Structure

- Structure is a user-defined datatype in C language which allows us to combine data of different types together.
- Structure helps to construct a complex data type which is more meaningful.
- It is somewhat similar to an Array, but an array holds data of similar type only.
- But structure on the other hand, can store data of any type, which is practical more useful.
- In structure, data is stored in form of **records**.

Defining a structure

- struct keyword is used to define a structure. struct defines a new data type which is a collection of primary and derived data types.
- **Syntax:**
- struct [structure_tag]

```
{   //member variable 1
        //member variable 2
        //member variable 3 ...
    }[structure_variables];
```

```
struct Student
{
char name[25];
int age;
char branch[10];
// F for female and M for male
char gender;
};
```

- Here struct Student declares a structure to hold the details of a student which consists of 4 data fields, namely name, age, branch and gender. These fields are called **structure elements or members**.
- Each member can have different data type, like in this case, name is an array of char type and age is of int type etc. **Student** is the name of the structure and is called as the **structure tag**.

- Struct student
- {
- char name[10]; 10 bytes
- int roll_no; 4bytes
- float avg_marks; 8 bytes total =22bytes
- }
- Size of the structure is 22 bytes no space is reserved for the above structure
- Memory is reserved only if the above definition is associated with variable

Structure Variables

- Struct
- {
- Type member1;
- Type member2;

- }v1,v2……..vn;
- struct
- {
- char name[10];
- int roll_no;
- float avg_marks;
- }cse,ise;

Structure declaration

- Structure can be declared using three different ways
- Tagged structure
- Structure variables
- Type define structures

Tagged structure

- struct student
- {
- char name[10]; 10 bytes
- int roll_no; 4bytes
- float avg_marks; 8 bytes total =22bytes
- };
- struct student cse,ise;

Type defined Structure

- The structure definition with keyword typedef is called type defined structure
- typedef makes the code short and improves readability.
- In the above discussion we have seen that while using structs every time we have to use the lengthy syntax, which makes the code confusing, lengthy, complex and less readable.
- The simple solution to this issue is use of typedef. It is like an alias of struct.

- Syntax
- typedef struct
- {
- type1 member1;
- type2 member2;
- }TYPE_ID;

- typedef struct
- {
- char name[10];
- int roll_num;
- float avg-marks;
- }STUDENT;

- STUDENT is the type created by the user,it can be called as user defined data type.
- STUDENT can be used as datatype and declare variables
- STUDENT cse,ise;
- the var cse and ise are variables of type STUDENT

Structure Intialization

- Structure can be intialized various way
- Struct employee
- {
- char name[20];
- int salary;
- int id
- }a={“raj”,10897,2001);

- Structure can be intialized various ways
- struct employee
- {
- char name[20];
- int salary;
- int id
- }
- struct employee a={“raj”,10897,2001};

Accessing structures

- **Access members of a structure**
- There are two types of operators used for accessing members of a structure.
- . - Member operator
- -> - Structure pointer operator

Struct employee
{
char name[20];
int salary;
int id
}a={“raj”,10897,2001);
The members of a structure can be accessed by specifying the variable followed by dot operator followed by the name of the member.
a.name , a.salary, a.id

```
#include<stdio.h>
#include<string.h>
struct Student
{ char name[25];
int age;
char branch[10];
//F for female and M for male
char gender;
};
int main()
{
struct Student s1;
/* s1 is a variable of Student type and age is a member of Student */
s1.age = 18;
 /* using string function to add name */
strcpy(s1.name, "raaj");
/* displaying the stored values */
printf("Name of Student 1: %s\n", s1.name);
printf("Age of Student 1: %d\n", s1.age);
 return 0; }
```

```
struct student
{
   char name[50];
   int roll;
   float marks;
} s[10];
int main()
{
   int i;
   printf("Enter information of students:\n");
   // storing information
   for(i=0; i<10; ++i)
   {
      s[i].roll = i+1;
      printf("\nFor roll number%d,\n",s[i].roll);
      printf("Enter name: ");
      scanf("%s",s[i].name);
      printf("Enter marks: ");
      scanf("%f",&s[i].marks);
      printf("\n");
   }
```

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

struct student
{
        int id;
        char name[20];
        float percentage;
};

int main()
{
        struct student record = {0}; //Initializing to null

        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;

        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
        return 0;
}
```

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

struct student
{
        int id;
        char name[20];
        float percentage;
} record;

int main()
{

        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;

        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
        return 0;
```

Structure using Pointer

- Dot(.) operator is used to access the data using normal structure variable and arrow (->) is used to access the data using pointer variable.

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

struct student
{
    int id;
    char name[30];
    float percentage;
};

int main()
{
    int i;
    struct student record1 = {1, "Raju", 90.5};
    struct student *ptr;

    ptr = &record1;

        printf("Records of STUDENT1: \n");
        printf("  Id is: %d \n", ptr->id);
        printf("  Name is: %s \n", ptr->name);
        printf("  Percentage is: %f \n\n", ptr->percentage);

    return 0;
}
```

Array of structures

```
#include<stdio.h>

struct Point
{
   int x, y;
};

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

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

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