C Programming Unions

Unions are quite similar to the structures in C. Union is also a derived type as structure. Union can be defined in same manner as structures just the keyword used in defining union in **union** where keyword used in defining structure was **struct.**

```
union car{
   char name[50];
   int price;
};
```

Union variables can be created in similar manner as structure variable.

```
union car{
   char name[50];
   int price;
}c1, c2, *c3;

OR;

union car{
   char name[50];
   int price;
};
-------Inside Function-----------
union car c1, c2, *c3;
```

In both cases, union variables c1, c2 and union pointer variable c3 of type **union car** is created.

Accessing members of an union

The member of unions can be accessed in similar manner as that structure. Suppose, we you want to access price for union variable c1 in above example, it can be accessed as c1.price. If you want to access price for union pointer variable c3, it can be accessed as (*c3).price or as c3->price.

Difference between union and structure

Though unions are similar to structure in so many ways, the difference between them is crucial to understand. This can be demonstrated by this example:

```
#include <stdio.h>
union job {          //defining a union
   char name[32];
   float salary;
   int worker_no;
}u;
struct job1 {
   char name[32];
   float salary;
   int worker_no;
}s;
int main(){
   printf("size of union = %d",sizeof(u));
   printf("\nsize of structure = %d", sizeof(s));
   return 0;
}
```

Output

```
size of union = 32
size of structure = 40
```

There is difference in memory allocation between union and structure as suggested in above example. The amount of memory required to store a structure variables is the sum of memory size of all members.

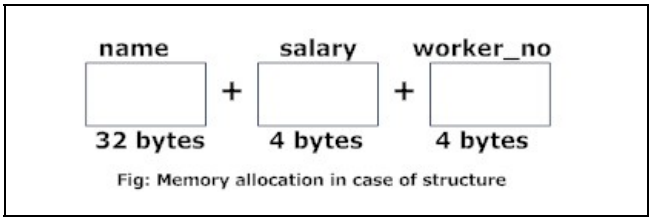

Fig: Memory allocation in case of structure

But, the memory required to store a union variable is the memory required for largest element of an union.

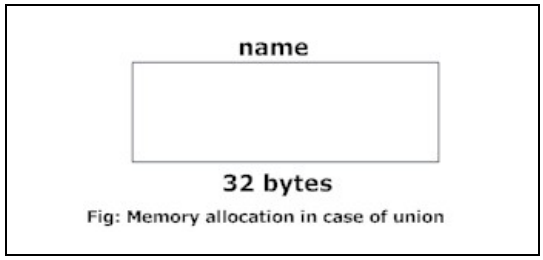

Fig: Memory allocation in case of union

What difference does it make between structure and union?

As you know, all members of structure can be accessed at any time. But, only one member of union can be accessed at a time in case of union and other members will contain garbage value.

```
#include <stdio.h>
union job {
    char name[32];
    float salary;
    int worker_no;
}u;
int main(){
    printf("Enter name:\n");
    scanf("%s",&u.name);
    printf("Enter salary: \n");
    scanf("%f",&u.salary);
    printf("Displaying\nName :%s\n",u.name);
    printf("Salary: %.1f",u.salary);
    return 0;
}
```

Output

```
Enter name
Hillary
Enter salary
1234.23
Displaying
Name: f%Bary
Salary: 1234.2
```

Note: You may get different garbage value of name.

Why this output?

Initially, Hillary will be stored in name and other members of union will contain garbage value. But when user enters value of salary, 1234.23 will be stored in u.salary and other members will contain garbage value. Thus in output, salary is printed accurately but, name displays some random string.

Passing Union To a Function

Union can be passed in similar manner as structures in C programming. Visit this page to learn more about: How structure can be passed to function in C programming?

« Structure & Function

Structure Examples »