Fundamentals of Programming

Lab Manual 10

Structure in C Programming

SCHOOL OF INTERDISCIPLINARY ENGINEERING AND SCIENCES NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST)

Objective:

The objective of this lab is to understand the concepts of Structure in C programming.

Structure:

The structure in C is a user-defined data type that can be used to group items of possibly different types into a single type. The **struct keyword** is used to define the structure in the C programming language. The items in the structure are called its **member** and they can be of any valid data type.

Define Structures

Before you can create structure variables, you need to define their data type. To define a struct, the struct keyword is used.

Syntax of struct:

```
struct structureName {
  dataType member1;
  dataType member2;
  ...
};
```

For example:

```
struct Person {
  char name[50];
  int citNo;
  float salary;
};
```

Here, a derived type struct Person is defined. Now, you can create variables of this type.

Create struct Variables

When a struct type is declared, no storage or memory is allocated. To allocate memory of a given structure type and work with it, we need to create variables.

Here's how we create structure variables:

```
struct Person {
  // code
};


int main() {
  struct Person person1, person2, p[20];
  return 0;
}
```

Another way of creating a struct variable is:

```
struct Person {
   // code
} person1, person2, p[20];
```

In both cases,

- person1 and person2 are struct Person variables
- p[] is a struct Person array of size 20.

Access Members of a Structure

There are two types of operators used for accessing members of a structure.

. - Member operator

-> - Structure pointer operator (will be discussed in the next tutorial)

Suppose, you want to access the salary of person2. Here's how you can do it.

```
person2.salary
```

Example 1: C structs

```
#include <stdio.h>
#include <string.h>

// create struct with person1 variable
struct Person {
  char name[50];
  int citNo;
  float salary;
} person1;

int main() {

  // assign value to name of person1
  strcpy(person1.name, "George Orwell");

  // assign values to other person1 variables
  person1.citNo = 1984;
  person1. salary = 2500;

  // print struct variables
  printf("Name: %s\n", person1.name);
  printf("Citizenship No.: %d\n", person1.citNo);
  printf("Salary: %.2f", person1.salary);
  return 0;
}
```

Run Code

Output

```
Name: George Orwell
Citizenship No.: 1984
Salary: 2500.00
```

In this program, we have created a struct named Person. We have also created a variable of Person named person1.
In main(), we have assigned values to the variables defined in Person for the person1 object.

```
strcpy(person1.name, "George Orwell");
person1.citNo = 1984;
person1. salary = 2500;
```

Notice that we have used strcpy() function to assign the value to person1.name.
This is because the name is a char array (C-string) and we cannot use the assignment operator = with it after we have declared the string.
Finally, we printed the data of person1.

C structs and Pointers

Here's how you can create pointers to structs.

```
struct name {
   member1;
   member2;
   };
```

```
int main()
{
    struct name *ptr, Harry;
}
```

Here, ptr is a pointer to struc

Example: Access members using Pointer

To access members of a structure using pointers, we use the -> operator.

```
#include <stdio.h>
struct person
{
   int age;
   float weight;
};

int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;

    printf("Enter age: ");
    scanf("%d", &personPtr->age);

    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);
    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);

    return 0;
}
```

In this example, the address of person1 is stored in the personPtr pointer using personPtr = &person1;.

Now, you can access the members of person1 using the personPtr pointer.

By the way,

personPtr->age is equivalent to (*personPtr).age

personPtr->weight is equivalent to (*personPtr).weight

Dynamic memory allocation of structs

Before you proceed with this section, we recommend you check C dynamic memory allocation.

Sometimes, the number of struct variables you declared may be insufficient. You may need to allocate memory during run-time. Here's how you can achieve this in C programming.

Example: Dynamic memory allocation of structs

```
#include <stdio.h>
#include <stdlib.h>
struct person {
   int age;
   float weight;
   char name[30];
};

int main()
{
   struct person *ptr;
   int i, n;

   printf("Enter the number of persons: ");
   scanf("%d", &n);

   // allocating memory for n numbers of struct person
   ptr = (struct person*) malloc(n * sizeof(struct person));

   for(i = 0; i < n; ++i)
   {
       printf("Enter first name and age respectively: ");

       // To access members of 1st struct person,
       // ptr->name and ptr->age is used

       // To access members of 2nd struct person,
       // (ptr+1)->name and (ptr+1)->age is used
       scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);
   }

   printf("Displaying Information:\n");
   for(i = 0; i < n; ++i)
       printf("Name: %s\tAge: %d\n", (ptr+i)->name, (ptr+i)->age);

   return 0;
}
```

When you run the program, the output will be:

```
Enter the number of persons:  2
Enter first name and age respectively:  Harry 24
Enter first name and age respectively:  Gary 32
Displaying Information:
Name: Harry          Age: 24
Name: Gary Age: 32
```

In the above example, n number of struct variables are created where n is entered by the user.

To allocate the memory for n number of struct person, we used

```
ptr = (struct person*) malloc(n * sizeof(struct person));
```

Then, we used the ptr pointer to access the elements of the person.

C Structure and Function

Similar to variables of built-in types, you can also pass structure variables to a function.

Passing structs to functions

Here's how you can pass structures to a function

```
#include <stdio.h>
struct student {
   char name[50];
   int age;
};

// function prototype
void display(struct student s);

int main() {
   struct student s1;

   printf("Enter name: ");

   // read string input from the user until \n is entered
   // \n is discarded
   scanf("%[^\n]%*c", s1.name);

   printf("Enter age: ");
   scanf("%d", &s1.age);

   display(s1); // passing struct as an argument
   return 0;
}

void display(struct student s) {
   printf("\nDisplaying information\n");
   printf("Name: %s", s.name);
   printf("\nAge: %d", s.age);
}
```

Output

```
Enter name: Bond
Enter age: 13
Displaying information
Name: Bond
Age: 13
```

Here, a struct variable s1 of type struct student is created. The variable is passed to the display() function using the display(s1); statement.

Return struct from a function

Here's how you can return structure from a function:

```
#include <stdio.h>
struct student
{
    char name[50];
    int age;
};

// function prototype
```

```
struct student getInformation();

int main()
{
    struct student s;

    s = getInformation();

    printf("\nDisplaying information\n");
    printf("Name: %s", s.name);
    printf("\nRoll: %d", s.age);

    return 0;
}
struct student getInformation()
{
  struct student s1;

  printf("Enter name: ");
  scanf ("%[^\n]%*c", s1.name);

  printf("Enter age: ");
  scanf("%d", &s1.age);

  return s1;
}
```

Here, the getInformation() function is called using s = getInformation(); statement. The function returns a structure of type struct student. The returned structure is displayed from the main() function.

Notice that, the return type of getInformation() is also struct student.

Passing struct by reference

You can also pass structs by reference (in a similar way like you pass variables of built-in type by reference). We suggest you read the pass-by tutorial before you proceed.

During pass-by reference, the memory addresses of struct variables are passed to the function.

```
#include <stdio.h>
typedef struct Complex
{
    float real;
    float imag;
} complex;

void addNumbers(complex c1, complex c2, complex *result);

int main()
{
    complex c1, c2, result;

    printf("For first number,\n");
    printf("Enter real part: ");
    scanf("%f", &c1.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c1.imag);

    printf("For second number, \n");
    printf("Enter real part: ");
    scanf("%f", &c2.real);
    printf("Enter imaginary part: ");
    scanf("%f", &c2.imag);

    addNumbers(c1, c2, &result);
    printf("\nresult.real = %.1f\n", result.real);
    printf("result.imag = %.1f", result.imag);

    return 0;
}
void addNumbers(complex c1, complex c2, complex *result)
{
     result->real = c1.real + c2.real;
     result->imag = c1.imag + c2.imag;
}
```

Output

```
For first number,
Enter real part:  1.1
Enter imaginary part:  -2.4
For second number,
Enter real part:  3.4
Enter imaginary part:  -3.2


result.real = 4.5
result.imag = -5.6
```

In the above program, three structure variables c1, c2, and the address of the result are passed to the addNumbers() function. Here, the result is passed by reference.

When the result variable inside the addNumbers() is altered, the result variable inside the main() function is also altered accordingly.

Lab Task:

Task 1:

You are tasked with creating a program to manage and display details of books using structs in C programming.

Requirements:

1. Define a struct named Book with the following members:
   - title: a string to store the title of the book.
   - author: a string to store the author of the book.
   - price: a float to store the price of the book.
2. Write a function initializeBook() to initialize the details of a book:
   - The function should take a Book struct pointer as input.
   - Inside the function, prompt the user to enter the title, author, and price of the book.
   - Assign the entered values to the respective members of the struct.
3. Write a function printBookDetails() to print the details of a book:
   - The function should take a Book struct as input.
   - Print out the title, author, and price of the book.

4. In the main() function:
   - Declare variables of type Book.

- Initialize the details of each book using the initializeBook() function.
- Print out the details of each book using the printBookDetails() function.

Example:

```
Enter details of Book 1:
Title: Introduction to Programming
Author: John Smith
Price: 29.99

Enter details of Book 2:
Title: Algorithms and Data Structures
Author: Jane Doe
Price: 39.99

Book 1 Details:
Title: Introduction to Programming
Author: John Smith
Price: $29.99

Book 2 Details:
Title: Algorithms and Data Structures
Author: Jane Doe
Price: $39.99
```

Task 2:

Your task is to create a program that handles rectangle calculations using structs in C programming.

Requirements:

1. Define a struct named Rectangle with the following members:
   - length: a float to store the length of the rectangle.
   - width: a float to store the width of the rectangle.
2. Write a function calculateArea() to calculate the area of a rectangle:
   - The function should take a Rectangle struct as input.
   - Calculate the area of the rectangle using the formula: area = length * width.
   - Return the calculated area.
3. Write a function calculatePerimeter() to calculate the perimeter of a rectangle:
   - The function should take a Rectangle struct as input.
   - Calculate the perimeter of the rectangle using the formula: perimeter = 2 * (length + width).
   - Return the calculated perimeter.
4. In the main() function:
   - Declare a variable of type Rectangle.
   - Prompt the user to enter the length and width of the rectangle.
   - Initialize the length and width of the rectangle using user input.
   - Call the calculateArea() function to calculate the area of the rectangle and print the result.
   - Call the calculatePerimeter() function to calculate the perimeter of the rectangle and print the result.

Example:

```
Enter the length of the rectangle: 5.5

Enter the width of the rectangle: 3.8


Area of the rectangle: 20.9 square units

The perimeter of the rectangle: 18.6 units
```

Task 3:

Your task is to create a program that updates the age of a person using structs and pointers in C programming.

Requirements:

1. Define a struct named Person with the following members:
   - name: a string to store the name of the person.
   - age: an integer to store the age of the person.
   - address: a string to store the address of the person.
2. Write a function updateAge() to update the age of a person:
   - The function should take a pointer to a Person struct as input.
   - Inside the function, prompt the user to enter the new age of the person.
   - Update the age member of the person struct with the entered value.
3. In the main() function:
   - Declare a variable of type Person.
   - Prompt the user to enter the name, age, and address of the person.
   - Initialize the name, age, and address of the person using user input.
   - Call the updateAge() function to update the age of the person.
   - Print out the updated details of the person.

Example:

```
Enter the name of the person: John Doe
Enter the age of the person: 30
Enter the address of the person: 123 Main Street
Enter the new age of the person: 35

Updated Details:
Name: John Doe
Age: 35
Address: 123 Main Street
```

Task 4:

Your task is to create a program that dynamically allocates memory for an array of student structures in C programming.

Requirements:

1. Define a struct named Student with the following members:
   - ID: an integer to store the student ID.
   - name: a string to store the name of the student.
   - marks: a float to store the marks of the student.
2. Prompt the user to enter the number of students for which memory needs to be allocated.
3. Dynamically allocate memory for an array of Student structs based on the input provided by the user.
4. Prompt the user to enter the details (ID, name, and marks) of each student and store them in the allocated memory.
5. Display the details of all the students entered by the user.
6. Free the dynamically allocated memory before exiting the program.

Example:

```
Enter the number of students: 3
Enter details for Student 1:
ID: 101
Name: Alice
Marks: 85.5
Enter details for Student 2:
ID: 102
Name: Bob
Marks: 78.9
Enter details for Student 3:
ID: 103
Name: Claire
```

```
Marks: 92.0

Student Details:

ID: 101, Name: Alice, Marks: 85.5

ID: 102, Name: Bob, Marks: 78.9

ID: 103, Name: Claire, Marks: 92.0
```

Task 5:

Your task is to create a program that displays the details of a car using structs in C programming.

Requirements:

1. Define a struct named Car with the following members:
   - model: a string to store the model of the car.
   - year: an integer to store the year of the car.
   - price: a float to store the price of the car.
2. Write a function named displayCarDetails() to display the details of a car:
   - The function should take a Car struct as input.
   - Print out the model, year, and price of the car.
3. In the main() function:
   - Declare a variable of type Car.
   - Prompt the user to enter the model, year, and price of the car.
   - Initialize the model, year, and price of the car using user input.
   - Call the displayCarDetails() function to display the details of the car.

Example:

```
Enter the model of the car: Toyota Camry

Enter the year of the car: 2022

Enter the price of the car: 25000.50


Car Details:

Model: Toyota Camry
```

Year: 2022

Price: $25000.50