**Union**

Syntax same like union.

union tag\_name

{

<data\_type> member1;

<data\_type>member2;

} union\_var; // union variable…

**How to make union variable ?**

Union tag\_name var1, var2,…… ….. ;

**How to access?**

union\_var.member\_name;

Example:

#include<stdio.h>

union student

{

Int roll;

char name[10];

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

main()

{

union student stu;

printf(“Enter id:”);

scanf(“%d”,&stu.roll); // var.Meber

printf(“Enter name:”);

scanf(“%d”,&stu.name);

printf(“student details: “);

printf(“id=%d”,stu.roll);

printf(“student name=%s”,stu.name);

}

Union shares one memory location between members. Here roll number ( id) and name both are sharing the same memory location. So the last value will be overlapped on the previous value.

**Output:**

Enter id:101

Enter name:rakan

id=3333 // the last value will be overlapped on the previous value. So output will be garbage value. Union can handle only one member at a time.

student name = rakan

**Difference Between Structure and Union**

**Structure:**

struct student

{

int roll; // size roll member : 2 bytes

char name[10]; //size 10 bytes

} stu; //size of stu variable will be 12 bytes

In the case of structure, The variables are allocated in the sequence.

For example, if member **roll** is stored at memory location 2516 than the member **name** will be at memory location 2158.

So In the case of structure, The variables are allocated in the sequence, (the gap between their location will be the size required by itself that member. Here, 2516 ( roll’s memory address) and 2518 (name’s memory address)

Structure can handle more than one member at a time.

in the case of structure. Each member is located on the different-different location so it will not affect other members.

**Union:**

union student

{

int roll; // required: 2 bytes

char name[10]; //required 10 bytes

// in case of union all member shares the location. Last entered value will overlap to previous member.

} stu; //size of union stu variable will be 10 bytes. It will take maximum size of the members

**printf**(“size:%d”,sizeof(stu); // how to check the size?