Structure in C Language

- **What is a structure?**

A structure is a user defined data type in C and C++ language. It creates a data type that can be used to group items of possibly different types into a single type.

- **How to create a structure?**

'struct' keyword is used to create a structure. An example is given below

Syntax:
```
struct<structure_name>
{
        data_type variable1;
        data_type variable2;
        data_type variable3;
        - - - -
        - - - -
};
```
Example:
```
struct address
{
char name[50];
char city[50];
char state[20];
int pin;
};
```

- **How to declare structure variables?**

A structure variable can either be declared with structure declaration or as a separate declaration like basic types.

// Type-1Structure variable declaration.
```
struct Point
{
        int x, y;
} p1;                   // The variable p1 is declared with 'Point'
```

// Type-2 Structure variable declaration.
```
struct Point
{
        int x, y;
};

void main()
{
        struct Point p1;        // The variable p1 is declared like a normal variable
}
```

- **How to initialize structure members?**

Structure members cannot be initialized with declaration.
For example, the following C program used to show **WRONG** initialization

```
struct Point
{
        int x = 0;                  // COMPILER ERROR:  cannot initialize members here
        int y = 0;                  // COMPILER ERROR:  cannot initialize members here
};
```

The reason for above error is simple, when a datatype is declared; no memory is allocated for it. Memory is allocated only when variables are created.

Structure members can be initialized using curly braces '{}'.
For example, following is a **CORRECT** initialization.

```
struct Point
{
        int x, y;
};

void main()
{
        struct Point p1 = {0, 1};
}
```

- **How to access structure elements?**

Structure members are accessed using dot (.) operator.

```
#include<stdio.h>
#include<conio.h>
structTEST
{
        int x, y;
};

void main()
{
        structTESTt1 = {0, 1};

        // Accessing members of TEST t1
        t1.x = 20;
        printf ("x = %d, y = %d", t1.x, t1.y);
        getch();
}
```

- **What is an array of structures?**

Structure is collection of different data type. An object of structure represents a single record in memory, but if we want more than one record of structure type, we have to create an array of structure.Like other primitive data types, we can create an array of structures.

```
#include<stdio.h>
#include<conio.h>

struct Point     // Structure declaration
{
        int x, y;
};

void main()
{
        struct Point arr[10];           // Create an array of structures

        arr[0].x = 10;                  // Access array members
        arr[0].y = 20;

        printf("%d %d", arr[0].x, arr[0].y);
        getch();
}
```

- **Limitations of C Structures**

In C language, Structures provide a method for packing together data of different types. A Structure is a helpful tool to handle a group of logically related data items. However, C structures have some limitations.

a. The C structure does not allow the struct data type to be treated like built-in data types.We cannot use operators like +,- etc. on Structure variables.
b. **No Data Hiding:**C Structures do not permit data hiding. Structure members can be accessed by any function, anywhere in the scope of the Structure.
c. **Functions inside Structure:** C structures do not permit functions inside Structure.
d. **Static Members:** C Structures cannot have static members inside their body.
e. **Access Modifiers:** C Programming language do not support access modifiers. So they cannot be used in C Structures.