Structures in C

A structure is a user-defined data type that groups together variables of different data types under a single name. Structures are used to represent a record, and they are particularly useful when you need to store a collection of data that may not be of the same type.

Here's an example of how to define a structure in C:

```c
struct Student {


    char name[50];
    int roll;
    float marks;
};
```

In this example, we've defined a structure called `Student` that has three members: `name`, `roll`, and `marks`. The `name` member is an array of characters, while `roll` is an integer and `marks` is a floating-point number.

To use a structure, you need to create an instance of it. Here's how you can do that:

```c
struct Student s1;
```

In this example, we've created an instance of the `Student` structure called `s1`. You can access the members of the structure using the dot (`.`) operator. Here's an example:

```c
strcpy(s1.name, "John Doe");
s1.roll = 123;
s1.marks = 95.5;
```

In this example, we've assigned values to the members of the `s1` instance of the `Student` structure. We've used the `strcpy` function to copy the string `"John Doe"` into the `name` member, and we've assigned values to the `roll` and `marks` members directly.

You can also create an array of structures. Here's an example:

```c
struct Student s[10];
```

In this example, we've created an array of 10 instances of the `Student` structure. You can access individual instances using array indexing, and you can access the members of each instance using the dot (`.`) operator.