Lecture 14

Structure of array, array of structure
Union

Array of structures:

It is possible to store a structure has an array element. i.e., an array in which each element is a structure. Just as arrays of any basic type of variable are allowed, so are arrays of a given type of structure. Although a structure contains many different types, the compiler never gets to know this information because it is hidden away inside a sealed structure capsule, so it can believe that all the elements in the array have the same type, even though that type is itself made up of lots of different types.

The declaration statement is given below.

```
struct struct_name
{
type element 1;
type element 2;
.....
type element n;
}array name[size];
```

Array of structures:

Example:

```
struct student
{
int rollno;
char name[25];
float totalmark;
} stud[100];
```

In this declaration stud is a 100-element array of structures. Hence, each element of stud is a separate structure of type student. An array of structure can be assigned [illegible] other array. So the above structure can hold information of 100 students.

Array of structures:

program:

```
#include <stdio.h>
#include <conio.h>
int main()
{
struct student
{
int rollno;
char name[25];
int totalmark;
}stud[100];
int n,i;
clrscr();
printf("Enter total number of
students\n\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter details of %d-th
student\n",i+1);
printf("Name:\n");
scanf("%s",&stud[i].name);
printf("Roll number:\n");
scanf("%d",&stud[i].rollno);
printf("Total mark:\n");
scanf("%d",&stud[i].totalmark);
}
printf("STUDENTS DETAILS:\n");
for(i=0;i<n;i++)
{
printf("\nRoll
[illegible]
printf("Name:%s\n",stud[i].[illegible]
printf("Totel
mark:%d\n",stud[i].totalmark);
}
getch();
```

Union in C

Union is user defined data type used to stored data under unique variable name at single memory location.

Union is similar to that of stucture. Syntax of union is similar to stucture. But the major difference between structure and union is 'storage.' In structures, each member has its own storage location, whereas all the members of union use the same location. Union contains many members of different types, it can handle only one member at a time.

To declare union data type, "union" ke

Union in C

Syntax:

```
union union_name
{
        <data-type> element 1;
        <data-type> element 2;
        <data-type> element 3;
}union_variable;
```

Example:

```
union techno
{
        int comp_id;
        char nm;
        float sal;
}tch;
```

Union in C

* MEMORY ALLOCATION :

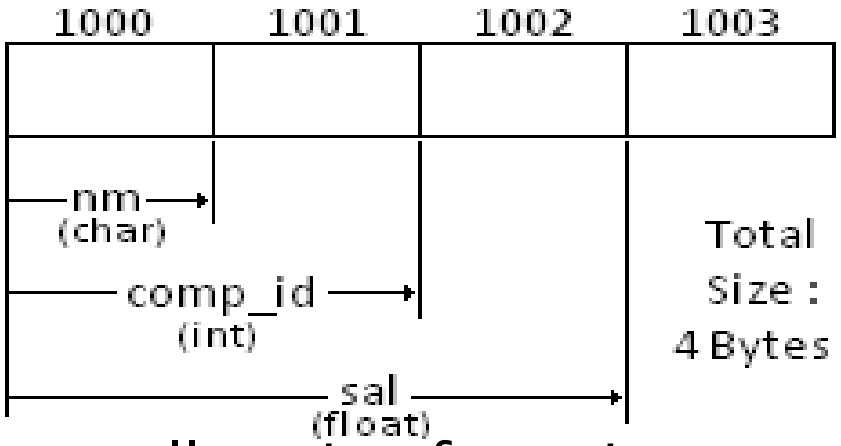

Fig : Memory allocation for union

To access union members, we can use the following syntax.

tch.comp_id

tch.nm

tch.sal

Union in C

/* Program to demonstrate union.

```
#include <stdio.h>
#include <conio.h>

union techno
{
        int id;
        char nm[50];
}tch;


void main()
{
        clrscr();
        printf("\n\t Enter developer id : ");
        scanf("%d", &tch.id);
        printf("\n\n\t Enter developer name : ");
        scanf("%s", tch.nm);
        printf("\n\n Developer ID : %d", tch.id);//Garbage
        printf("\n\n Developed By : %s", tch.nm);
        getch();
}
```

Union in C

OUTPUT

Enter developer id : 101

Enter developer name : technowell

Developer ID : 25972

Developed By : technowell_