FACULTY OF ENGINEERING AND TECHNOLOGY
UNIVERSITY OF LUCKNOW
LUCKNOW

Computer System and Programming in ‘C’

CS-101/201

Er. Zeeshan Ali Siddiqui
Assistant Professor
Deptt. of C.S.E.

STRUCTURE DATA TYPE

Structure-Overview

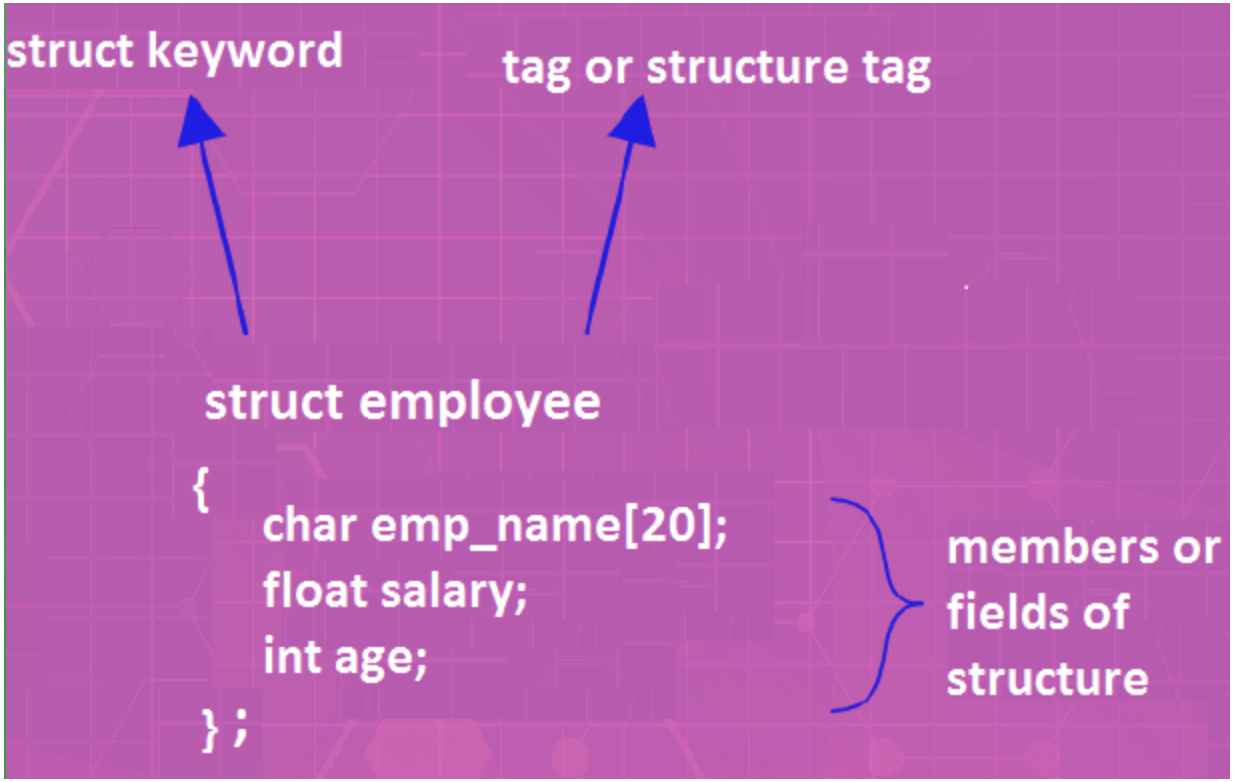
- Structure is an user defined *heterogeneous* data type.
- When one or more variables of different data types are grouped under one name it is known as *structure*.
- Structure is used to group *logically related* different data values under one umbrella.
- It is also known as *compound data-type* that grouped finite set of data field members of different data types.
- It is used to group logically related information together.

Structure-Syntax

```
struct structure_tag_name
{
    data_type member_name1, member_name11, …;
    data_type member_name2, member_name22, …;
    ……
} variable_name;
```

- Note: structure_tag_name is optional.

Structure-Example



Structure variable: Example 1

```
struct student
{
    char name[20];
    int roll_num;
    float per;
} stud1, stud2, stud3;
```

Structure variable: Example 2

```
struct employee
{
    char name[50];
    int emp_id;
    float salary;
} ;
struct employee emp1, emp2;
```

Typedef Keyword

- Typedef is used to give *symbolic* name to an existing data type or user defined data types.

- It is a way to define *alias* to the commands.

- It makes the code easier to understand and change.
- **Syntax**:

```
typedef existing_data_type alias_data_type;
```

- **Example**:

```
typedef int myint;
```

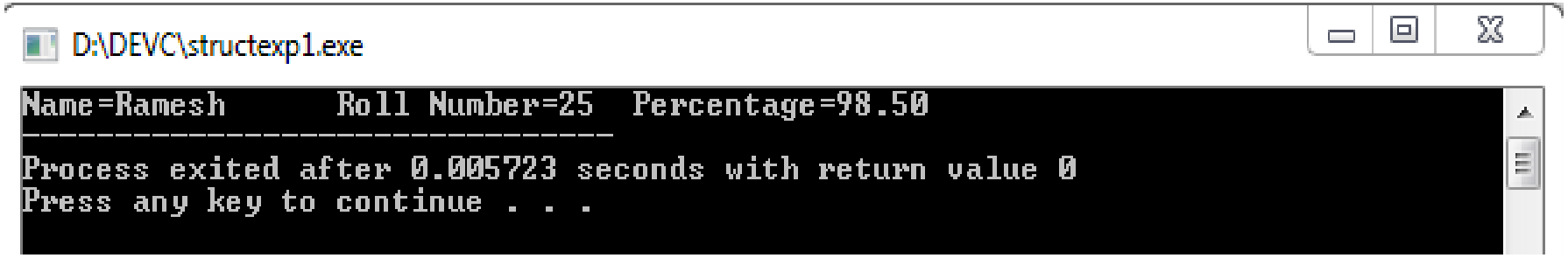
Structure variable: Example 2 Revisit

```
struct employee
{
    char name[50];
    int emp_id;
    float salary;
} ;
typedef struct employee emp;
emp emp1, emp2;
```

Sample Program-1

```c
#include<stdio.h>
struct student
{
    char name[50];
    int roll;
    float per;
};
typedef struct student student;

int main()
{
    student stud1={"Ramesh", 25, 98.5};
    printf("Name=%s\t Roll Number=%d\t Percentage=%.2f",stud1.name,stud1.roll,stud1.per);
    return 0;
}
```

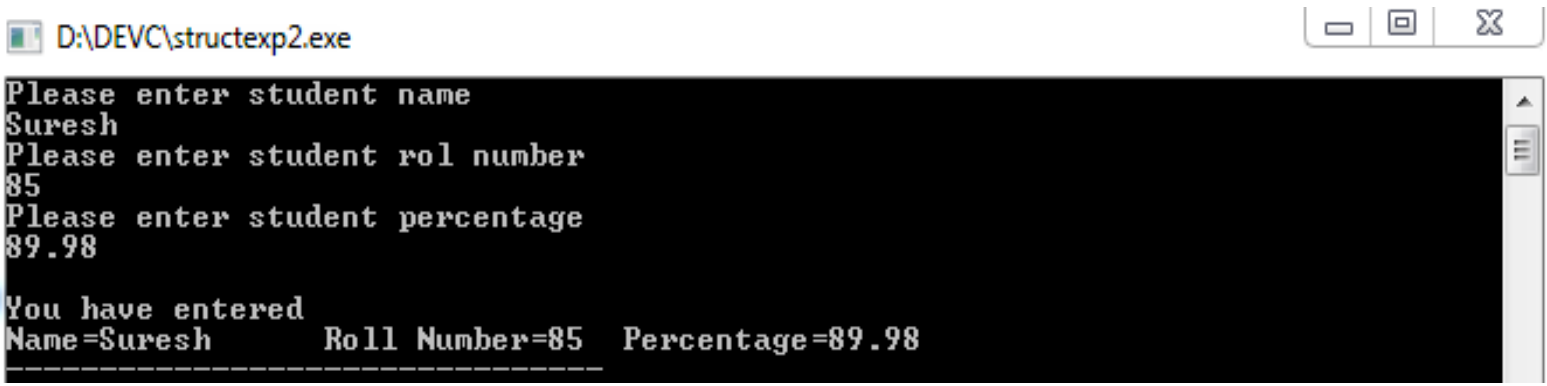
- Output



Sample Program-2

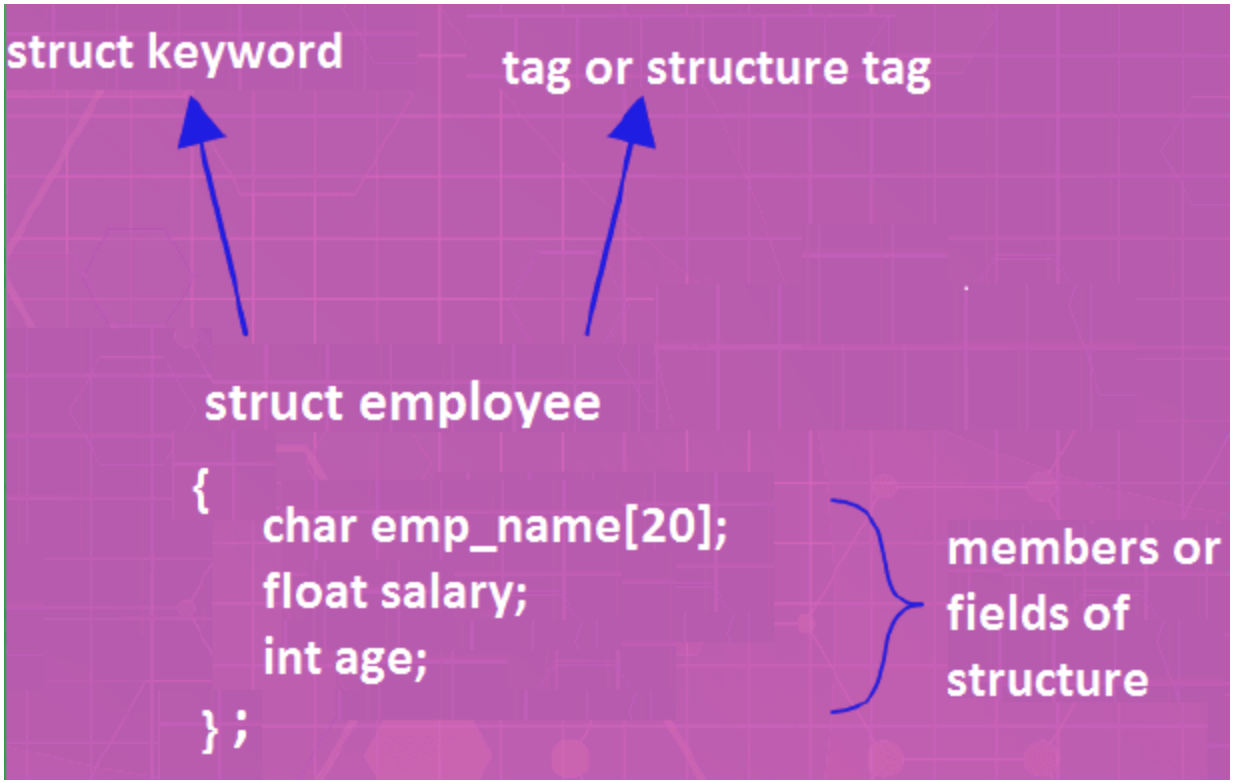
```c
#include<stdio.h>
struct student
{
    char name[50];
    int roll;
    float per;
};
typedef struct student student;
int main()
{
    student stud1;
    printf("Please enter student name\n");
    scanf("%s",&stud1.name);
    printf("Please enter student rol number\n");
    scanf("%d",&stud1.roll);
    printf("Please enter student percentage\n");
    scanf("%f",&stud1.per);
    printf("\nYou have entered\n");
    printf("Name=%s\t Roll Number=%d\t Percentage=%.2f",stud1.name,stud1.roll,stud1.per);
    return 0;
}
```

- Output



Structure: Size calculation 1/2

- Method 1: Simple Addition



- Size = size of emp_name + size of salary + size of age
- Size=20+4+4=28 Bytes
- Note: here the size of data types is given for 64-Bit compiler.

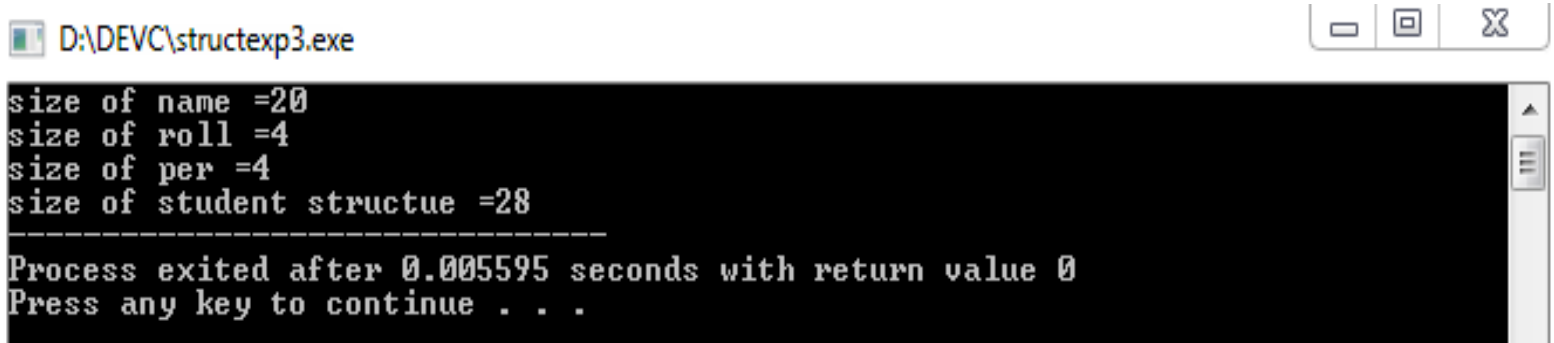
Structure: Size calculation 2/2

- Method 2: Using ***sizeof()*** operator

```c
#include<stdio.h>
struct student
{
    char name[20];
    int roll;
    float per;
};
typedef struct student student;
int main()
{
    student stud1;
    printf("size of student structue =%d",sizeof(stud1));
    return 0;
}
```

- **Output**:

(For 64 bit Compiler)



Structure: Try it yourself

Nested Structures

Program: *Define a structure type, 'struct personal', that would contain person's name, date_of_joining, and salary. date_of_joining will also contain day, month, and year. Using this structure, write a program to read this information for two persons from the keyboard and print the same on the output screen.*

UNION DATA TYPE

Union-Overview

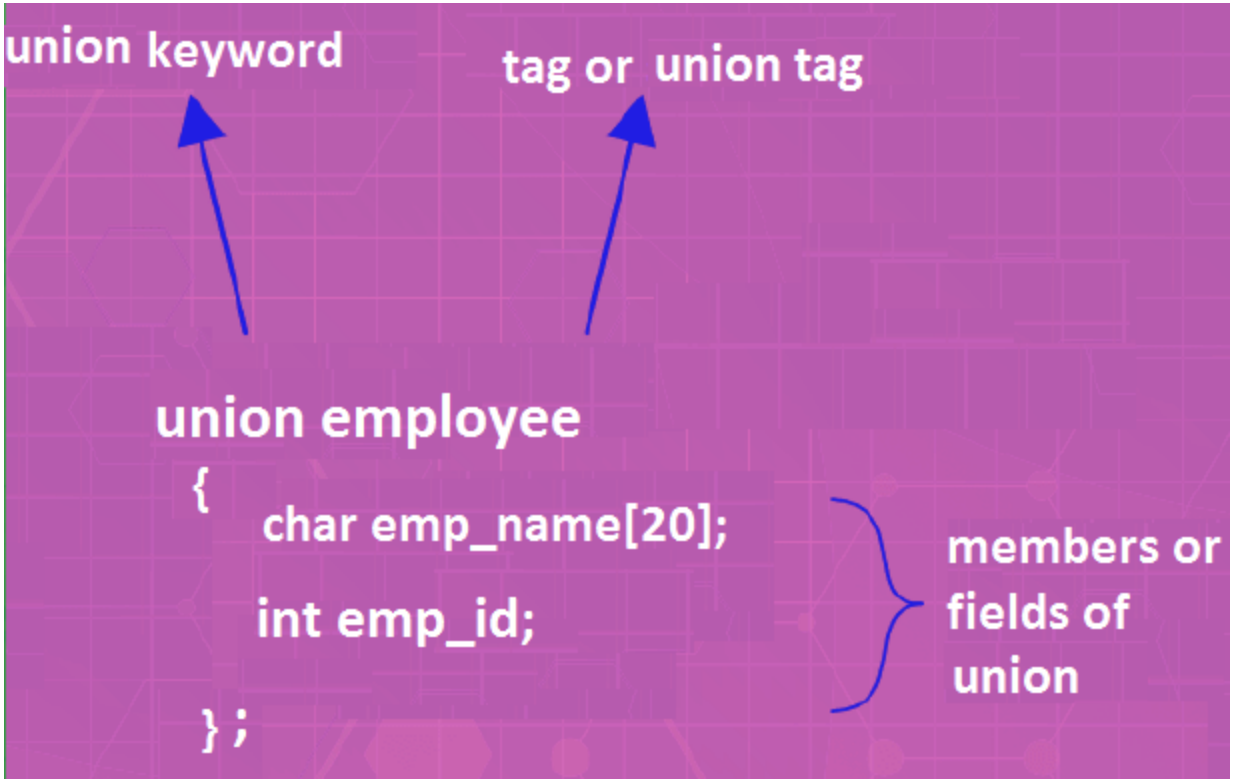
- Union is an user defined *heterogeneous* data type.
- Union is used to group *logically related* different data values under one umbrella.
- In contrast to structures, union allocates *one common storage* space for all its members.
- We can access only *one* member of union at a time.

Union-Syntax

```
union union_tag_name
{
    data_type member_name1, member_name11, …;
    data_type member_name2, member_name22, …;
    ……
} variable_name;
```

- Note: union_tag_name is optional.

Union-Example



Union variable: Example 1

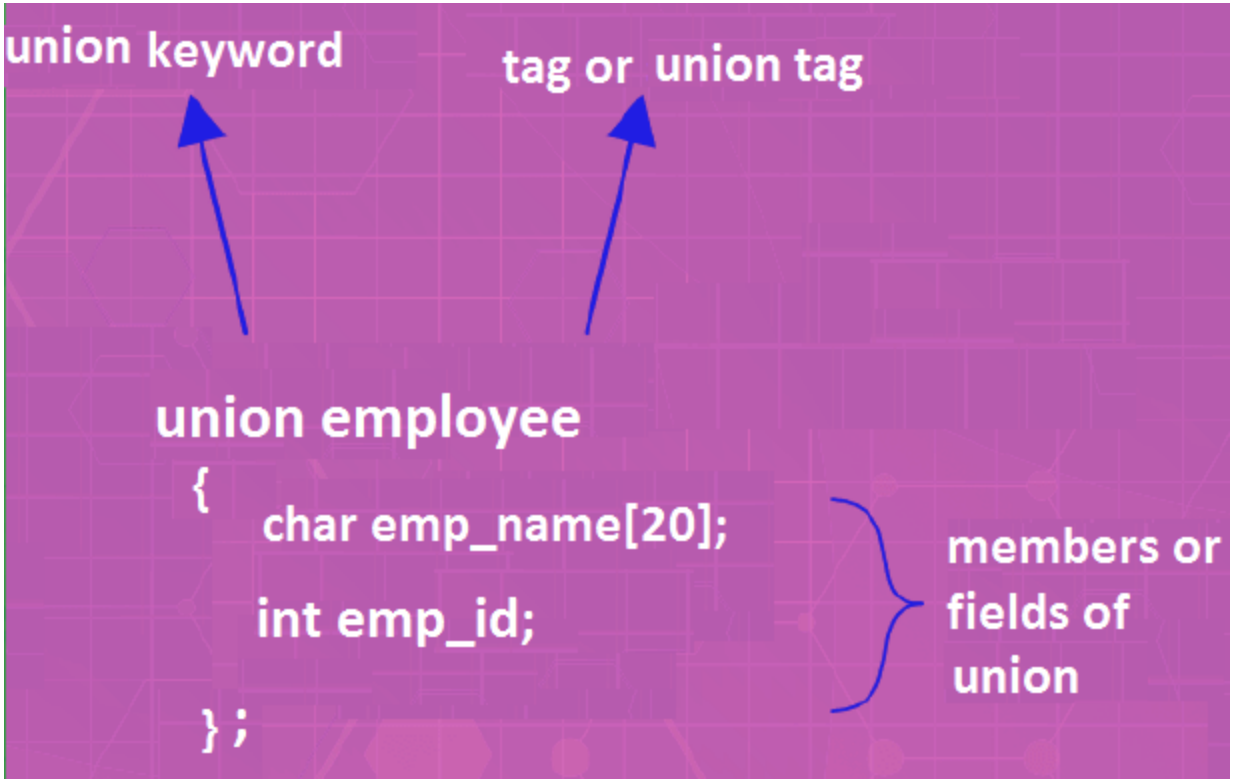
```
union student
{
    char name[20];
    int roll_num;
} stud1, stud2, stud3;
```

Union variable: Example 2

```
union employee
{
   char name[50];
   int emp_id;
} ;
union employee emp1, emp2;
```

Union: Size calculation 1/2

- Method 1:



- Size = max size of (emp_name, emp_id)
- Size=20 Bytes
- Note: here the size of data types is given for 64-Bit compiler.

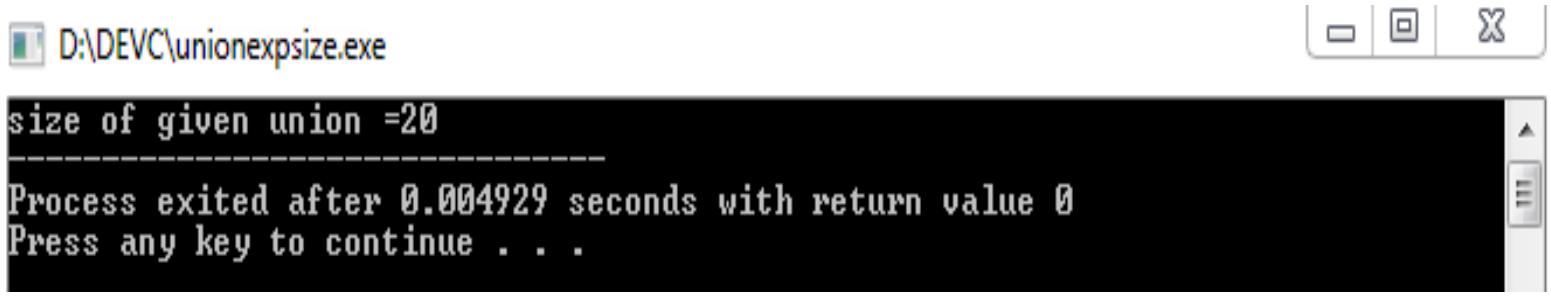
Union: Size calculation 2/2

- Method 2: Using ***sizeof()*** operator

```c
#include<stdio.h>
union student
{
    char name[20];
    int roll;
};
typedef union student student;
int main()
{
    student stud1;
    printf("size of given union =%d",sizeof(stud1));
    return 0;
}
```

- **Output**:

(For 64 bit Compiler)



Union-Usage

- Union can be prove very useful when declared *inside* a structure.

- For example: consider an example in which we want a field of structure to contain either an integer or a string value depending on the requirements of the user.

Sample Program 1/2

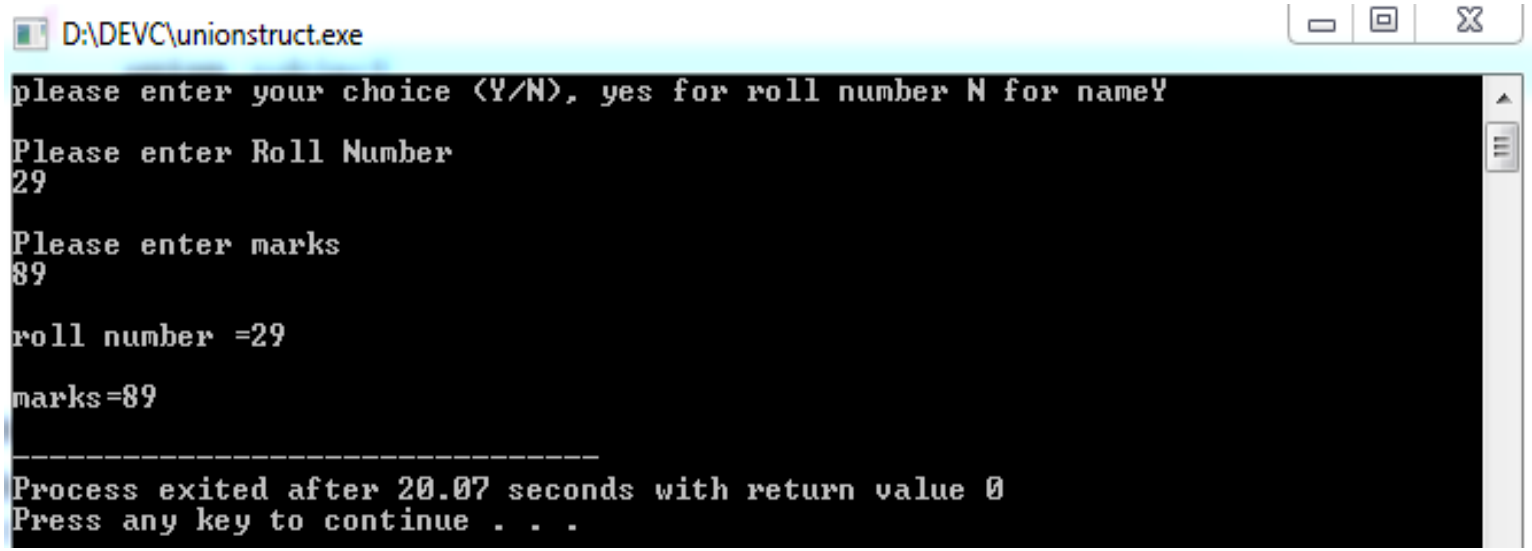
```c
#include<stdio.h>
    int main()
    {
        struct student
        {
            union subject
            {
            int rollnum;
            char name[100];
            }sub;
        int marks;
        };
     char choice;
     typedef struct student student;
     student stud1;
     printf("please enter your choice (Y/N), yes for roll number N for name");
     scanf("%c",&choice);
     if(choice=='Y'||choice=='y')
     {
        printf("\nPlease enter Roll Number\n");
        scanf("%d",&stud1.sub.rollnum);
        printf("\nPlease enter marks\n");
        scanf("%d",&stud1.marks);
     }
```

Sample Program 2/2

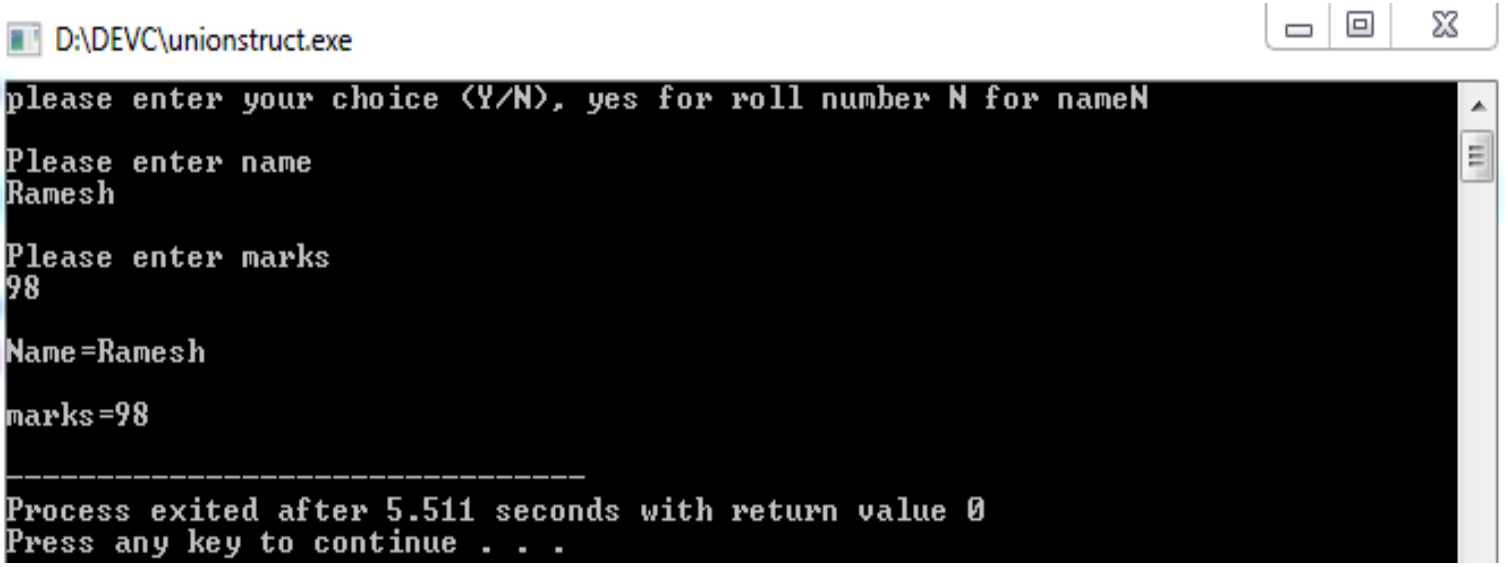
```
else
{
    printf("\nPlease enter name\n");
    scanf("%s",stud1.sub.name);
    printf("\nPlease enter marks\n");
    scanf("\n%d",&stud1.marks);
}

if(choice=='Y'||choice=='y')
 {
    printf("\nroll number =%d\n",stud1.sub.rollnum);
 }
else
{
    printf("\nName=%s\n",stud1.sub.name);
}
    printf("\nmarks=%d\n",stud1.marks);
return 0;
}
```

Sample Program: Output

Output 1:



Output 2:



Explore

Difference between structure and union.

ENUMERATED DATA TYPE

Enumerated Data Type-Overview

- The *enumerated* data type is a user defined data type based on the standard integer data type.
- Enumeration creates new data types to contain values that are not limited to the values that *fundamental* data types may take.
- The enum keyword is used to declare and initialize a *sequence* of integer constants.

Enumerated Data Type-Syntax/Example

Syntax:

enum enumeration_name {identifier1, identifier2, …identifierN};

Note: enumeration_name is optional.

Example 1:

enum DAYS {MONDAY, TUESDAY, WEDNESDAY, THRUSDAY, FRIDAY, SATURDAY, SUNDAY};

Example 2:

enum {MONDAY, TUESDAY, WEDNESDAY, THRUSDAY, FRIDAY, SATURDAY, SUNDAY};

Enumeration variable

Example 1:

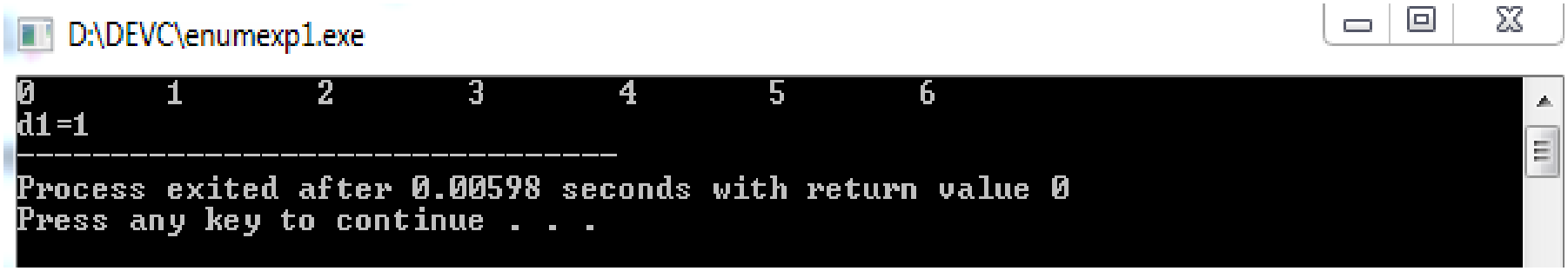
```
enum DAYS {MONDAY, TUESDAY, WEDNESDAY, THRUSDAY, FRIDAY, SATURDAY, SUNDAY} d1,d2;
```

Example 2:

```
enum DAYS {MONDAY, TUESDAY, WEDNESDAY, THRUSDAY, FRIDAY, SATURDAY, SUNDAY} ;
enum DAYS d3,d4;
```

Sample Program

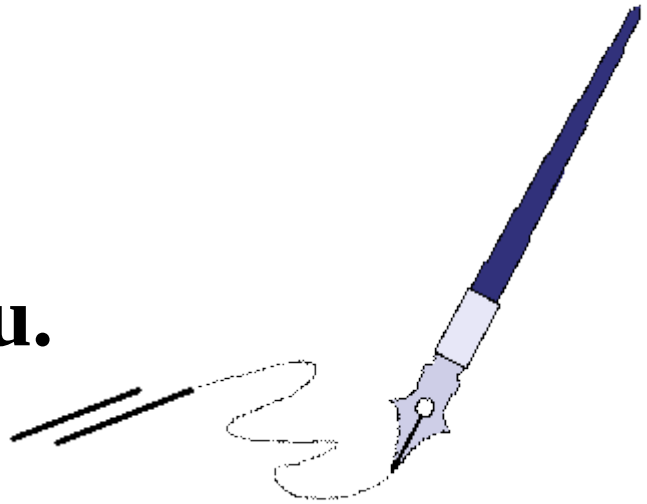
```c
#include<stdio.h>
int main()
{
    enum DAYS {MONDAY, TUESDAY, WEDNESDAY, THRUSDAY, FRIDAY, SATURDAY, SUNDAY};
    typedef enum DAYS days;
    days d1;
    int i;
    for(i=MONDAY;i<=SUNDAY;i++)
    {
        printf("%d\t",i);
    }
    d1=TUESDAY;
    printf("\nd1=%d",d1);
    return 0;
}
```



Exercise

- What is the advantage of using structures?
- Differentiate between structure and union.
- How is a structure name different from a structure variable?
- Differentiate between structure and array.
- Explain the utility of typedef keyword in structures.
- Explain with an example how structures are initialized.
- Explain the utility of unions.
- Define nested structures.
- Write a program using structures to read and display the information about an employee, where employee structure would contain emp_name, emp_id, and emp_sal.

Thank You.



BTQ

BTQ: Brain Teaser Question

What mathematical symbol can be placed between 4 and 9, to get a number greater than 4 and smaller than 9?