Structure

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:

- Construct individual arrays for storing names, roll numbers, and marks.
- Use a special data structure to store the collection of different data types.
- Let's look at the first approach in detail.

Structure

Syntax:

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

Declaring Structure

First Way:

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

Second Way:

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

Example of Structure

```
#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;
}
```