Structures and Union

Structure are use the array chapter, Store similar type of data but if we want to store dissimilar type of data like if data consist of integer, float, character and string then such type of dissimilar data can be stored with the help of structure.

Define:

A structure is Defined struct. struct keyword is used.

Syntax:

struct struct_name{member};

Example: Fees of School

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

Declaring Structure Variables:

The above example only define the 'student' structure with variable of integer and character type. Suppose we want to store id, name and fees of there students, then we need structure variable. to access any member of a structure, we use the member access operator.

Declare structure variables:

```
struct struct_name
{
data type member 1;
data type member 2;
------
data type member n;
}e1, e2, en;
```

Where e1, e2,en are the name of structure variable.

Example:

```
struct student
{
        int id;
        char name[20];
        float fees;
}s1,s2,s3;
```

Accessing Structure Members:

They are two types access structure member:

1. By. (member or dot operator).
2. By-> (structure pointer operator)

Example 1:

```
#include <stdio.h>
#include <string.h>
struct Person {
 char name[50];
 char country[50];
 int pincode;
 float salary;
} s;

int main() {
 strcpy(s.name, "Gyansabha");
 strcpy(s.country, "India");
 s.pincode = 321456;
 s. salary = 150000;
 printf("Your Name: %s\n", s.name);
 printf("Your Country Name.: %s\n", s.country);
 printf("PIN code No.: %d\n", s.pincode);
 printf("Salary: %.2f", s.salary);
 return 0;
}
```

Output:

Your Name: Gyansabha

Your Country Name.: India

PIN code No.: 321456

Salary: 150000.00

Example 2:

```
#include<stdio.h>
struct stud
{
        int roll;
        char name[15];
        int marks;
};
int main()
{
        struct stud s;
        printf("\n Size of Structure: %d", sizeof(s));
        return(0);
}
```

Output:

Size of Structure: 24

Nested Structure

Nested Structure means that one structure has another structure as member variable. One Structure can be declared inside other structure as we declare structure members inside a structure. The structure variables can be a normal structure variable or a pointer variable to access the data.

Syntax:

```
struct tagname
{
        member1;
        member2;
        ------
        member n;
        struct tagname2
        {
                member 1;
                member 2;
                -----
                member n;
        }, var1
} var2;
```

Example:

```
#include<stdio.h>
struct address
{
   char city[20];
   int pin;
   char phone[14];
};
struct employee
{
   char name[20];
   struct address add;
};
void main ()
{
   struct employee emp;
   printf("Enter employee information?\n");
   printf("Enter Name:");
   scanf("%s",emp.name);
   printf("Enter City:");
   scanf("%s",emp.add.city);
   printf("Enter pin:");
   scanf("%d",&emp.add.pin);
   printf("Enter phone:");
   scanf("%s",emp.add.phone);
   printf("\n\n");
   printf("\nPrinting the employee information\n");
```

```
   printf("name: %s\nCity: %s\nPincode: %d\nPhone:
%s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

Output:

```
Enter employee information?
Enter Name: Gyansabha
Enter City: Solapur
Enter pin:413004
Enter phone:9000000000
Printing the employee information
name: Gyansabha
City: Solapur
Pincode: 413004
Phone: 9000000000
```

Array of structure

The array of structure is also known as collection of structure.

Example:

```
#include<stdio.h>
struct student
{
   char name[20];
   int id;
   float marks;
};
void main()
{
   struct student s1,s2,s3;
   int dummy;
   printf("Enter the name, id, and marks of student 1 \n");
   printf("Enter Student Name:");
   scanf("%s", s1.name);
   printf("Enter Student ID:");
   scanf("%d",&s1.id);
   printf("Enter Student Marks:");
   scanf("%2f", &s1.marks);
   scanf("%c",&dummy);
   printf("\nEnter the name, id, and marks of student 2 \n");
   printf("Enter Student Name:");
   scanf("%s", s2.name);
   printf("Enter Student ID:");
   scanf("%d", &s2.id);
```

```
    printf("Enter Student Marks:");
    scanf("%2f", &s2.marks);
    scanf("%c",&dummy);
    printf("\nEnter the name, id, and marks of student 3 \n");
    printf("Enter Student Name:");
    scanf("%s", s3.name);
    printf("Enter Student ID:");
    scanf("%d", &s3.id);
    printf("Enter Student Marks:");
    scanf("%2f", &s3.marks);
    scanf("%c",&dummy);
    printf("\n Show The Mark list in details:\n");
    printf("%s %d %2f\n",s1.name,s1.id,s1.marks);
    printf("%s %d %2f\n",s2.name,s2.id,s2.marks);
    printf("%s %d %2f\n",s3.name,s3.id,s3.marks);
}
```

Output:

```
Enter the name, id, and marks of student 1
Enter Student Name:abc
Enter Student ID:01
Enter Student Marks:30

Enter the name, id, and marks of student 2
Enter Student Name:xyz
Enter Student ID:02
Enter Student Marks:40
```

Enter the name, id, and marks of student 3

Enter Student Name:pqr

Enter Student ID:03

Enter Student Marks:50

Show The Mark list in details:

abc 1 30.00

xyz 2 40.00

pqr 3 50.00

Array within Structure:

Structure is collection of different data types. It can also store an array and we can access array elements within structure and can also display array elements.

Syntax:

```
struct struct_name
{
        datatype var1;
        datatype array[size];
        -------
        ------
        datatype varn;
};
struct struct_name obj;
```

Union

A union user-defined type similar to structure. A union can be define with many members, but only one member can contain a value at any given time. It is provide an efficient way of using the same memory location for multiple-purpose.

- In union, memory is shared among all variables
- The size of union is the largest sizeof its member.
- The keyword union is used to define a union.
- Size of union is equal to the size of largest member.
- Memory allocated is shared by individual members of union.
- In union the address is same for all the members of a union.
- In union, altering the value of any of the member will alter other member value.

Union is declared using the “union” keyword. num 1, num 2, num 3 are individual members of union.

syntax:

```
union [name of union]
  {
    type member1;
    type member2;
    type member3;
  };
```

The body part is terminated with a semicolon.

Example:

```
#include<stdio.h>
#include<string.h>
 union num
{
        int i;
        float f;
        char str[50];
}
int main()
{
        union num data;
        printf("Memory size occupied by data: %d\n", sizeof(data));
        return(0);
}
```

Output:

Memory size occupied by data: 52

Declaration Of Union:

A union is declared in the same way as a structure. It has a list of member, as in the example:

```
union stud
{
        int member 1;
```

float member 2;

} g1, g2;

Declaring union variable is similar to declaring structure variable: union stud g1, g2

Differentiate between Structure and Union

Structure	Union
In structure, memory gets allocated to each and every member separately.	In union, memory is shared among all variables.
The size of structure is the sum of size of all its members.	The size of union is the largest sizeof its member.
The keyword struct is used to define a structure.	The keyword union is used to define a union
Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
In structure, the address each member will be in ascending order.	In union the address is same for all the members of a union.
The size of structure is greater then or equal to the sum of size of its members.	Size of union is equal to the size of largest member.
In structure, altering the value of a member will not affect other members of the structure.	In union, altering the value of any of the member will alter other member values.