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 a structure

Following is the syntax of a structure.

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

As you can see in the syntax above, 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;

};
```

Points 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.

Creating structure variables

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);
```

Write a program in C to get student details from the user and store it in a structure variable and then print the details

```
#include <stdio.h>

int main(void) {
  // creating a student structure template
  struct student {
    char firstname[64];
    char lastname[64];
    char id[64];
    int score;
  };

  // creating a student structure variable
  struct student std1;

  // taking user input
  printf("Enter First Name: ");
  scanf("%s", std1.firstname);

  printf("Enter Last Name: ");
  scanf("%s", std1.lastname);

  printf("Enter ID: ");
  scanf("%s", std1.id);

  printf("Enter Score: ");
  scanf("%d", &std1.score);
  // output
  printf("\nStudent Detail:\n");
  printf("Firstname: %s\n", std1.firstname);
  printf("Lastname: %s\n", std1.lastname);
  printf("ID: %s\n", std1.id);
  printf("Score: %d\n", std1.score);
  return 0;
}
```

```
Enter First Name: Yusuf
Enter Last Name: Shakeel
Enter ID: student-01
Enter Score: 8

Student Detail:
Firstname: Yusuf
Lastname: Shakeel
ID: student-01
Score: 8
```

C - Structures and Arrays

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.

Write a program in C to collect details of 3 students and print the result

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.

```
#include <stdio.h>

int main(void) {

  // creating a student structure template

  struct student {

    char firstname[64];

    char lastname[64];

    char id[64];

    int score;

  };


  // creating a student structure array variable

  struct student stdArr[3];


  // other variables

  int i;


  // taking user input

  for (i = 0; i < 3; i++) {

    printf("Enter detail of student #%d\n", (i+1));
    printf("Enter First Name: ");
    scanf("%s", stdArr[i].firstname);

    printf("Enter Last Name: ");
    scanf("%s", stdArr[i].lastname);

    printf("Enter ID: ");
    scanf("%s", stdArr[i].id);

    printf("Enter Score: ");
    scanf("%d", &stdArr[i].score);
  }
  // output
  for (i = 0; i < 3; i++) {
    printf("\nStudent #%d Detail:\n", (i+1));
    printf("Firstname: %s\n", stdArr[i].firstname);
    printf("Lastname: %s\n", stdArr[i].lastname);
    printf("ID: %s\n", stdArr[i].id);
    printf("Score: %d\n", stdArr[i].score);
  }
  return 0;
}
```

In the above 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.

C - 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 student structure.

```
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]);
```

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

```
#include <stdio.h>

// creating a student structure template
struct student {
  char firstname[64];
  char lastname[64];
  char id[64];
  int score;
};

// function declaration
void displayDetail(struct student std);

int main(void) {

  // creating a student structure array variable
  struct student stdArr[3];

  // other variables
  int i;

  // taking user input
  for (i = 0; i < 3; i++) {
    printf("Enter detail of student #%d\n", (i+1));

    printf("Enter First Name: ");
    scanf("%s", stdArr[i].firstname);
    printf("Enter Last Name: ");
    scanf("%s", stdArr[i].lastname);

    printf("Enter ID: ");
    scanf("%s", stdArr[i].id);

    printf("Enter Score: ");
    scanf("%d", &stdArr[i].score);
  }

  // output
  for (i = 0; i < 3; i++) {
    printf("\nStudent #%d Detail:\n", (i+1));
    displayDetail(stdArr[i]);
  }

  return 0;
}

void displayDetail(struct student std) {
  printf("Firstname: %s\n", std.firstname);
  printf("Lastname: %s\n", std.lastname);
  printf("ID: %s\n", std.id);
  printf("Score: %d\n", std.score);
}
```

```
Enter detail of student #1
Enter First Name: Bruce
Enter Last Name: Wayne
Enter ID: dc-01
Enter Score: 8
Enter detail of student #2
Enter First Name: Peter
Enter Last Name: Parker
Enter ID: mc-01
Enter Score: 9
Enter detail of student #3
Enter First Name: Tony
Enter Last Name: Stark
Enter ID: mc-02
Enter Score: 7

Student #1 Detail:
Firstname: Bruce
Lastname: Wayne
ID: dc-01
Score: 8

Student #2 Detail:
Firstname: Peter
Lastname: Parker
ID: mc-01
Score: 9
```

```
Student #3 Detail:
Firstname: Tony
Lastname: Stark
ID: mc-02
Score: 7
```

C - 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.

Write a program in C to take details of 3 students as input and print the details using functions

In the following example we are using two functions getDetail to get student details and displayDetail to display student details.

```
#include <stdio.h>

// creating a student structure template
struct student {
  char firstname[64];
  char lastname[64];
  char id[64];
  int score;
};

// function declaration
```

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

int main(void) {

  // creating a student structure array variable
  struct student stdArr[3];

  // other variables
  int i;

  // taking user input
  for (i = 0; i < 3; i++) {
    printf("Enter detail of student #%d\n", (i+1));
    stdArr[i] = getDetail();
  }

  // output
  for (i = 0; i < 3; i++) {
    printf("\nStudent #%d Detail:\n", (i+1));
    displayDetail(stdArr[i]);
  }

  return 0;
}

struct student getDetail(void) {
  // temp structure variable
  struct student std;

  printf("Enter First Name: ");
  scanf("%s", std.firstname);

  printf("Enter Last Name: ");
  scanf("%s", std.lastname);

  printf("Enter ID: ");
  scanf("%s", std.id);

  printf("Enter Score: ");
  scanf("%d", &std.score);

  return std;
}

void displayDetail(struct student std) {
  printf("Firstname: %s\n", std.firstname);
  printf("Lastname: %s\n", std.lastname);
  printf("ID: %s\n", std.id);
  printf("Score: %d\n", std.score);
}
```

```
Enter detail of student #1
Enter First Name: Bruce
Enter Last Name: Wayne
Enter ID: dc-01
Enter Score: 8
Enter detail of student #2
Enter First Name: Peter
Enter Last Name: Parker
Enter ID: mc-01
Enter Score: 9
Enter detail of student #3
Enter First Name: Tony
Enter Last Name: Stark
Enter ID: mc-02
Enter Score: 7

Student #1 Detail:
Firstname: Bruce
Lastname: Wayne
ID: dc-01
Score: 8

Student #2 Detail:
Firstname: Peter
Lastname: Parker
ID: mc-01
Score: 9
```

Student #3 Detail:

Firstname: Tony

Lastname: Stark

ID: mc-02

Score: 7

C - Structure in Structure

Here we will learn to handle structure in a structure in C programming language.

We have already learned how to create and access members of a structures in the structure tutorial.

Now, lets add a structure inside a structure and learn how to work with it.

In the following example we have a student structure. Inside it we have another structure address to hold the address detail.

```
// student structure having address structure inside
struct student {
  char name[255];
  char id[20];

  // address of the student
  struct {
    char line1[255];
    char line2[255];
    char city[255];
    char state[100];
    char country[255];
    char pincode[20];
  } address;
};
```

In the above code snippet we can see that we have an address structure inside the student structure.

We already know by now that to access any member of a structure variable we use the . operator.

So, if we want to access the name member of the student structure we have to write the following.

```
std.name
```

Similarly if we want to access the city member of the address structure which is inside the student structure, we have to write the following.

```
std.address.city
```

In the following code we will take student detail from the user and then print the output.

```
#include <stdio.h>

int main(void) {
  // student structure having address structure inside
  struct student {
    char name[255];
    char id[20];

    // address of the student
    struct {
```

```
    char line1[255];
    char line2[255];
    char city[255];
    char state[100];
    char country[255];
    char pincode[20];
   } address;
};

// create structure variable
struct student std;

// take student data
printf("Enter student detail:\n");
printf("Name: ");
gets(std.name);
printf("ID: ");
gets(std.id);
printf("Address Line1: ");
gets(std.address.line1);
printf("Address Line2: ");
gets(std.address.line2);
printf("City: ");
gets(std.address.city);
printf("State: ");
gets(std.address.state);
printf("Country: ");
```

```
  gets(std.address.country);
  printf("Pincode: ");
  scanf("%s", std.address.pincode);

  // display result
  printf("Student Detail:\n");
  printf("Name: %s\n", std.name);
  printf("ID: %s\n", std.id);
  printf("\nAddress of the student:\n");
  printf("%s\n", std.address.line1);
  printf("%s\n", std.address.line2);
  printf("%s, %s %s\n", std.address.city, std.address.state, std.address.country);
  printf("%s\n", std.address.pincode);

  return 0;
}
```

```
Enter student detail:
Name: Yusuf Shakeel
ID: s01
Address Line1: House #123
Address Line2: Awesome Street, 1024th Main Road
City: Bangalore
State: KA
Country: India
Pincode: 560001
```

Student Detail:

Name: Yusuf Shakeel

ID: s01

Address of the student:

House #123

Awesome Street, 1024th Main Road

Bangalore, KA India

560001

C - Pointers and Structures