Structure in C++

Structure is commonly referred to as user-defined data type. Structure is similar to an array but the only difference is that array is collection of similar data type on the other hand structure is collection of different data type. A structure can contain any data type including array and another structure as well. Each variable declared inside structure is called member of structure.

Structure declaration

Declaration of structure must start with the keyword ***struct*** followed by the structure name and structure's member variables are declared within braces.

Syntax for declaring structure

```
struct structure-name
{
     datatype var1;
     datatype var2;
     ----------
     ----------
     datatype varN;
};
```

Example for declaring structure

```
struct Employee
{
     int Id;
     char Name[25];
     int Age;
     long Salary;
};
```

Accessing the structure members

We have to create an object of structure to access its members. Object is a variable of type structure. Structure members are accessed using the dot operator(.) between structure's object and structure's member name.

Syntax for creating object

```
structure-name obj;
```

Example for creating object & accessing structure members

```
#include<iostream.h>

struct Employee
{
      int Id;
      char Name[25];
      int Age;
      long Salary;
};

void main()
{

      Employee E;            //Statement 1

            cout << "\nEnter Employee Id : ";
            cin >> E.Id;

            cout << "\nEnter Employee Name : ";
            cin >> E.Name;

            cout << "\nEnter Employee Age : ";
            cin >> E.Age;

            cout << "\nEnter Employee Salary : ";
            cin >> E.Salary;

            cout << "\n\nEmployee Id : " << E.Id;
            cout << "\nEmployee Name : " << E.Name;
            cout << "\nEmployee Age : " << E.Age;
            cout << "\nEmployee Salary : " << E.Salary;


}
```

Output :

```
Enter Employee Id : 1
Enter Employee Name : Ahmed
Enter Employee Age : 29
Enter Employee Salary : 45000
```

Statement 1 is creating an object E of Employee type.

Initialization of structure

Like normal variable structures can be initialized at the time of declaration. Initialization of structure is almost similar to initializing array. The structure object is followed by equal sign and the list of values enclosed in braces and each value is separated with comma.

Example for declaring & initializing structure at same time

```
#include<iostream.h>

struct Employee
{
      int Id;
      char Name[25];
      int Age;
      long Salary;
};

void main()
{

      Employee E = {2,"Ali",35,35000};

           cout << "\n\nEmployee Id : " << E.Id;
           cout << "\nEmployee Name : " << E.Name;
           cout << "\nEmployee Age : " << E.Age;
           cout << "\nEmployee Salary : " << E.Salary;

}
```

<u>Array of Structure</u>

Structure is collection of different data type. An object of structure represents a single record in memory, if we want more than one record of structure type, we have to create an array of structure or object. As we know, an array is a collection of similar type, therefore an array can be of structure type.

Syntax for declaring structure array

```
struct structure-name
{
      datatype var1;
      datatype var2;
      ----------
      ----------
      datatype varN;
};
```

```
structure-name obj [ size ];
```

Example for declaring structure array

```
#include<iostream.h>

struct Employee
{
    int Id;
    char Name[25];
    int Age;
    long Salary;
};

void main()
{
    int i;
    Employee Emp[ 3 ];        //Statement   1

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

    cout << "\nEnter details of " << i+1 << " Employee";

        cout << "\n\tEnter Employee Id : ";
        cin >> Emp[i].Id;

        cout << "\n\tEnter Employee Name : ";
        cin >> Emp[i].Name;

        cout << "\n\tEnter Employee Age : ";
        cin >> Emp[i].Age;

        cout << "\n\tEnter Employee Salary : ";
        cin >> Emp[i].Salary;
    }

    cout << "\nDetails of Employees";
    for(i=0;i<3;i++)
    cout << "\n"<< Emp[i].Id <<"\t"<< Emp[i].Name <<"\t"
          << Emp[i].Age <<"\t"<< Emp[i].Salary;


}
```

In the above example, we are getting and displaying the data of 3 employee using array of object. Statement 1 is creating an array of Employee Emp to store the records of 3 employees.

Structure and Function in C++

Using function we can pass structure as function argument

Passing Structure by Value

In this approach, the structure object is passed as function argument

here object is representing the members of structure with their values.

Example for passing structure object by value

```
#include<iostream.h>

    struct Employee
    {
        int Id;
        char Name[25];
        int Age;
        long Salary;
    };

    void Display(struct Employee);
    void main()
    {
        Employee Emp = {1,"Ahmed",29,45000};

        Display(Emp);

    }

    void Display(struct Employee E)
    {
            cout << "\n\nEmployee Id : " << E.Id);
            cout << "\nEmployee Name : " << E.Name);
            cout << "\nEmployee Age : " << E.Age);
            cout << "\nEmployee Salary : " << E.Salary);
    }
```

Union in C++

Both structure and union are collection of different datatype. They are used to group number of variables of different type in a single unit.

Difference Between Structure And Union

1. Declaration and Initialization of structure starts with struct keyword. Declaration and Initialization of union starts with union keyword.

2. Structure allocates different memory locations for all its members while union allocates common memory location for all its members. The memory occupied by a union will be large enough to hold the largest member of the union.

Union declaration

Declaration of union must start with the keyword ***union*** followed by the union name and union's member variables are declared within braces.

Syntax for declaring union

```
        union union-name
        {
            datatype var1;
            datatype var2;
            - - - - - - - - - -
            - - - - - - - - - -
            datatype varN;
        };
```

Example for creating object & accessing union members

```
#include<iostream.h>

    union Employee
    {
        int Id;
        char Name[25];
        int Age;
        long Salary;
    };

    void main()
    {

        Employee E;

            cout << "\nEnter Employee Id : ";
            cin >> E.Id;
            cout << "Employee Id : " << E.Id;

            cout << "\n\nEnter Employee Name : ";
            cin >> E.Name;
            cout << "Employee Name : " << E.Name;

            cout << "\n\nEnter Employee Age : ";
            cin >> E.Age;
            cout << "Employee Age : " << E.Age;

            cout << "\n\nEnter Employee Salary : ";
            cin >> E.Salary;
            cout << "Employee Salary : " << E.Salary;

    }
```