Structure

The **structure** is the collection of different data types grouped under the same name using the **struct** keyword. It is also known as the user-defined data type that enables the programmer to store different data type records in the Structure. Furthermore, the collection of data elements inside the Structure is termed as the **member.**

For example, suppose we want to create the records of a person containing name, age, id, city, etc., and these records cannot be grouped in the single dimension array. Therefore, we use the Structure to store multiple collections of data items.

Syntax to define Structure

```
struct structure_name
{
char name[40];
int roll_no;
float percentage;
char sub[30];
};
struct structure_name  str;  // str is a variable of the structure_name
```

Here **structure_name** is the name of the Structure that is defined using the **struct** keyword. Inside the structure_name, it collects different data types (int, char, float) elements known as the **member**. And the last, **str**, is a variable of the Structure.

Program to demonstrate the structure and access their member

In this program creates a student structure and access its member using structure variable and dot (.) operator.

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

// create the Structure of student to store multiples items
struct student
{
   char name[ 30];
   int roll_no;
   char state[100];
   int age;
};
struct student s1, s2; // declare s1 and s2 variables of student structure

int main()
{
   // records of the student s1
   strcpy (s1.name, "John");
   s1.roll_no = 1101;
   strcpy (s1.state, "Los Angeles");
   s1.age = 20;

   // records of the student s2
   strcpy (s2.name, " Mark Douglas");
   s2.roll_no = 111;
   strcpy (s2.state, "California");
   s2.age = 18;

   // print the details of the student s1;
   printf (" Name of the student s1 is: %s\t ", s1.name);
   printf (" \n Roll No. of the student s1 is: %d\t ", s1.roll_no);
   printf (" \n The state of the student s1 is: %s\t ", s1.state);
   printf (" \n Age of the student s1 is: %d\t ", s1.age);


      // print the details of the student s2;
   printf ("\n Name of the student s1 is: %s\t ", s2.name);
   printf (" \n Roll No. of the student s1 is: %d\t ", s2.roll_no);
   printf (" \n The state of the student s1 is: %s\t ", s2.state);
   printf (" \n Age of the student s1 is: %d\t ", s2.age);
   return 0;
}
```

Output:

```
Name of the student s1 is: John
Roll No. of the student s1 is: 1101
 state of the student s1 is: Los Angeles
Age of the student s1 is: 20
Name of the student s1 is:  Mark Douglas
Roll No. of the student s1 is: 111
 The state of the student s1 is: California
Age of the student s1 is: 18
```

Explanation of the program: As we can see in the above program, we have created a structure with name **student,** and the student structure contains different members such as name (char), roll_no (int), state (char), age (int). The student structure also defines two variables like **s1** and **s2**, that access the structure members using dot operator inside the main() function.

Structure Pointer

The **structure pointer** points to the address of a memory block where the Structure is being stored. Like a pointer that tells the address of another variable of any data type (int, char, float) in memory. And here, we use a structure pointer which tells the address of a structure in memory by pointing pointer variable **ptr** to the structure variable.

Declare a Structure Pointer

The declaration of a structure pointer is similar to the declaration of the structure variable. So, we can declare the structure pointer and variable inside and outside of the main() function. To declare a pointer variable in C, we use the asterisk (*) symbol before the variable's name.

```
struct structure_name *ptr;
```

Initialization of the Structure Pointer

```
ptr = &structure_variable;
```

We can also initialize a Structure Pointer directly during the declaration of a pointer.

```
struct structure_name *ptr = &structure_variable;
```

Access Structure member using pointer:

There are two ways to access the member of the structure using Structure pointer:

- © Using (*) asterisk or indirection operator and dot (.) operator.
- © Using arrow (->) operator or membership operator.

Example Program

The following program shows the usage of pointers to structures –

```
#include<stdio.h>
struct student{
   int sno;
   char sname[30];
   float marks;
};
main ( ){
   struct student s;
   struct student *st;
   printf("enter sno, sname, marks:");
   scanf ("%d%s%f", & s.sno, s.sname, &s. marks);
   st = &s;
   printf ("details of the student are");
   printf ("Number = %d\n", st ->sno);
   printf ("name = %s\n", st->sname);
   printf ("marks =%f\n", st ->marks);
   getch ( );
}
```

Output:

```
enter sno, sname, marks:1 Lucky 98
details of the student are:
Number = 1
name = Lucky
marks =98.000000
```