**Introduction to 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.

**Ex : Storing Student information like name, age ,branch and gender**

struct Student  
{  
 char name[25];  
 int age;  
 char branch[10];  
 char gender;  
};

**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 datatypes.

**Syntax:**

struct [structure\_tag]  
{  
 //member variable 1  
 //member variable 2  
 //member variable 3  
 ...  
}

[structure\_variables];

**Declaring Structure Variables**

It is possible to declare variables of a structure, either along with structure definition or after the structure is defined. Structure variable declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

1. Declaring Structure variables separately
2. Declaring Structure variables with structure definition

#### **Declaring Structure variables separately //recommended**

struct Student  
{  
 char name[25];  
 int age;  
 char branch[10];  
 //F for female and M for male  
 char gender;  
};  
  
struct Student S1, S2; //declaring variables of struct Student

**2. Declaring Structure variables with structure definition**

struct Student  
{  
 char name[25];  
 int age;  
 char branch[10];  
 char gender;  
}S1, S2;

Here S1 and S2 are variables of structure Student. However this approach is not much recommended.

**Accessing Structure Members**

Structure members can be accessed and assigned values in a number of ways. Structure members have no meaning individually without the structure. In order to assign a value to any structure member, the member name must be linked with the **structure** variable using a dot . operator also called **member access** operator.

Ex :

#include<stdio.h>

#include<string.h>

struct Student

{

char name[25];

int age;

char branch[10];

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, "Viraaj");

/\*

displaying the stored values

\*/

printf("Name of Student 1: %s\n", s1.name);

printf("Age of Student 1: %d\n", s1.age);

return 0;

}

//output

Name of Student 1: Viraaj  
Age of Student 1: 18

**Struct initialization**

Like a variable of any other datatype, structure variable can also be initialized at compile time.

struct Patient  
{  
 float height;  
 int weight;   
 int age;   
};  
  
struct Patient p1 = { 180.75 , 73, 23 }; //initialization

or,

struct Patient p1;  
p1.height = 180.75; //initialization of each member separately  
p1.weight = 73;  
p1.age = 23;

**Array of Structure**

We can also declare an array of structure variables. in which each element of the array will represent a structure variable.

Example : struct employee emp[5];

Program to read information of first 3 employees

#include<stdio.h>

struct Employee

{

char ename[10];

int sal;

};

struct Employee emp[5];

int i, j;

void ask()

{

for(i = 0; i < 3; i++)

{

printf("\nEnter %dst Employee record:\n", i+1);

printf("\nEmployee name:\t");

scanf("%s", emp[i].ename);

printf("\nEnter Salary:\t");

scanf("%d", &emp[i].sal);

}

printf("\nDisplaying Employee record:\n");

for(i = 0; i < 3; i++)

{

printf("\nEmployee name is %s", emp[i].ename);

printf("\nSlary is %d", emp[i].sal);

}

}

void main()

{

ask();

}

**Nested Structures**

Nesting of structures, is also permitted in C language. Nested structures means, that one structure has another structure as member variable.

#include<stdio.h>

struct **Employee**

{

int Id;

char Name[25];

float Salary;

struct **Address**

{

char HouseNo[25];

char City[25];

char PinCode[25];

}Add;

};

void main()

{

int i;

struct Employee E;

printf("\n\tEnter Employee Id : ");

scanf("%d",&E.Id);

printf("\n\tEnter Employee Name : ");

scanf("%s",&E.Name);

printf("\n\tEnter Employee Salary : ");

scanf("%f",&E.Salary);

printf("\n\tEnter Employee House No : ");

scanf("%s",&E.Add.HouseNo);

printf("\n\tEnter Employee City : ");

scanf("%s",&E.Add.City);

printf("\n\tEnter Employee House No : ");

scanf("%s",&E.Add.PinCode);

printf("\nDetails of Employees");

printf("\n\tEmployee Id : %d",E.Id);

printf("\n\tEmployee Name : %s",E.Name);

printf("\n\tEmployee Salary : %f",E.Salary);

printf("\n\tEmployee House No : %s",E.Add.HouseNo);

printf("\n\tEmployee City : %s",E.Add.City);

printf("\n\tEmployee House No : %s",E.Add.PinCode);

}

**Note** : two ways to create nested structs

struct **Employee**

{

int Id;

char Name[25];

float Salary;

struct **Address**

{

char HouseNo[25];

char City[25];

char PinCode[25];

}Add;

};

Same as =>

struct Address  
 {  
 char HouseNo[25];  
 char City[25];  
 char PinCode[25];  
 };  
  
 struct Employee  
 {  
 int Id;  
 char Name[25];  
 float Salary;  
 **struct Address Add;**  
 };

### **Structure as Function Arguments**

We can pass a structure as a function argument just like we pass any other variable or an array as a function argument.

#include<stdio.h>

struct Student

{

char name[10];

int roll;

};

void show(struct Student st);

void show(struct Student st)

{

printf("\nstudent name is %s", st.name);

printf("\nroll is %d", st.roll);

}

void main()

{

struct Student std;

printf("\nEnter Student record:\n");

printf("\nStudent name:\t");

scanf("%s", std.name);

printf("\nEnter Student rollno.:\t");

scanf("%d", &std.roll);

show(std);

}