PROGRAMMING

IT 112: Introduction to Programming

Dr. Manish Khare
Dr. Bakul Gohel

Lecture 22

STRUCTURES

- Structure
- Structure in Structure
- Structure with Array
- Structure with Function
- Structure with Pointer
- Structure with Pointer and Array
- Structure with Pointer and Function

Structures

- If we want to store a single value in C we can use any of the fundamental data types like int, float etc.
- And if we want to store values of same data type under one variable name then we take help of an array.
- But we can't store different data type values that are logically related using simple variable or arrays. For this we take help of **structure** denoted by the struct keyword.
- Structure is a user-defined datatype in C language which allows us to combine data of different types together.
- Structure helps to construct a complex data type which is more meaningful.

- **For example:** If I have to write a program to store Student information, which will have Student's name, age, branch, permanent address, father's name etc, which included string values, integer values etc, how can I use arrays for this problem, I will require something which can hold data of different types together.

- In structure, data is stored in form of **records**.

Syntax of Structure

- Following is the syntax of a structure.

```
struct tagName
        {
        dataType member1;
        dataType member2;
        };
```

- As you can see in the syntax, we start with the struct keyword, then it's optional to provide your structure a name, we suggest you to give it a name, then inside the curly braces, we have to mention all the member variables, which are nothing but normal C language variables of different types like int, float, array etc.

- After the closing curly brace, we can specify one or more structure variables, again this is optional.

- **Note:** The closing curly brace in the structure type declaration must be followed by a semicolon(;).

Defining a Structure

- We use the struct keyword to define a structure in C programming language.

- In the following example we have a student structure which consists of firstname, lastname, id and score of a student.

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        };
```

Point to note!

- In the above example we have defined a structure using the struct keyword. The name of the structure is student and is also referred as the **structure tag**.

- The student structure consists of four data fields namely **firstname**, **lastname**, **id** and **score**. These data fields are also known as the **structure elements** or **members**.

- The structure template ends with a semicolon.

Createing Structure Variable

- To create a structure variable we use the structure tag name.

- In the following example we are creating a structure variable std1 of type student structure.

- It is possible to declare variables of a **structure**, either along with structure definition or after the structure is defined. **Structure** variable declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

1) Declaring Structure variables separately

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        };
struct Student S1, S2; //declaring variables of struct Student
```

2) Declaring Structure variables with structure definition

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        } s1,s2;
```

Accessing the members of a structure

- To access the members of a structure we use the structure variable name and the . member operator followed by the name of the member.

- In the following example we are printing out the id of the std1 structure variable.

```
printf("ID: %s\n", std1.id);
```

- program in C to get student details from the user and store it in a structure variable and then print the details (1)

Structure and Array

- ***Create an array variable for a given structure***

In the following example we are creating a structure student to hold student detail.

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        };
```

- Here we created a single student structure variable by the name std1. Now we will create a student structure array variable stdArr

- In the following example we are creating a student structure array variable stdArr to hold details of 3 students so the size of the array is 3.

```
struct student stdArr[3];
```

Accessing members of a structure array variable

- To access a member of a structure array variable we first select the index then we target the member.

- In the following example we are selecting the first element of the structure array variable stdArr and then targeting the firstname member.

```
stdArr[0].firstname
```

- Array indexing starts from 0 so, the first element of the array is at index 0.

- ***program in C to collect details of 3 students and print the result (2)***

- In this example we will be using the student structure to create an array variable stdArr of size 3 to hold details of 3 students.

- In the code we are using & like &stdArr[i].score when taking integer value. For string input we don't need the ampersand so, we have std[i].firstname, std[i].lastname and std[i].id.

Passing structure to function

- In the Structures and Arrays we learned how to create array of structures. Here we will be using some of those concepts.

- Lets get started...
- To pass a structure to a function we have to properly declare the function parameter list.

In the following example we are creating a structure student

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        };
```

- Now, lets say we want to create a displayDetail() function which takes the student structure variable as argument and prints the details.

- For this we will have to first declare the displayDetail() function.

- **Syntax of a function declaration taking structure as argument**

- Following is the function declaration syntax to accept structure variable as argument.

```
returnType functionName(struct tagName argName);
```

- Example:

```
void displayDetail(struct student std);
```

- In the above code we are declaring a function named displayDetail. The return type of this function is set to void which means the function will return no value.

- In the list of parameters we have std of struct student type. This means std is a variable of student structure. So, the function displayDetail can take any variable of type student structure as argument.

- Passing structure variable to function

- To pass a structure variable to a function all we have to do is write the name of the variable and it will pass a copy of the structure variable.

- In the following example code we are passing stdArr[i] variable of type student structure to the displayDetail function.

```
displayDetail(stdArr[i]);
```

- **program in C to take details of 3 students as input and display the result by passing the structure to a function. (3)**

Function returning structure

- First lets create a student structure.

```
struct student
        {
        char firstname[64];
        char lastname[64];
        char id[64];
        int score;
        };
```

Here we will create a function that will return variable of type student structure.

Syntax of a function declaration returning structure

Following is the syntax of a function declaration that will return structure.

```
returnType functionName(dataType paramName, ...);
```

Example:

```
struct student getDetail(void);
```

- In the above example we have a function by the name getDetail. The parameter list is set to void which means this function takes no argument.

- The return type of the function is of type struct student which means it will return a value of type student structure.

- **program in C to take details of 3 students as input and print the details using functions (4)**