Structures

Definition of structure

A structure is a data type with the collection of homogeneous or heterogeneous data element **or** a set of data items with the collection of same type or of different data types

Construction of structures or defining a structure

We create the structure with following syntax

struct structure-name
{
data-type member1;
data-type member2;

};

The keyword struct declares the structure . The structure-name is any valid identifier given to identify the structure. The each variable declared in the structure is called member or field of the structure, it can also called as **template**. The structure does not allocate the memory unless we create the variable of structure

Example

```
struct  student
{
char name[20];
int rno;
float fees;
};
```

Note: after closing flower bracket of structure definition we should put semicolon
The structure definition will not allocate the memory. The memory will allocate only after creating the object of structure

Declaration of structure

To allocate the memory for structure we have to declare it after construction of structure with following syntax

struct structure-name var1,var2,…;

Example ; For above created structure we can declare as

```
struct  student std;
```

where std is a variable of structure student
We can combine the construction with declaration as follows

```
struct  student
{
char name[20];
int rno;
float fees;
} std;
```

The above declaration can also be modified without using the structure-name as follows

```
struct
{
char name[20];
int rno;
float fees;
}std;
```

Note

- The structure is always terminated by semicolon
- Structure members cannot be initialized within side of template we can initialize only when we declare it
- The definition of the structure should come either before the main or immediately after the main before declaration of any other variable
- Declaration of the structure always comes after the definition

Accessing structure members or Accessing elements of structure

The members of structures can be accessed by using dot (**.**) operators
Example to access rno of above declaration we can write as **std.rno**

Initialization of structure

At the time of declaring the structure if we assign the values for its members then it is called initialization. We can initialize by using two different methods and are
Method 1

```
struct  student
{
char name[20];
int rno;
float fees;
}std={“sms”, 1001,5674.50};
```

Or Method 2
By using dot operator

```
struct  student
{
char name[20];
int rno;
float fees;
}std;
std.name=”sms”;
std.rno=1001;
std.fees=5674.50;
```

Array of structures

An array of structure is an array in which each element of the array is a structure. Thus it is a collection of structures put together in an array
When the number of variables have to assign to the same structure an array may be used instead of using number of variables. Example
struct student std[20];
The array called std contains 20 elements of the structure student.
Note: If we specify the array subscript while creating object of the array then we call it as array of structure

Nested structure

If one structure is present within side of another structure then is called the nested structure

```
struct date
{
 int dd;
int mm;
int yy;
};
struct student
{
int rno;
char name[20]; // array within side of the structure
struct date dob; //illustrate the nested structure
}std;
```

Note

- The structure can be declared outside of main or immediately after the main before declaring any other variable
- If any one member of the structure contain array subscript then we call it as array within side of the structure