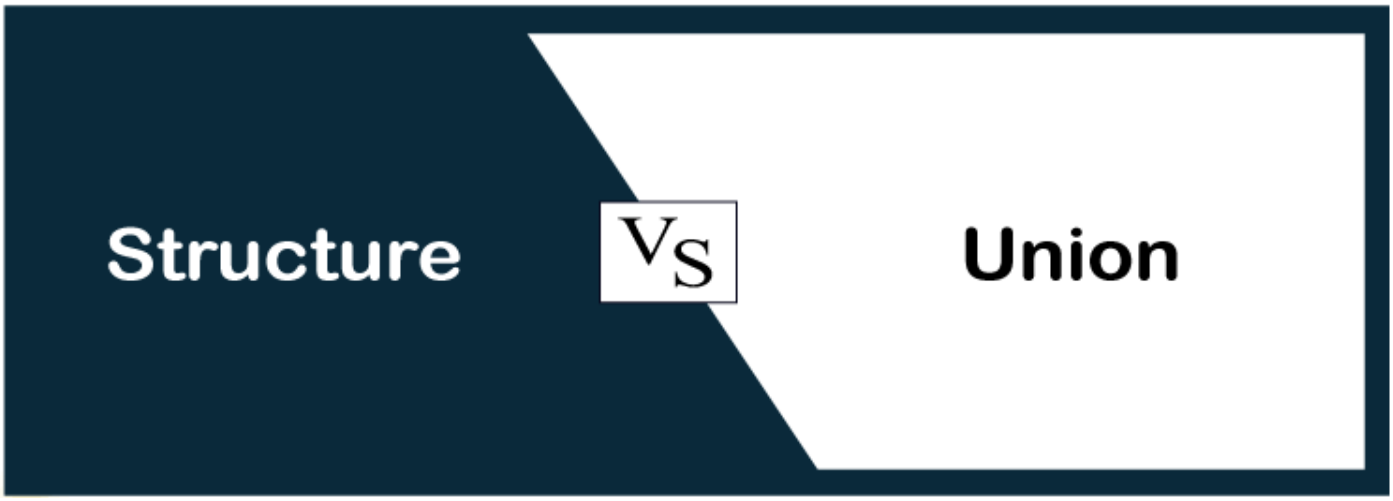
Structure and Union in C

Structure and **union** both are *user-defined* data types in the C/C++ programming language. In this section, we will see what the Structure and Union are; and the differences between them.



What is a structure (struct)?

Structure (struct) is a user-defined data type in a programming language that stores different data types' values together. The **struct** keyword is used to define a structure data type in a program. The struct data type stores one or more than one data element of different kinds in a variable.

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 of struct

```
1. struct [structure_name]
2. {
3.     type member_1;
4.     type member_2;
5.     . . .
6.     type member_n;
7. };
```

Example

```
1. struct employee
2. {
3.     int id;
4.     char name[50];
5.     string department;
6.     string email;
7. };
```

What is a Union?

In "c," programming union is *a user-defined data type* that is used to store the different data type's values. However, in the union, one member will occupy the memory at once. In other words, we can say that the size of the union is equal to the size of its largest data member size. Union offers an effective way to use the same memory location several times by each data member. The **union** keyword is used to define and create a union data type.

Syntax of Union

```
1. union [union name]
2.     {
3. type member_1;
4.     type member_2;
5.     . . .
6.     type member_n;
7.     };
```

Example

```
union employee
{
    string name;
    string department;
    int phone;
    string email;
};
```

Example of Structure and Union showing the difference in C

Let's see an example of structure (struct) and union in c programming, illustrating the various differences between them.

```
// A simple C program showing differences between Structure and Union
#include <stdio.h>
#include <string.h>
// declaring structure
struct struct_example
{
    int integer;
    float decimal;
    char name[20];
};
// declaraing union
union union_example
{
    int integer;
    float decimal;
    char name[20];
};
void main()
{
    // creating variable for structure and initializing values difference six
    struct struct_example stru ={5, 15, "John"};

    // creating variable for union and initializing values
    union union_example uni = {5, 15, "John"};

    printf("data of structure:\n integer: %d\n decimal: %.2f\n name: %s\n", stru.integer, stru.decimal, stru.name);
```

```
27.    printf("\ndata of union:\n integer: %d\n" "decimal: %.2f\n name: %s\n", uni.integer, uni.de
    cimal, uni.name);
28.
29.    // difference five
30.    printf("\nAccessing all members at a time:");
31.    stru.integer = 163;
32.    stru.decimal = 75;
33.    strcpy(stru.name, "John");
34.    printf("\ndata of structure:\n integer: %d\n " "decimal: %f\n name: %s\n", stru.integer, str
    u.decimal, stru.name);
35.
36.    uni.integer = 163;
37.    uni.decimal = 75;
38.    strcpy(uni.name, "John");
39.    printf("\ndata of union:\n integeer: %d\n " "decimal: %f\n name: %s\n", uni.integer, uni.de
    cimal, uni.name);
40.
41.    printf("\nAccessing one member at a time:");
42.    printf("\ndata of structure:");
43.    stru.integer = 140;
44.    stru.decimal = 150;
45.    strcpy(stru.name, "Mike");
46.
47.    printf("\ninteger: %d", stru.integer);
48.    printf("\ndecimal: %f", stru.decimal);
49.    printf("\nname: %s", stru.name);
50.
51.    printf("\ndata of union:");
52.    uni.integer = 140;
53.    uni.decimal = 150;
54.    strcpy(uni.name, "Mike");
55.
56.    printf("\ninteger: %d", uni.integer);
57.    printf("\ndecimal: %f", uni.decimal);
58.    printf("\nname: %s", uni.name);
59.
60.    //difference four
61.    printf("\nAltering a member value:\n");
62.    stru.integer = 512;
63.    printf("data of structure:\n integer: %d\n decimal: %.2f\n name: %s\n", stru.integer, stru.d
    ecimal, stru.name);
```

```
64.     uni.integer = 512;
65.     printf("data of union:\n integer: %d\n decimal: %.2f\n name: %s\n", uni.integer, uni.decim
        al, uni.name);
66.
67.     // difference two and three
68.     printf("\nsizeof structure: %d\n", sizeof(stru));
69.     printf("sizeof union: %d\n", sizeof(uni));
70. }
```

Output

```
data of structure:
 integer: 5
 decimal: 15.00
 name: John

data of union:
 integer: 5
decimal: 0.00
 name:

Accessing all members at a time:
data of structure:
 integer: 163
 decimal: 75.000000
 name: John

data of union:
 integer: 1852337994
 decimal: 17983765624912253034071851008.000000
 name: John

Accessing one member at a time:
data of structure:
integer: 140
decimal: 150.000000
name: Mike
data of union:
integer: 1701538125
decimal: 694811612523029405368 32.000000
name: Mike
Altering a member value:
data of structure:
 integer: 512
 decimal: 150.00
 name: Mike
data of union:
 integer: 512
 decimal: 0.00
 name:

sizeof structure: 28
sizeof union: 20
```

Difference between Structure and Union

Let's summarize the above discussed topic about the Struct and Union in the form of a table that highlight the differences between structure and union:

Struct	Union
The struct keyword is used to define a structure.	The union keyword is used to define union.
When the variables are declared in a structure, the compiler allocates memory to each variables member. The size of a structure is equal or greater to the sum of the sizes of each data member.	When the variable is declared in the union, the compiler allocates memory to the largest size variable member. The size of a union is equal to the size of its largest data member size.
Each variable member occupied a unique memory space.	Variables members share the memory space of the largest size variable.
Changing the value of a member will not affect other variables members.	Changing the value of one member will also affect other variables members.
Each variable member will be assessed at a time.	Only one variable member will be assessed at a time.
We can initialize multiple variables of a structure at a time.	In union, only the first data member can be initialized.
All variable members store some value at any point in the program.	Exactly only one data member stores a value at any particular instance in the program.
The structure allows initializing multiple variable members at once.	Union allows initializing only one variable member at once.
It is used to store different data type values.	It is used for storing one at a time from different data type values.
It allows accessing and retrieving any data member at a time.	It allows accessing and retrieving any one data member at a time.

Comparative advantages and disadvantages of the structure

Below is the list of some advantages and disadvantages of using structure:

Advantages of Structure

- Structure stores more than one data type of the same object together.
- It is helpful when you want to store data of different or similar data types such as name, address, phone, etc., of the same object.
- It makes it very easy to maintain the entire record as we represent complete records using a single name.
- The structure allows passing a whole set of records to any function with the help of a single parameter.
- An array of structures can also be created to store multiple data of similar data types.

Disadvantages of Structure

- If the complexity of the project goes increases, it becomes hard to manage all your data members.
- Making changes to one data structure in a program makes it necessary to change at several other places. So it becomes difficult to track all changes.
- The structure requires more storage space as it allocates memory to all the data members, and even it is slower.
- The structure takes more storage space as it gives memory to all the different data members, whereas union takes only the memory size required by the largest data size parameters, and the same memory is shared with other data members.

Comparative advantages and disadvantages of union

Below is the list of some advantages and disadvantages of using union:

Advantages of Union

- Union takes less memory space as compared to the structure.
- Only the largest size data member can be directly accessed while using a union.
- It is used when you want to use less (same) memory for different data members.
- It allocates memory size to all its data members to the size of its largest data member.

Disadvantages of Union

- It allows access to only one data member at a time.
- Union allocates one single common memory space to all its data members, which are shared between all of them.
- Not all the union data members are initialized, and they are used by interchanging values at a time.