronics
Enter book price:135.25
Enter book pages:300


Enter details of book:
Enter name of book:Engg_Math
Enter book price:350.50
Enter book pages:800



Details of book:

Book name=Let_Us_C
Book price=198.000000
Book pages=738


Details of book:

Book name=Basic_Electronics
Book price=135.250000
Book pages=300


Details of book:

Book name=Engg_Math
Book price=350.500000
Book pages=800
```

➢ Pointer to Structure

We can have pointer pointing to structure as shown in program40.

Note: - We know that we can access structure members through structure variable using dot(**.**) operator. And we can access structure members through pointer to structure using the arrow (**->**) operator.

That means on left side of dot (**.**) there must be a structure variable. And on the left side of arrow (->) there must be pointer to structure.

40. Program that demonstrates pointer to structure.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void main()
{
struct book b1={"Let_Us_C",198,738};
struct book *ptr;
ptr=&b1;
clrscr();
printf("\nDetails of book b1:\n");
printf("\nBook name=%s",b1.name);
printf("\nBook price=%f",b1.price);
printf("\nBook pages=%d",b1.pages);

printf("\n\n\nDetails of book b1:\n");
printf("\nBook name=%s",ptr->name);
printf("\nBook price=%f",ptr->price);
printf("\nBook pages=%d",ptr->pages);

getch();
}
```

OUTPUT

```

Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738


Details of book b1:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

Note:-

- Here **struct book *ptr;** statement indicates that ptr is a pointer variable that can hold address of variable of type struct book.
- The statement **ptr=&b1;** indicates that ptr is now holding address of b1.
- To access members of b1 through pointer ptr use arrow (**->)** operator.
- To access members of b1 through b1 use dot (**.**) operator.

➢ How to pass the entire structure variable to a function.

41. Program that demonstrates how to pass the entire structure variable to a function.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void display(struct book b);

void main()
{
struct book b1={"Let_Us_C",198,738};

clrscr();
display(b1);
getch();
}

void display(struct book b)
{
printf("\nDetails of book:\n");
printf("\nBook name=%s",b.name);
printf("\nBook price=%f",b.price);
printf("\nBook pages=%d",b.pages);
}
```

OUTPUT

```

Details of book:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

Note:-
- We can pass the entire structure variable to a function.
- See function display is taking one argument of type struct book.

- **How to pass address of structure variable to a function**

42. Program that demonstrates how to pass the address of structure variable to a function.

```
#include<stdio.h>
#include<conio.h>
struct book
{
char name[20];
float price;
int pages;
};

void display(struct book *ptr);

void main()
{
struct book b1={"Let_Us_C",198,738};
clrscr();
display(&b1);
getch();
}

void display(struct book *ptr)
{
printf("\nDetails of book:\n");
printf("\nBook name=%s",ptr->name);
printf("\nBook price=%f",ptr->price);
printf("\nBook pages=%d",ptr->pages);
}
```

OUTPUT

```

Details of book:

Book name=Let_Us_C
Book price=198.000000
Book pages=738
```

Note:-
- We can pass the address of structure variable to a function.
- See function display is taking one pointer argument of type struct book.

43. Program that demonstrates how to store time using structure.

```c
#include<stdio.h>
#include<conio.h>
struct time
{
int hours;
int minutes;
};
void main()
{
struct time t1;
clrscr();
printf("\nEnter details of time t1:\n");
printf("\nEnter hours:");
scanf("%d",&t1.hours);
printf("\nEnter minutes:");
scanf("%d",&t1.minutes);
printf("\n\nDetails of time t1:");
printf("\nHours=%d",t1.hours);
printf("\nMinutes=%d",t1.minutes);
getch();
}
```

OUTPUT

```

Enter details of time t1:

Enter hours:2

Enter minutes:35


Details of time t1:
Hours=2
Minutes=35
```

44. Program that demonstrates how to store time using structure.

```c
#include<stdio.h>
#include<conio.h>
struct time
{
int hours;
int minutes;
};
void main()
{
struct time t1;
clrscr();
t1.hours=2;
t1.minutes=35;
printf("\n\nDetails of time t1:");
printf("\nHours=%d",t1.hours);
printf("\nMinutes=%d",t1.minutes);
getch();
}
```

OUTPUT

```

Enter details of time t1:

Enter hours:2

Enter minutes:35


Details of time t1:
Hours=2
Minutes=35
```

45. Program that demonstrates addition of two time.

```
#include<stdio.h>
#include<conio.h>
struct time
{
int hours;
int minutes;
};
void main()
{
struct time t1;
struct time t2;
struct time t3;
clrscr();
t1.hours=2;
t1.minutes=35;

t2.hours=1;
t2.minutes=35;

t3.minutes=t1.minutes+t2.minutes;
t3.hours=t3.minutes/60;
t3.minutes=t3.minutes%60;
t3.hours=t1.hours+t2.hours+t3.hours;

printf("\n\nDetails of time t1:");
printf("\nHours=%d",t1.hours);
printf("\nMinutes=%d",t1.minutes);

printf("\n\nDetails of time t2:");
printf("\nHours=%d",t2.hours);
printf("\nMinutes=%d",t2.minutes);

printf("\n\nDetails of time t3:");
printf("\nHours=%d",t3.hours);
printf("\nMinutes=%d",t3.minutes);

getch();
}
```

OUTPUT

```

Details of time t1:
Hours=2
Minutes=35

Details of time t2:
Hours=1
Minutes=35

Details of time t3:
Hours=4
Minutes=10
```

III. Union:-

Unions are similar to structure. The main difference between structure and union is in terms of storage. In structure, each structure member has its own memory location, whereas in union, all union members share the same memory locations.

For example,

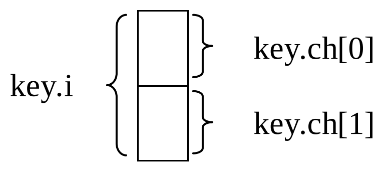
46. Program demonstrates union.

```
#include<stdio.h>
#include<conio.h>
union a
{
int i;
char ch[2];
};
void main()
{
union a key;
clrscr();
key.i=512;
printf("\nDetails of Key:");
printf("\ni=%d\n",key.i);
printf("ch[0]=%d\n",key.ch[0]);
printf("ch[1]=%d\n",key.ch[1]);
getch();
}
```

OUTPUT

```
Details of Key:
i=512
ch[0]=2
ch[1]=0
```

See above program, where we have declared **union a,** and then we have defined one variable **key** of type **union a.** We can access union members using dot (.) operator. This is similar to the way we did with structure. The main difference between structure and union is in terms of storage. **In structure, each structure member has its own memory location, whereas in union, all union members share the same memory locations. And the size of union variable is equal to the size of longest member in the union.** Therefore **key** occupy 2 bytes in memory. And this 2 bytes space is shared by both i and ch.



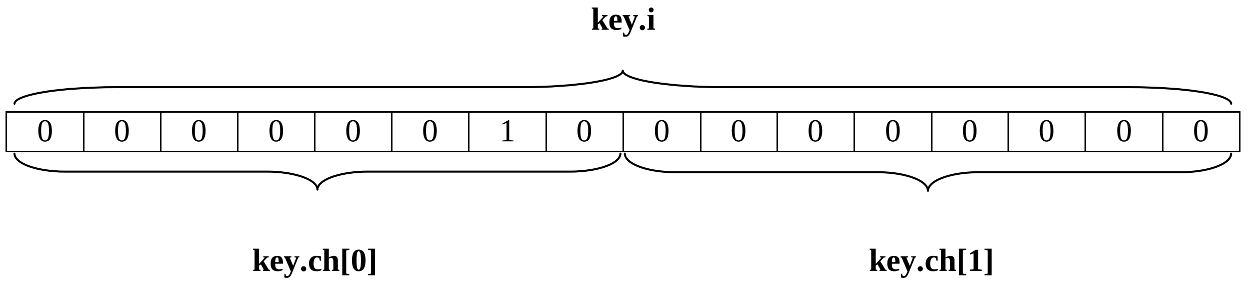
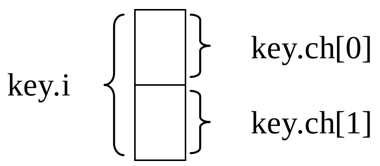
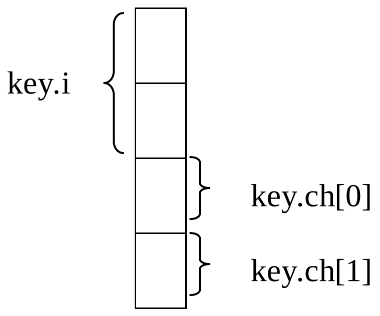


Fig: - Union members in memory.

Difference between Union and Structure,

Union	Structure

union a
{
int i;
char ch[2];
};
union a key;
	struct a
	{
	int i;
	char ch[2];
	};
	struct a key;

○ In union, all union members share the same memory locations. ○ The size of union variable is equal to the size of longest member in the union. Therefore key occupy 2 bytes in memory. And this 2 bytes space is shared by both i and ch.	○ In structure, each structure member has its own memory location. ○ The size of structure variable is equal to the sum of sizes of all members in the structure. Therefore key occupy 4 bytes in memory. 2 bytes for i and 2 bytes for ch.
 **Fig: - Union members in memory.**	 **Fig: -Structure members in memory.**
Only the first member of union can be initialized as, union a key={512};	All members of structure can be initialized as, struct a key={512,"aa"};