```
    int  day, mon, year;
    } dob'
  } student;
```

In the above declaration, student is a variable of structure type consisting of the members namely rollno, name, avgmarks and the structure variable dob.

The dob structure is within another structure **student** and thus structure is nested. In this type of definitions, the elements of the require structure can be referenced by specifying appropriate qualifications to it, using the period operator (.) .

For example, **student. dob. day** refers to the element day of the inner structure dob.

5.3 Structures and Arrays

Array is group of identical stored in consecutive memory locations with a single / common variable name. This concept can be used in connection with the structure in the following ways**.**

a. Array of structures

b. structures containing arrays (or) arrays within a structure

c. Arrays of structures contain arrays.

5.3.1 Array of Structures

Student details in a class can be stored using structure data type and the student details of entire class can be seen as an array of structure.

Examaple

```
struct student
{
   int  rollno;
   int   year;
  int tmarks;
}
struct student class[40];.
```

In the above class [40] is structure variable accommodating a structure type student up to 40.

The above type of array of structure can be initialized as under

```
struct  student class [2] = { 001,2011,786},
                            { 002, 2012, 710}
                             };
```

i.e class[0] . rollno = 001

class[0] . year = 2011

class[0] . tmarks = 777 and

class[1] . rollno = 002

class[1] . year = 2012

class[1] . tmarks = 777 .

5.3.2 Structures containing Arrays

A structure data type can hold an array type variable as its member or members. We can declare the member of a structure as array data type similar to int, float or char.

Example

```
struct employee
{
        char ename [20];
        int eno;
};
```

In above, the structure variable employee contains character array type ename as its member. The initialization of this type can be done as usual.

```
struct employee = { ' Rajashekar', 7777};
```

5.3.3 Arrays of Structures Contain Arrays

Arrays of structures can be defined and in that type of structure variables of array type can be used as members.

Example

```
struct rk
{
   int empno;
  char ename[20];
   flat  salary;
  } mark[50];
```

In the above, mark is an array of 50 elements and such element in the array is of structure type rk. The structure type rk, in turn contains ename as array type which is a member of the structure. Thus mark is an array of sutures and these structures in turn holds character names in array ename.

The initialization of the above type can be done as:

```
{
7777, ' Prasad' , 56800.00}
};
i.e mark[0] . empno = 7777;
   mark[0] . eame  = 'Prasad';
    mark[0] . salary =  56800.00
```

Program

Write a C program to accept the student name, rollno, average marks present in the class of student and to print the name of students whose average marks are greater than 40 by using structure concept with arrays.

```
# include <stdio.h>
main( )
{
 int  i, n,
 struct
 {
 char name [20];
```

```
  int rollno;
  flat avgmarks;
 }
class [40];
 printf(" Enter the no. of students in the class\n"0;
scanf ( " %d", & n );
for ( i = 0, i < n, i++)
{
    print ( " Enter students name, rollno, avgmarks\n");
    scanf( " %s %d", &class[i].name, class[i].rollno, &class[i].avgmarks)'
}
printf (" The name of the students whose average");
printf ( " marks is greater than 40 \n");
for ( i = 0, i < n, i++)
if ( class[i].avgmarks > 40)
pirintf (" %s", class[i].name);
}
```

5.3.4 Advantages of Structure Type over Array Type Variables

1. Using structures, we can group items of different types within a single entity, which is not possible with arrays, as array stores similar elements.
2. The position of a particular structure type variable within a group is not needed in order to access it, whereas the position of an array member in the group is required, in order to refer to it.
3. In order to store the data about a particular entity such as a 'Book', using an array type, we need three arrays, one for storing the 'name', another for storing the 'price' and a third one for storing the 'number of pages' etc., hence, the overhead is high. This overhead can be reduced by using structure type variable.

4. Once a new structure has been defined, one or more variables can be declared to be of that type.

5. A structure type variable can be used as a normal variable for accepting the user's input, for displaying the output etc.,

6. The assignment of one 'struct' variable to another, reduces the burden of the programmer in filling the variable's fields again and again.

7. It is possible to initialize some or all fields of a structure variable at once, when it is declared.

8. Structure type allows the efficient insertion and deletion of elements but arrays cause the inefficiency.

9. For random array accessing, large hash tables are needed. Hence, large storage space and costs are required.

10. When structure variable is created, all of the member variables are created automatically and are grouped under the given variable's name.

5.4 Structure Contains Pointers

A pointer variable can also be used as a member in the structure.

Example:

```
struct
{
        int *p1;
        int  * p2;
} *rr;
```

In the above, *rr is a pointer variable of structure type which holds inside it another two pointer variables p1 and p2 as its members.

```
# include <stdio.h>
main( )
{
 sturct
    int  *p1, *p2;
} *rr;
```