SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

COURSE NAME : 19CST101 PROGRAMMING FOR PROBLEM SOLVING

I YEAR/ I SEMESTER

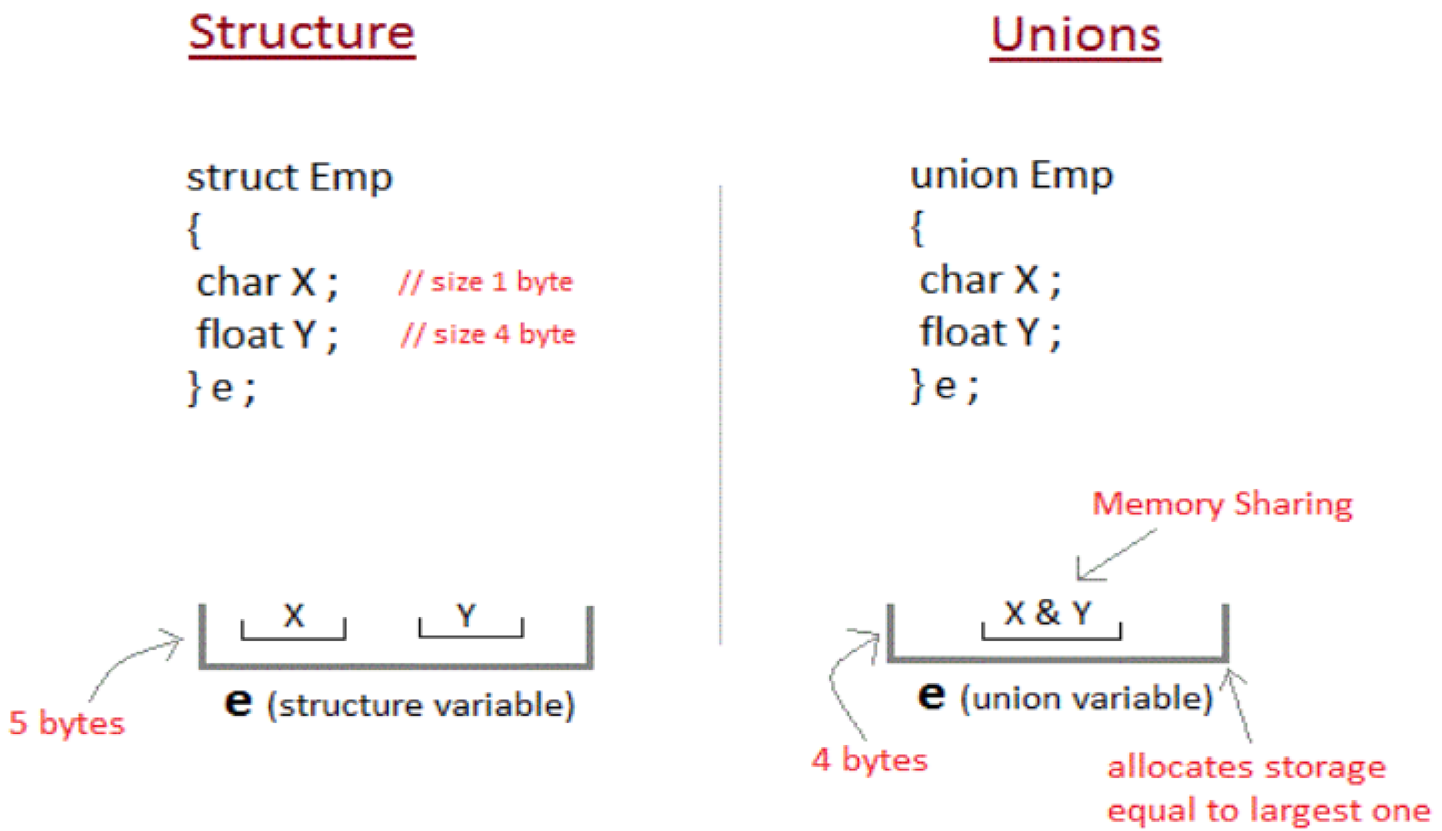
UNIT-V STRUCTURES AND UNIONS

Topic: Unions

Ms.Devi G
Assistant Professor
Department of Computer Science and Engineering

C Unions

Unions are conceptually similar to **structures**. The syntax to declare/define a union is also similar to that of a structure. The only differences is in terms of storage. In **structure** each member has its own storage location, whereas all members of **union** uses a single shared memory location which is equal to the size of its largest data member.



Structure

```
struct Emp
{
 char X ;      // size 1 byte
 float Y ;     // size 4 byte
}e ;
```

Unions

```
union Emp
{
 char X ;
 float Y ;
}e ;
```

C Unions

This implies that although a **union** may contain many members of different types, **it cannot handle all the members at the same time**. A **union** is declared using the `union` keyword.

```
union item
{
    int m;
    float x;
    char c;
}It1;
```

This declares a variable `It1` of type `union item`. This `union` contains three members each with a different data type. However only one of them can be used at a time. This is due to the fact that only one location is allocated for all the `union` variables, irrespective of their size. The compiler allocates the storage that is large enough to hold the largest variable type in the union.

In the union declared above the member `x` requires **4 bytes** which is largest amongst the members for a 16-bit machine. Other members of union will share the same memory address.

C Unions

Accessing a Union Member in C

Syntax for accessing any `union` member is similar to accessing structure members,

```
union test
{
    int a;
    float b;
    char c;
}t;

t.a;    //to access members of union t
t.b;
t.c;
```

Time for an Example

```
#include <stdio.h>

union item
{
    int a;
    float b;
    char ch;
};

int main( )
{
    union item it;
    it.a = 12;
    it.b = 20.2;
    it.ch = 'z';

    printf("%d\n", it.a);
    printf("%f\n", it.b);
    printf("%c\n", it.ch);

    return 0;
}
```

OUTPUT:

```
-26426

20.1999

z
```

- As you can see here, the values of a and b get corrupted and only variable c prints the expected result.
- This is because in union, the memory is shared among different data types.
- Hence, the only member whose value is currently stored will have the memory.
- In the above example, value of the variable c was stored at last, hence the value of other variables is lost.

Thank You!