Structure and union both are *user-defined* data types in the C programming language.

What is Structure?

Structure is a user-defined data type in C programming language that combines logically related data items of different data types together.

Structure stores the different types of elements i.e heterogeneous elements. The struct keyword is used to define structure.

Suppose that you want to store the data of employee in your C/C++ project, where you have to store the following different parameters:

- Id
- Name
- Department
- Email Address

One way to store 4 different data by creating 4 different arrays for each parameter, such as id[], name[], department[], and email[]. Using array id[i] represents the id of the ith employee. Similarly, name[i] represents the name of ith employee (name).

Array element department[i] and email[i] represent the ith employee's department and email address.

The advantage of using a separate array for each parameter is simple if there are only a few parameters. The disadvantage of such logic is that it is quite difficult to manage employee data. Think about a scenario in which you have to deal with 100 or even more parameters associated with one employee. It isn't easy to manage 100 or even more arrays. In such a condition, a **struct** comes in.

Syntax:

```
struct structure_name
{
data_type member1;

data_type memberN;
};
```

Example

```
struct employee

{
```

```
    int id;
    char name[50];
    char department[50];
    char email[40];
};
```

Declaring structure variable

You can declare structure variable in two ways:-

1. By struct keyword within main() function

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

```
int main()
{
struct Student S1, S2;      //declaring variables of struct Student
return 0;
}
```

2. By declaring variables at the time of defining structure.

```
struct struct_name{
   type element1;
   type element2;
   .
   .
} variable1, variable2, ...;
```

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

Accessing Structure Members

Structure members can be accessed and assigned values in a number of ways. 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 period or member access operator.

For example:

```
#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, "Viraaj");
    /*
    displaying the stored values
    */
    strcpy(s1.branch,"Computer");

    s1.gender='M';
    printf("Name of Student 1: %s\n", s1.name);
    printf("Age of Student 1: %d\n", s1.age);
    printf("Branch of Student 1: %s\n", s1.branch);
     printf("Gender of Student 1: %c\n", s1.gender);

    return 0;
}
```

Output:

```
Name of Student 1: Viraaj
Age of Student 1: 18
Branch of Student 1: Computer
Gender of Student 1: M
```

Simple structure

```
#include<stdio.h>
#include<string.h>
 struct Book{
  int id,price;
  char name[100];
 }b1,b2;
int main()
{

  printf("Enter book 1 id\n");
  scanf("%d",&b1.id);
  printf("Enter book 1 name\n");
  scanf("%s",&b1.name);
    printf("Enter book 1 price\n");
  scanf("%d",&b1.price);

  printf("Enter book 2 id\n");
  scanf("%d",&b2.id);
  printf("Enter book 2 name\n");
  scanf("%s",&b2.name);
    printf("Enter book 2 price\n");
  scanf("%d",&b2.price);
```

```
   printf("Book 1 id=%d\n",b1.id);
   printf("Book 1 Name=%s\n",b1.name);
   printf("Book 1 Price=%d\n",b1.price);

   printf("Book 2 id=%d\n",b2.id);
   printf("Book 2 Name=%s\n",b2.name);
   printf("Book 2 Price=%d\n",b2.price);
   return 0;
}
```

Output:

```
Enter book 1 id
1
Enter book 1 name
abc
Enter book 1 price
100
Enter book 2 id
2
Enter book 2 name
xyz
Enter book 2 price
450
Book 1 id=1
Book 1 Name=abc
```

Book 1 Price=100
Book 2 id=2
Book 2 Name=xyz
Book 2 Price=450

(Array of Structure)

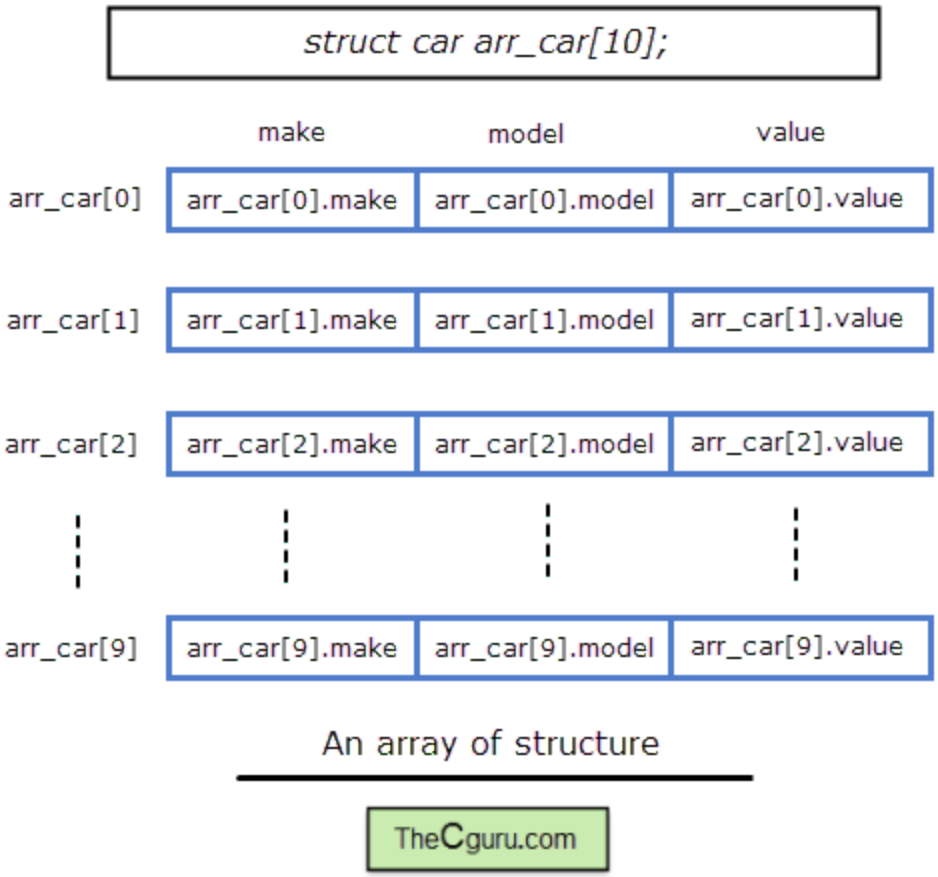
Declaring an array of structure is the same as declaring an array of fundamental types. Since an array is a collection of elements of the same type. In an array of structures, each element of an array is of the structure type.

Let's take an example:

```
struct car
{
   char make[20];
   char model[30];
   int year;
};
```

Here is how we can declare an array of structure car.

```
struct car arr_car[10];
```



Here `arr_car` is an array of `10` elements where each element is of type `struct car`. We can use `arr_car` to store `10` structure variables of type `struct car`. To access individual elements we will use subscript notation (`[]`) and to access the members of each element we will use dot (`.`) operator as usual.

```
arr_stu[0] : points to the 0th element of the
array.
arr_stu[1] : points to the 1st element of the
array.
```

and so on.

```
arr_stu[0].name : refers to the name member of the 0th element of the
array.
arr_stu[0].roll_no : refers to the roll_no member of the 0th element of
the array.
arr_stu[0].marks : refers to the marks member of the 0th element of the
array.
```

Initializing Array of Structures

We can also initialize the array of structures using the same syntax as that for initializing arrays. Let's take an example:

```
struct car
{
    char make[20];
    char model[30];
    int year;
};
struct car arr_car[2] = {
                            {"Audi", "TT", 2016},
                            {"Bentley", "Azure",
2002}
                        };
```

Write a program to accept details of n number of books having fields roll no and name,store them and display them using structure.

```
#include <stdio.h>
struct student
{
	int rollno;
	char name[10];
};
int main ()
{
	int i,size;
	printf("Enter the size");
	scanf("%d",&size);
	struct student st[size];
	printf ("enter records of students\n");
	for (i=0; i<size; i++)
	{
		printf ("\n enter roll no\n");
		scanf ("%d", &st[i].rollno);
		printf ("enter name\n");
		scanf ("%s", &st [i].name);
	}
	printf ("\n student info list:");
	for (i=0; i<size; i++)
	{
printf ("\n roll no: %d, name: %s", st[i].rollno, st[i].name);
	}
```

```
        return 0;
}
```

Output:

```
Enter the size3
enter records of students

 enter roll no
12
enter name
manish

 enter roll no
1
enter name
ram

 enter roll no
2
enter name
shyam

 student info list:
 roll no: 12, name: manish
```

roll no: 1, name: ram
roll no: 2, name: shyam

Write a program to accept details of n number of items having fields code,name,qty and store them and Sort the item data by code parameter in ascending order in structure and display them.
(Integer or numerical sorting)

```
#include<stdio.h>
struct item
{
	int code;
	char name[20];
	int qty;
};
int main()
{

	int n,i,j;
	printf("How many items?");
	scanf("%d",&n);
```

```
struct item it[n],t;
for(i=0;i<n;i++)
{
printf("Item#%d\n",i);
printf("Enter Code,Name,Quantity:");
scanf("%d%s%d",&it[i].code,&it[i].name,&it[i].qty);
}
for(i=0;i<n-1;i++)
{
	for(j=i+1;j<n;j++)
	{
		if(it[i].code>it[j].code)
		{
			t=it[i];
			it[i]=it[j];
			it[j]=t;
		}
	}
}
printf("After sorting on code\n");
for(i=0;i<n;i++)
{
printf("%d\t%s\t%d\n",it[i].code,it[i].name,it[i].qty);
}
return 0;
```

```
}
```
Output:

```
How many items?3
Item#0
Enter Code,Name,Quantity:123 xyz 231
Item#1
Enter Code,Name,Quantity:23  abc  145
Item#2
Enter Code,Name,Quantity:29  efg   456
After sorting on code
23      abc     145
29      efg     456
123     xyz     231
```

Write a C program to create a student structure having fields stud_name and address. Accept the details of 'n' students, rearrange the data in alphabetical order of student name and display it.

```
#include<stdio.h>
#include<string.h>
struct stud
{
char name[50];
char add[50];
```

```
};
int main()
{
	int i,j,n;
	printf("/*How many student records you want to
enter?*/");
	scanf("%d",&n);
	struct stud s[n], t;
	for(i=0;i<n;i++)
	{
		printf("\nEnter Student-%d Details",i+1);
		printf("\n-------------------------\n");
		printf("Enter Name : ");
		scanf("%s",&s[i].name);
		printf("Address : ");
		scanf("%s",&s[i].add);
	}
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(strcmp(s[i].name,s[j].name)>0)
			{
			t=s[i];
			s[i]=s[j];
```

```
                s[j]=t;
            }
        }
        }
        printf("\n\tData after rearrangement");
        printf("\n--------------------------------\n");
        printf("Student Name\tAddress\n");
        printf("-----------------------------------\n");
        for(i=0;i<n;i++)
        {
        printf("\n%s\t\t%s\n",s[i].name,s[i].add);
        }
        return 0;
}
```

Output:

/*How many student records you want to enter?*/2

Enter Student-1 Details

Enter Name : manish

Address : kmc

Enter Student-2 Details

Enter Name : abhay
Address : balkumari

Data after rearrangement

Student Name Address
---

abhay balkumari

manish kmc

// Global Structure Example

#include <stdio.h>

/* Global Structure */
struct Distance

```
{
        int feet;
        float inches;
};

struct Distance d1; /*    Global variable      */

int main ()
{
        struct Distance d2;         /* Local variable */


        d1.feet = 23;
        d1.inches = 7.5;

        d2.feet = 14;
        d2.inches = 2.5;

        printf( "\n %d\'-%f\"", d1.feet, d1.inches );
        printf( "\n %d\'-%f\"", d2.feet, d2.inches );


        return 0;
}
```

```
// Local Structure Example

#include <stdio.h>

int main ()
{
	// Local Structure
	struct Distance
	{
		int feet;
		float inches;
	}d1;

	struct Distance d2; // Local variable

	d1.feet = 23;
	d1.inches = 7.5;

	d2.feet = 14;
	d2.inches = 2.5;

	printf( "\n %d\'-%f\"", d1.feet, d1.inches );
	printf( "\n %d\'-%f\"", d2.feet, d2.inches );
```

```
    return 0;
}
```