Structure

- **What is Structure**
- Structure in c is a user-defined data type that enables us to store the collection of different data types.
- Each element of a structure is called a member. Structures can simulate the use of classes and templates as it can store various information.
- The ,struct keyword is used to define the structure. Let's see the syntax to define the structure in c.

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memeberN;
};
```

Let's see the example to define a structure for an entity employee in c.

```
struct employee
{   int id;
    char name[20];
    float salary;
};
```

- **Declaring structure variable**
- We can declare a variable for the structure so that we can access the member of the structure easily. There are two ways to declare structure variable:
- By struct keyword within main() function
- By declaring a variable at the time of defining the structure.

- **1st way:**
- Let's see the example to declare the structure variable by struct keyword. It should be declared within the main function.

```
struct employee
{   int id;
   char name[50];
   float salary;
};
```

- Now write given code inside the main() function.
- struct employee e1, e2;
- The variables e1 and e2 can be used to access the values stored in the structure. Here, e1 and e2 can be treated in the same way as the objects in C++ and Java.

2nd way:

Let's see another way to declare variable at the time of defining the structure.

```
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;
```

- **Which approach is good**
- If number of variables are not fixed, use the 1st approach. It provides you the flexibility to declare the structure variable many times.
- If no. of variables are fixed, use 2nd approach. It saves your code to declare a variable in main() function.

- **Accessing members of the structure**
- There are two ways to access structure members:
- By . (member or dot operator)
- By -> (structure pointer operator)
- Let's see the code to access the id member of p1 variable by. (member) operator.

```
p1.id
```

C Structure example

Let's see a simple example of structure in C language.

```
#include<stdio.h>
#include <string.h>
struct employee
{   int id;
    char name[50];
}e1;  //declaring e1 variable for structure
```

```
int main( )
{
 //store first employee information
   e1.id=101;
   strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array
   //printing first employee information
   printf( "employee 1 id : %d\n", e1.id);
   printf( "employee 1 name : %s\n", e1.name);
return 0;
}
```

Output:

```
employee 1 id : 101
employee 1 name : Sonoo Jaiswal
```

- Let's see another example of the structure in C language to store many employees information.

```
#include<stdio.h>
#include <string.h>
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;  //declaring e1 and e2 variables for structure
```

```
int main( )
{
   //store first employee information
   e1.id=101;
   strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array
   e1.salary=56000;
```

```
//store second employee information
   e2.id=102;
   strcpy(e2.name, "James Bond");
   e2.salary=126000;
//printing first employee information
   printf( "employee 1 id : %d\n", e1.id);
 printf( "employee 1 name : %s\n", e1.name);
 printf( "employee 1 salary : %f\n", e1.salary);
```

```
//printing second employee information
   printf( "employee 2 id : %d\n", e2.id);
   printf( "employee 2 name : %s\n", e2.name);
   printf( "employee 2 salary : %f\n", e2.salary);
   return 0;
}
```

Output:

```
employee 1 id : 101
employee 1 name : Sonoo Jaiswal
employee 1 salary : 56000.000000
employee 2 id : 102
employee 2 name : James Bond
employee 2 salary : 126000.000000
```

- **typedef in C**
- The typedef is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program
- It behaves similarly as we define the alias for the commands. In short, we can say that this keyword is used to redefine the name of an already existing variable.
- **Syntax of typedef**
- typedef <existing_name> <alias_name>

- In the above syntax, 'existing_name' is the name of an already existing variable while 'alias name' is another name given to the existing variable.
- For example, suppose we want to create a variable of type unsigned int, then it becomes a tedious task if we want to declare multiple variables of this type.
- To overcome the problem, we use a typedef keyword.

- typedef unsigned int unit;
- In the above statements, we have declared the unit variable of type unsigned int by using a typedef keyword.
- Now, we can create the variables of type unsigned int by writing the following statement:
- unit a, b;
- instead of writing:
- unsigned int a, b;

- Till now, we have observed that the typedef keyword provides a nice shortcut by providing an alternative name for an already existing variable.
- This keyword is useful when we are dealing with the long data type especially, structure declarations.
- Let's understand through a simple example.

```
#include <stdio.h>
int main()
{
typedef unsigned int unit;
unit i,j;
i=10;
j=20;
printf("Value of i is :%d",i);
printf("\nValue of j is :%d",j);
return 0;
```

Output

```
Value of i is :10
Value of j is :20
```

Using typedef with structures

Consider the below structure declaration:

```
struct student
{
char name[20];
int age;
};
struct student s1;
```

In the above structure declaration, we have created the variable of student type by writing the following statement:

struct student s1;

The above statement shows the creation of a variable, i.e., s1, but the statement is quite big.

To avoid such a big statement, we use the typedef keyword to create the variable of type student.

```
struct student
{
char name[20];
int age;
};
typedef struct student stud;
stud s1, s2;
```

In the above statement, we have declared the variable stud of type struct student. Now, we can use the stud variable in a program to create the variables of type struct student.

The above typedef can be written as:

```
typedef struct student
{
char name[20];
int age;
} stud;
stud s1,s2;
```

From the above declarations, we conclude that typedef keyword reduces the length of the code and complexity of data types. It also helps in understanding the program.