Structures in C

Arrays are used to store similar type of data. Have you ever thought if there is any way to store **dissimilar** data? In C, there are cases where we need to store multiple attributes of an entity. It is not necessary that an entity has all the information of one type only. It can have different attributes of different data types. For example, an entity **Student** may have its name (string), roll number (int), marks (float). To store such type of information regarding an entity student, we have the following approaches:

Either we have to Construct individual arrays for storing names, roll numbers, and marks or to Use a special data structure to store the collection of different data types, obvsly the second approach is better.

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 ca; 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.

Just as we declare variables of type int, char etc; we can declare variables of structure as well.

Suppose, we want to store the roll no., name and phone number of three students. For this, we will define a structure of name 'student' (as declared above) and then declare three variables, say 'p1', 'p2' and 'p3' (which will represent the three students respectively) of the structure 'student'. The structure will look like this,

struct student
{
  int roll_no;
  char name[30];
  int phone_number;
};
main()
{
  struct student p1, p2, p3;
}
	struct student
	{
	  int roll_no;
	  char name[30];
	  int phone_number;
	}p1, p2,
		struct employee
		{ **int** id;
		    char name[50];
		    float salary;
		};

Here, p1, p2 and p3 are the variables of the structure 'student'. We can also declare structure variables at the time of defining structure as follows.

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.

Suppose, we want to assign a roll number to the first student. For that, we need to access the roll number of the first student. We do this by writing

```
p1.roll_no = 1;
```

This means that use dot (.) to use variables in a structure. *p1.roll_no* can be understood as roll_no of p1.

Example 1

```c
#include <stdio.h>
#include <string.h>
int main()
{
  struct student
  {
    int roll_no;
    char name[30];
    int phone_number;
  };
  struct student p1 = {1,"Brown",123443};
  struct student p2, p3;
  p2.roll_no = 2;
  strcpy(p2.name,"Sam");
  p2.phone_number = 1234567822;
  p3.roll_no = 3;
  strcpy(p3.name,"Addy");
  p3.phone_number = 1234567844;
  printf("First Student\n");
  printf("roll_no : %d\n", p1.roll_no);
  printf("name : %s\n", p1.name);
  printf("phone_number : %d\n", p1.phone_number);
  printf("Second Student\n");
  printf("roll_no : %d\n", p2.roll_no);
  printf("name : %s\n", p2.name);
  printf("phone_number : %d\n", p2.phone_number);
  printf("Third Student\n");
  printf("roll_no : %d\n", p3.roll_no);
  printf("name : %s\n", p3.name);
  printf("phone_number : %d\n", p3.phone_number);
  return 0;
}
```

Example 2:

If we want to store the information of n student then we have to use array of structures. Here, `stud[0]` stores the information of first student, `stud[1]` for the second and so on.

```c
#include <stdio.h>
struct student
{
  int roll_no;
  char name[30];
  int phone_number;
};
int main()
{
  struct student stud[5];
  int i;
  for(i=0; i<4; i++)
    {
      printf("Student %d\n",i+1);
      printf("Enter roll no. :\n");
      scanf("%d", &stud[i].roll_no);
      printf("Enter name :\n");
      scanf("%s",stud[i].name);
      printf("Enter phone number :\n");
      scanf("%d", &stud[i].phone_number);
    }
  for(i=0; i<4; i++)
    {
      printf("Student %d\n",i+1);
      printf("Roll no. : %d\n", stud[i].roll_no);
      printf("Name : %s\n", stud[i].name);
      printf("Phone no. : %d\n", stud[i].phone_number);
    }
  return 0;
}
```

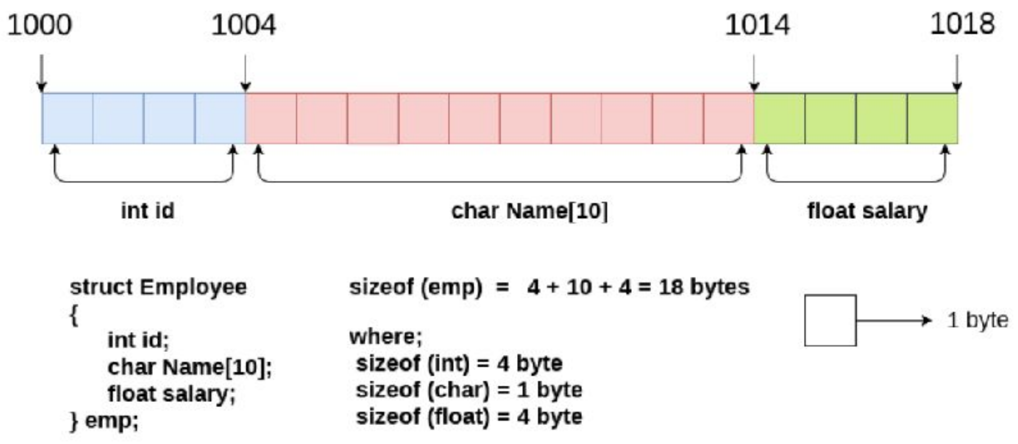
Accessing members of the structure

There are two ways to access structure members:

1. By. (member or dot operator)
2. By -> (structure pointer operator)

Examples 3

```
#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;
}
```

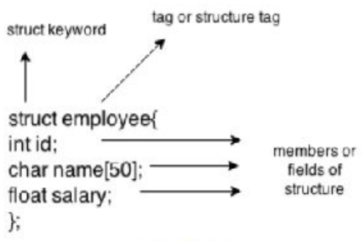


```
struct Employee
{
     int id;
     char Name[10];
     float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
sizeof (int) = 4 byte
sizeof (char) = 1 byte
sizeof (float) = 4 byte

Here, **struct** is the keyword; **employee** is the name of the structure; **id**, **name**, and **salary** are the members or fields of the structure. Let's understand it by the diagram given below:



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