Difference between Structure and Union in C

structures in C

A structure is a user-defined data type available in C that allows to combining data items of different kinds. Structures are used to represent a record.

Defining a structure: To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows:

```
struct [structure name]
{
    member definition;
    member definition;
    ...
    member definition;
};
```

union

A union is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.

Defining a Union: To define a union, you must use the **union** statement in the same way as you did while defining a structure. The union statement defines a new data type with more than one member for your program. The format of the union statement is as follows:

```
 union [union name]
 {
    member definition;
    member definition;
    ...
    member definition;
 };
```

Similarities between Structure and Union

1. Both are user-defined data types used to store data of different types as a single unit.
2. Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
3. Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.

4. A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
5. '.' operator is used for accessing members.

Differences

	STRUCTURE	UNION
Keyword	The keyword **struct** is used to define a structure	The keyword **union** is used to define a union.
Size	When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**	when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**
Memory	Each member within a structure is assigned unique storage area of location.	Memory allocated is shared by individual members of union.
Value Altering	Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Accessing members	Individual member can be accessed at a time.	Only one member can be accessed at a time.
Initialization of Members	Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

```c
// C program to illustrate 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 s={18,38,"geeksforgeeks"};

    // creating variable for union
    // and initializing values
    union union_example u={18,38,"geeksforgeeks"};


    printf("structure data:\n integer: %d\n"
            "decimal: %.2f\n name: %s\n",
            s.integer, s.decimal, s.name);
    printf("\nunion data:\n integeer: %d\n"
             "decimal: %.2f\n name: %s\n",
            u.integer, u.decimal, u.name);


    // difference two and three
    printf("\nsizeof structure : %d\n", sizeof(s));
    printf("sizeof union : %d\n", sizeof(u));

    // difference five
    printf("\n Accessing all members at a time:");
    s.integer = 183;
    s.decimal = 90;
    strcpy(s.name, "geeksforgeeks");

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

    u.integer = 183;
    u.decimal = 90;
    strcpy(u.name, "geeksforgeeks");

    printf("\nunion data:\n integeer: %d\n "
           "decimal: %.2f\n name: %s\n",
        u.integer, u.decimal, u.name);

    printf("\n Accessing one member at time:");

    printf("\nstructure data:");
    s.integer = 240;
    printf("\ninteger: %d", s.integer);

    s.decimal = 120;
    printf("\ndecimal: %f", s.decimal);

    strcpy(s.name, "C programming");
    printf("\nname: %s\n", s.name);

    printf("\n union data:");
    u.integer = 240;
    printf("\ninteger: %d", u.integer);

    u.decimal = 120;
    printf("\ndecimal: %f", u.decimal);

    strcpy(u.name, "C programming");
    printf("\nname: %s\n", u.name);

    //difference four
    printf("\nAltering a member value:\n");
    s.integer = 1218;
    printf("structure data:\n integer: %d\n "
           " decimal: %.2f\n name: %s\n",
           s.integer, s.decimal, s.name);

    u.integer = 1218;
    printf("union data:\n integer: %d\n"
         " decimal: %.2f\n name: %s\n",
        u.integer, u.decimal, u.name);
}
```

Run on IDE

Output:

```
structure data:
 integer: 18
 decimal: 38.00
 name: geeksforgeeks

union data:
 integeer: 18
 decimal: 0.00
 name: ?

sizeof structure: 28
sizeof union: 20

 Accessing all members at a time: structure data:
```

```
 integer: 183
 decimal: 90.00
 name: geeksforgeeks

union data:
 integeer: 1801807207
 decimal: 277322871721159510000000000.00
 name: geeksforgeeks

 Accessing one member at a time:
structure data:
integer: 240
decimal: 120.000000
name: C programming

 union data:
integer: 240
decimal: 120.000000
name: C programming

Altering a member value:
structure data:
 integer: 1218
 decimal: 120.00
 name: C programming
union data:
 integer: 1218
 decimal: 0.00
 name: ?
```

In my opinion, structure is better because as memory is shared in union ambiguity is more.

Quiz on structures and Union

This article is contributed by **Harish Kumar.** If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Practice Tags : C

Article Tags : C Difference Between

Login to Improve this Article

Recommended Posts:

Difference between C structures and C++ structures
Enumeration (or enum) in C
Structures in C
Union in C