

Structure is user define data type. Structure is the collection of variables of different types under a single name for better handling. For example: You want to store the information about person about his/her IdCard Number , Name, Age and Citizen Ship. You can create these information separately but, better approach will be collection of this information under single name because all these information are related to person. Therefore you can define a structure to hold this information.

### Structure Definition in C

Keyword **struct** is used for creating a structure. **struct** define a new data type which is a collection of different type of data.

**Syntax :** **struct** structure\_name  
{  
    DataType variable;  
    DataType variable;  
    :  
};

#### Example:

```
struct Persons  
{  
    int IdCardNo;  
    char name[15];  
    int Age;  
    char CitizenShip[20];  
};
```

### Declaring Structure Variables

It is possible to declare variables of a structure, after the structure is defined. Structure variable declaration is similar to the declaration of variables of any other data types. Structure variables can be declared in following two ways.

#### Declaring Structure variables separately

```
struct Persons  
{  
    int IdCardNo;  
    char name[15];  
    int Age;  
    char CitizenShip[20];  
};  
struct Persons p1,p2;
```

### Declaring Structure Variables with Structure definition

```
struct Persons
{
    int IdCardNo;
    char name[15];
    int Age;
    char CitizenShip[20];
}P1,P2;
```

Here P1 and P2 are variables of structure Persons.

### Accessing Structure Members

Structure members can be accessed and assigned values in number of ways. Structure member has no meaning independently. In order to assign a value to a structure member, the member name must be linked with the structure variable using dot . operator also called period or member access operator.

Any member of a structure can be accessed as;

#### Syntax:

***structure\_variable\_name . member\_name***

#### Example:

p2.Age

(If we want to access Age for variable P2. Then, it can be accessed like this)

### Example of C program:

```
struct Persons
{
    int IdCardNo;
    char name[15];
    int Age;
    char CitizenShip[20];
}P1,P2;
```

P1.Age=20; //P1 is variable of Person type and Age is member of Persons

We can also use **scanf()** to give values to structure members through screen terminal.

```
printf("Enter Person Name ");
scanf(" %s ", P1.name);
printf("Enter Person Age is ");
scanf(" %d ", &P1.Age);
```

©Copy Right

<http://www.sirmasood.com>

Page | 77

### Structure Initialization

Like any other data type, structure variable can also be initialized at compile time.

```
struct Persons
{
    int IdCardNo;
    char name[15];
    int Age;
    char CitizenShip[20];
}P1,P2
```

```
struct Persons P1 = { 1621 , "Asif Kahn ",23, "Karachi" }; //initialization
```

OR

```
struct Persons P1;
```

```
P1.Age = 53; //initialization of each member separately P1.Age = 53;
```

### Example C program (structure):

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

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

```
void main()
```

```
{
```

```
    struct Distance
```

```
    { int feet;
```

```
      int inch;
```

```
    } dist1, dist2, sum;
```

```
clrscr();
```

```
printf("1st distance\n");
```

```
// Input of feet for structure variable dist1
```

```
printf("Enter feet: ");
```

```
scanf("%d", &dist1.feet);
```

```
// Input of inch for structure variable dist1
```

```
printf("Enter inch: ");
```

```
scanf("%d", &dist1.inch);
```

©Copy Right

<http://www.sirmasood.com>

```
printf("2nd distance\n");

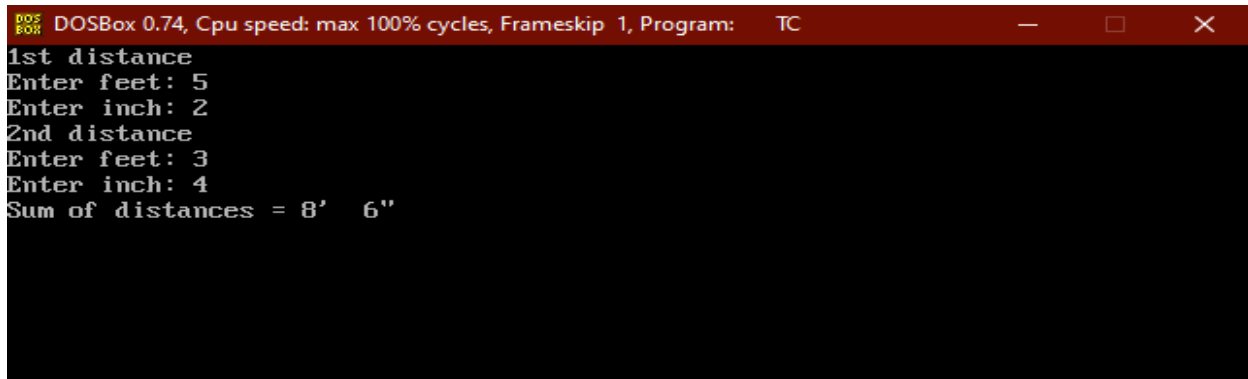
// Input of feet for structure variable dist2
printf("Enter feet: ");
scanf("%d", &dist2.feet);
// Input of feet for structure variable dist2

printf("Enter inch: ");
scanf("%d", &dist2.inch);

sum.feet = dist1.feet + dist2.feet;
sum.inch = dist1.inch + dist2.inch;

if (sum.inch > 12)
{
    //If inch is greater than 12, changing it to feet.
    ++sum.feet;
    sum.inch = sum.inch - 12;
}

// printing sum of distance dist1 and dist2
printf("Sum of distances = %d\ ' %d\ " ", sum.feet, sum.inch);
}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 1, Program: TC
1st distance
Enter feet: 5
Enter inch: 2
2nd distance
Enter feet: 3
Enter inch: 4
Sum of distances = 8' 6"
```

### Nested Structures

Nesting of structures, is also permitted in C language.

©Copy Right

<http://www.sirmasood.com>

**Example :****struct Persons**

```
{
    int IdCardNo;
    char name[15];
    int Age;
    char CitizenShip[20];
```

**struct Students**

```
{
    char[50] ClassName;
    int ObtainMarks;
    float Percentage;
};
};
```

**Structure as function arguments:**

We can pass a structure as a function argument in similar way as we pass any other variable or array.

**Example of C program:**

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

```
struct Students
{
    char name[10];
    int roll;
};
```

```
void show(struct Student st);
```

```
void main()
```

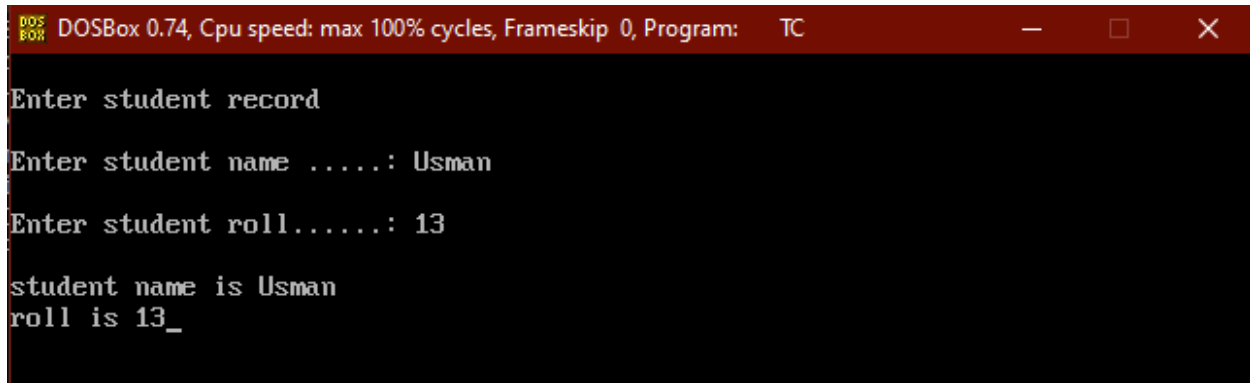
```
{
    struct student std;
    clrscr();
    printf("\nEnter student record\n");
    printf("\nstudent name .....: ");
    scanf("%s",&std.name);
    printf("\nEnter student roll.....");
    scanf("%d",&std.roll);
    show(std);
```

©Copy Right

<http://www.sirmasood.com>

```
    getch();
}

void show(struct Student st)
{
    printf("\nstudent name is %s",st.name);
    printf("\nroll is %d",st.roll);
}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter student record
Enter student name . . . . : Usman
Enter student roll . . . . . : 13
student name is Usman
roll is 13_
```

### Array of Structure in C/C++:

An array of structure in C programming is a collection of different datatype variables, grouped together under a single name. member1, member2 specifies the data items that make up structure.

**Example: array of structure in C/C++ program:**

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct Student{
    int Roll;
    char Name[30];
    float Percentage;
};
void main()
{
    int i;
    char temp[30];
    struct Student StudentsRecord[5];

    clrscr();
    for(i=0; i<4; i++)
    {
```

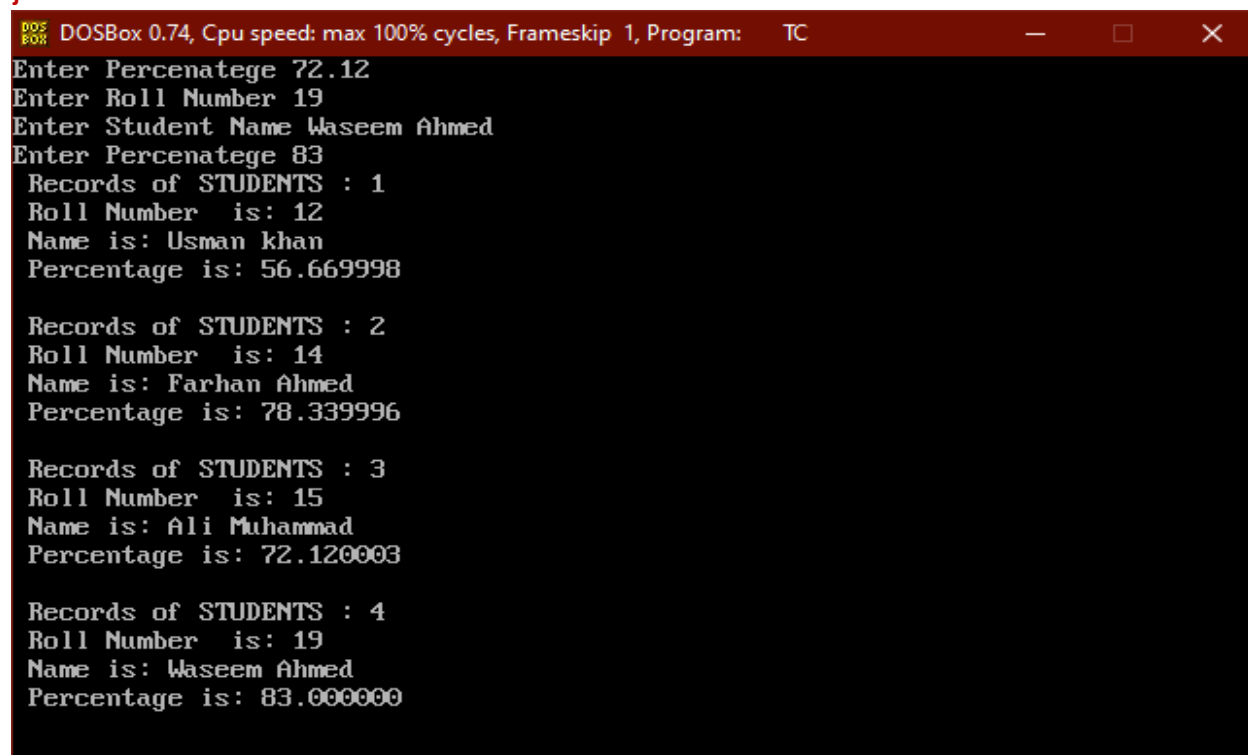
©Copy Right

<http://www.sirmasood.com>

Page | 81

```
printf("Enter Roll Number ");
gets(temp);
StudentsRecord[i].Roll = atoi(temp);
printf("Enter Student Name ");
gets(StudentsRecord[i].Name);
printf("Enter Percentage ");
gets(temp);
StudentsRecord[i].Percentage= atof(temp);
}

for(i=0; i<4; i++)
{
printf(" Records of STUDENTS : %d \n", i+1);
printf(" Roll Number is: %d \n", StudentsRecord[i].Roll);
printf(" Name is: %s \n", StudentsRecord[i].Name);
printf(" Percentage is: %f\n\n",StudentsRecord[i].Percentage);
}
}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 1, Program: TC
Enter Percentage 72.12
Enter Roll Number 19
Enter Student Name Waseem Ahmed
Enter Percentage 83
Records of STUDENTS : 1
Roll Number is: 12
Name is: Usman khan
Percentage is: 56.669998

Records of STUDENTS : 2
Roll Number is: 14
Name is: Farhan Ahmed
Percentage is: 78.339996

Records of STUDENTS : 3
Roll Number is: 15
Name is: Ali Muhammad
Percentage is: 72.120003

Records of STUDENTS : 4
Roll Number is: 19
Name is: Waseem Ahmed
Percentage is: 83.000000
```

### Union:

Union is a user defined datatype in C programming language. It is a collection of variables of different datatypes in the same memory location. We can define a union with many members, but at a given point of time only one member can contain a value. Unions can be very handy when you need to talk to peripherals through some memory mapped registers.

©Copy Right

### Difference between structure and union

The main difference between structure and a union is that.

Structure allocate enough space to store all of the fields in the struct. The first one is stored at the beginning of the struct, the second is stored after that, and so on. Unions only allocate enough space to store the largest field listed, and all fields are stored at the same space.

### Syntax for Declaring a C union

Syntax for declaring a union is same as that of declaring a structure except the keyword struct.

```
union union_name
{
    datatype field_name;
    datatype field_name;
    // more variables
};
```

**Note :** Size of the union is the size of its largest field because sufficient number of bytes must be reserved to store the largest sized field.

To access the fields of a union, use **dot(.)** operator , the variable name followed by dot operator followed by field name.

Example

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

```
union Data {
    int i;
    float f;
    char str[20];
};
```

```
void main( )
{
    union Data data;
    printf( "Memory size occupied by data : %d\n", sizeof(data));
}
```

Above, a variable of Data type can store an integer, a floating-point number, or a string of characters. It means a single variable, i.e., same memory location, can be used to store multiple types of data. You can use any built-in or user defined data types inside a union based on your requirement.

The memory occupied by a union will be large enough to hold the largest member of the union. For example, in the above example, Data type will occupy 20 bytes of memory space because this is the maximum space which can be occupied by a character string. The following example displays the total memory size occupied by the above union and result will be show

```
Memory size occupied by data : 20
```

©Copy Right

<http://www.sirmasood.com>

Page | 83



## Exercise

### Theory Questions

- 1) What is Structure in C/C++;
- 2) How you can differentiate between array and structure.
- 3) What is nested structure.

### Practical Questions

- 1) Write a program to store record of 3 employees, their *Emp\_Name*, *Emp\_code*, *Emp\_Salary* and *Emp\_Designation* using structure.
- 2) Write a program to maintain the library record for 10 books with *book\_name*, *authors\_name*, and *edition*, and *price* of the books. Using by array of structure.
- 3) How you can differentiate between the structure and union data types.

### Objective MCQ's

- 1) Array elements must all be of the same data types where structure members can be of different data types.
  - a) True
  - b) False
- 2) What is a structure in C language?
  - a) Structure is a collection of elements that can be of same data type.
  - b) A structure is a collection of elements that can be of different data type.
  - c) Elements of a structure are called members.
  - d) All the above
- 3) What is the size of a C structure?
  - a) Structure is always 128 bytes.
  - b) Size of C structure is the total bytes of all elements of structure.
  - c) Size of C structure is the size of largest element.
  - d) None of the above
- 4) Choose a correct statement about C structures.
  - a) Structure elements can be initialized at the time of declaration.
  - b) Structure members cannot be initialized at the time of declaration
  - c) Only integer members of structure can be initialized at the time of declaration
  - d) None of the above

5) What is the output of C program?

```
void main()
{
    struct car
    {
        int size;
        char color[10];
    }Car1, Car2;
    Car1.size=10;
    Car2 = car1;
    printf("boat2=%d",Car2.size);
}
```

- a) Car2=0
  - b) Car2=-1
  - c) Car2=10
  - d) Compiler error
- 6) Choose a correct statement about C structure elements.
- a) Structure elements are stored on random free memory locations
  - b) structure elements are stored in register memory locations
  - c) structure elements are stored in contiguous memory locations
  - d) None of the above.
- 7) Which operator connects the structure name to its member name?
- a) -
  - b) &
  - c) .
  - d) <-
- 8) size of union is size of the longest element in the union
- a) Yes
  - b) No
  - c) May Be
  - d) Can't Say
- 9) What is the similarity between a structure, union and enumeration?
- a) All of them let you define new values
  - b) All of them let you define new data types
  - c) All of them let you define new pointers
  - d) All of them let you define new structures