Structure and Union

Structure: Structures are also one of the derived data types in C. It is the collection of dissimilar data types like an int, float and char under a single name. So whenever you require putting dissimilar data type together you can use a structure.

Structure is a collection of Heterogeneous data type. All the elements of structure stored at contiguous memory locations.

It is Collection of variables under a single name. You can use variables of any type inside a structure. Each separate variable is called a member.

Definition A Structure is collection of dissimilar (heterogeneous) data elements stored at contiguous memory location.

struct keyword is used for creating Structure.

Syntax:

```
struct structure name
{
datatype1 member1;
datatype2 member2;
};
```

E.g. A Student could have name, roll no and marks etc.

```
struct student
 {
 char name[10];
 int roll_no;
 int class;
 };
```

Declaring a structure

- Like any other variables in C , a structure has to be declared, before it can used.
- A structure can be defined in C by the keyword struct followed by its name and a body enclosed in curly braces.
- The body structure contains the definition of its members and each member must have a name. The declaration ends by a semicolon. It is also called structure template.

```
struct <name>
{
data_type  variable_name1;
data_type   variable_name2
;
.

.
data_type variable_name n;
};
```

Where struct is the keyword.

"name" is the name of the structure and "variable_name" is the name of the member variable and the "data_type" is the data type of the member variable.

Let us now define the c structure to describe the information about the student,

```
struct student
{
char name [20];
 int roll;
};
```

The declaration means that the student is a type which is structure consisting of data members: name, age, roll and class.

Declare Structure Variables:

A programmer is allowed to declare one or more structure variables along with the structure declaration. There are two ways to declare structure variable:

1. By struct keyword within main() function
2. By declaring a variable at this time of defining the structure.

struct student s1, s2;

or

```
struct student
{
Int rollno'
char name[20];
}s1.s2.s3,s4;
```

Accessing Structure Members

Structure members are accessed using the structure member operator (.), also called the dot operator, between the structure name and the member name.

For example: The Structure variable s1 has four members; name, age, roll and class. These members can be designated as: s1.name, s1.age, s1.roll and s1.class respectively.

s1.name="PALVI";

s1.age=9; s1.roll=3;

s1.class="3rd"

Example of Structure

```
#include <stdio.h>

struct Person {
    char name[50];
    int age;
    float height;
};

int main() {
    // Declare a variable
    struct Person person1;


    printf("Enter name: ");
    scanf("%s", person1.name);

    printf("Enter age: ");
    scanf("%d", &person1.age);

    printf("Enter height (in meters): ");
    scanf("%f", &person1.height);
```

```
    printf("\nEntered details:\n");
    printf("Name: %s\n", person1.name);
    printf("Age: %d\n", person1.age);
    printf("Height: %.2f meters\n", person1.height);
    return 0;
}
```

Pointers and Structures

Similar to other types of pointers, we can have a structure pointer also .that means we can also point a pointer to a structure.

Pointer to structure means we can assign the address of a structure to the structure type pointer variable.

To access the member of structure by the structure pointer we can use the (->) operator.

Syntax: struct structure_name *pointer_variable_name;

Program

```
#include<stdio.h>

struct student {
    char name[20];
    int rollno;
};

int main() {
    struct student stu;
    struct student *p;
    p = &stu;

    printf("\nEnter Student Name: ");
    scanf("%s", p->name);

    printf("Enter Student Roll No: ");
    scanf("%d", &p->rollno);

    printf("\n******OUTPUT******");
    printf("\nStudent Name is: %s", p->name);
    printf("\nStudent Roll No is: %d\n", p->rollno);

    return 0;
}
```

Passing Structures to Functions

The Structure variables can also be passed to a function as arguments both by value and by a reference.

Call by Value

In this method of passing the structures to functions, a copy of the actual arguments is passed to the function.

Program

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

struct student {
   char name[20];
   int rollno;
};

void disp(struct student s1);

int main() {
   struct student stu;

   printf("Enter the Name: ");
   scanf("%s", stu.name);

   printf("Enter the RollNo: ");
   scanf("%d", &stu.rollno);

   disp(stu);

   return 0;
}

void disp(struct student s1) {
   printf("\n****output****");
   printf("\nStudent Name is %s", s1.name);
   printf("\nStudent Rollno is %d", s1.rollno);
}
```

Call by reference

In this method of passing the structures to functions, the address of the actual structure variable is passed.

The address of the structure variable is obtained with the help of address operator'&'.

Program

```
#include<stdio.h>

struct student {
   char name[20];
   int rollno;
};

void disp(struct student *s1);

int main() {

   struct student stu;

   printf("Enter the Name: ");
   scanf("%s", stu.name);

   printf("Enter the RollNo: ");
   scanf("%d", &stu.rollno);

   disp(&stu);
   return 0;
}

void disp(struct student *s1) {
   printf("\n****output****");
   printf("\nStudent Name is %s", s1->name);
   printf("\nStudent Rollno is %d", s1->rollno);
}
```

Unions

Unions are similar to Structures. A union is declared and used in the same ways that a structure.

Union is also a derived data type in C and they are much like structure. The only difference is that in structures, each variable has the separate memory allocation for each element but in union variables use the same memory location.

The general form of union declaration is:-

Unions are defined and declared in the same as structure. The only difference is the declaration is that the keyword union is used instead of struct.

Syntax

union<union_name>

{

datatype member 1;

datatype member 2;

.

.

datatype member n;

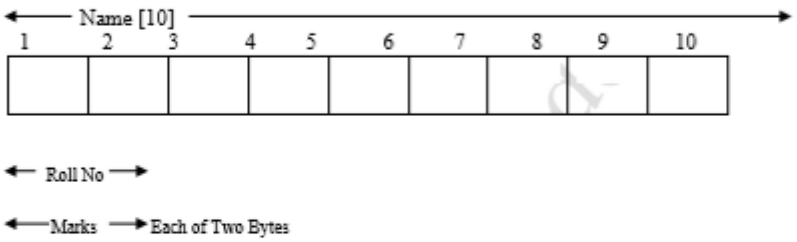
}

var1, var2……,varn;

where union is a required keyword.<name> is the name of the union. member 1,2….n are the individual member declarations.var1,var2,…varn are the union variables.

e.g. A Union with three variables

```
union student
{
char name [10];
int roll_no;
int marks;
} s1;
```



Accessing Union members

The union members can be accessed in the same manner as that of structure member i.e. with the help of dot operator.

The variable of union can be accessed as shown below:-

```
t.v1=15;

t.v2=9.87;

t.v3='P';
```

Features of Union

- Each member of a union shares the same block of memory.
- Nested unions can be used.
- A union can use a structure as its member.

How does Structure Differ from Union?

Structure	Union
1.The keyword Struct is used to define a structure.	1.The keyword union is used to define a union.
2. Structure are used to store different members at different places in memory.	2. Union are used to store different members at same memory location.
3.Each variable member occupied a unique memory space	3. Variables members share the memory space of the largest size variable.
4. Each variable member will be assessed at a time.	4. Only one variable member will be assessed at a time.
5.Syntax of declare a Structure in C is as follow :	5.On other syntax of declare a Union in C is as follow:
struct struct_name { type element1; type element2; . . } variable1, variable2, ...;	union u_name { type element1; type element2; . . } variable1, variable2, ...;