CS118 Programming Fundamentals

LAB 10

STRUCTURES in C

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

C Structures

C arrays allow you to define type of variables that can hold several data items of the same kind but structure is another user defined data type available in C programming, which allows you to combine data items of different kinds.

Structure is the collection of heterogeneous data items unlike an array. Structure can also be defined as a collection of a fixed number of components in which the components are accessed by name. The components may be of different types. All the data items in a structure may or may not be of same data type. There are some data items, which are group of multiple values instead of a single value. In short, Structures are used to represent a record.

Example: Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book:

- Title,
- Author,
- Subject
- Book ID

Defining a Structure

Structs

A struct (short for structure) in C is a grouping of variables together into a single type.

```
struct nameOfStruct
{
    type member;
    type member;
     …
};
```

Note the semicolon at the end.

```
struct Books
{
    char    title[50];
    char    author[50];
    char    subject[100];
    int     book_id;
};
```

Declaring a Structure

To declare a Structure variable:

Syntax: struct nameOfStruct variable_name;

Example: struct Books Book1;

Another Example:

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

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

Accessing a Structure:

Once structure is defined and its variable is created, we can access the individual members of the structure with the help of dot operator (.). The dot operator is also called as member access operator.

```c
#include<stdio.h>

struct Point
{
   int x, y;
};

int main()
{
   struct Point p1 = {0, 1};

   // Accessing members of point p1
   p1.x = 20;
   printf ("x = %d, y = %d", p1.x, p1.y);

   return 0;
}
```

Another Example:

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

struct Books
{
    char    title[50];
    char    author[50];
    char    subject[100];
    int     book_id;
} Book1, Book2;

int main( )
{

    /* book 1 specification */
    strcpy( Book1.title, "CProgramming");
    strcpy( Book1.author, "Nuha Ali");
    strcpy( Book1.subject, "C ProgrammingTutorial");
    Book1.book_id = 6495407;

    /* book 2 specification */
    strcpy( Book2.title, "TelecomBilling");
    strcpy( Book2.author, "Zara Ali");
    strcpy( Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;

    /* print Book1 info */
    printf( "Book 1 title : %s\n", Book1.title);
```

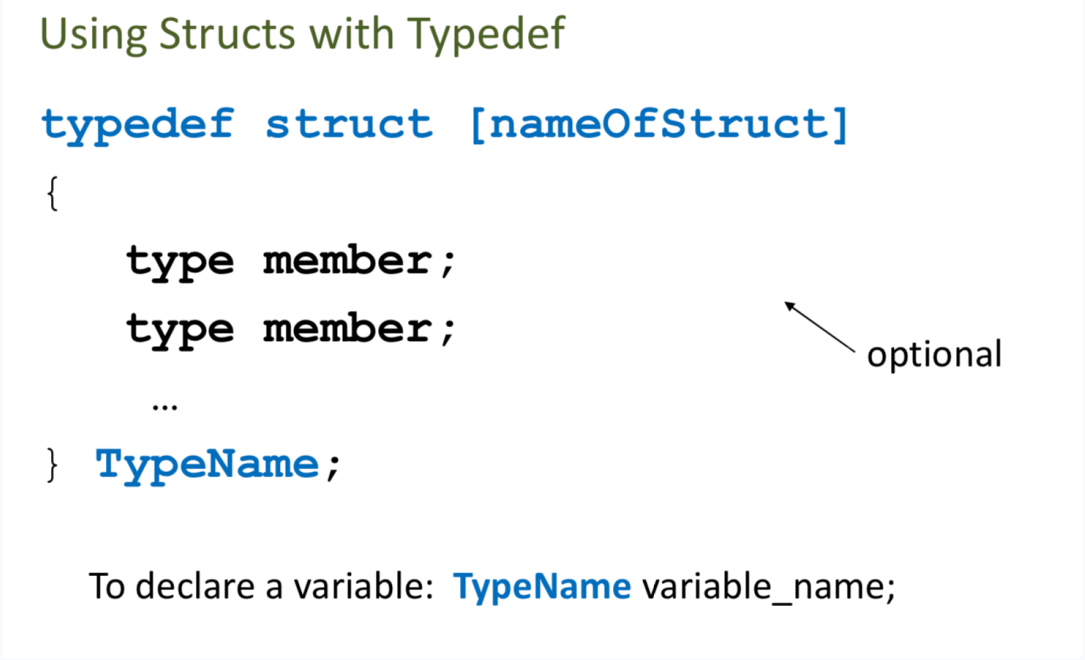
```
    printf( "Book 1 author : %s\n", Book1.author);
    printf( "Book 1 subject : %s\n", Book1.subject);
    printf( "Book 1 book_id : %d\n",
    Book1.book_id);

    /* print Book2 info */
    printf( "Book 2 title : %s\n", Book2.title);
    printf( "Book 2 author : %s\n", Book2.author);
    printf( "Book 2 subject : %s\n",
    Book2.subject);
    printf( "Book 2 book_id : %d\n",
    Book2.book_id);

    return 0;
}
```

Keyword typedef

We use the `typedef` keyword to create an alias name for data types. It is commonly used with structures to simplify the syntax of declaring variables.



```
#include <stdio.h>
typedef struct
{
        int width;
        int length;
        int height;
} Box;
typedef struct {  double radius; } Circle;
int main()
{
        Box b;   /* instead of struct Box */
        Circle c;/* instead of struct Circle */
        b.width = 10;
        b.length = 30;
        b.height = 10;
        c.radius = 10;
}
```

Size of a Struct: sizeof

```
typedef struct
{
  double radius;      /* 8 bytes */
  int x;              /* 4 bytes */
  int y;              /* 4 bytes */
  char name[10];      /* 10 bytes */
} Circle;

printf("Size of Circle struct is %d\n",
        sizeof(Circle));
```

Structure of Structures / Nested Structures

If one of the member of a structure is itself a structure then the parent structure is called as the structure of structure. It is also termed as nested structure. You can nest as many structures as required i.e. there is no any limitation on the levels of nested structures.

```
struct complex
{
 int imag;
 float real;
};

struct number
{
   struct complex comp;
   int integers;
} num1, num2;
```

Accessing Value in Nested Structure:

```
num2.comp.imag = 11;
```

Array of Structures

You can declare an array of a structure and manipulate each one

```
typedef struct
{
  double radius;
  int x;
  int y;
  char name[10];
} Circle;

Circle circles[5];
```

```
#include<stdio.h>

struct Point
{
   int x, y;
};

int main()
{
   // Create an array of structures
   struct Point arr[10];

   // Access array members
   arr[0].x = 10;
   arr[0].y = 20;

   printf("%d %d", arr[0].x, arr[0].y);
   return 0;
}
```

```
10 20
```

Structure With Functions

You can pass a structure as a function argument in very similar way as you pass any other variable or pointer.

Syntax:

int myFunction(struct Person p){

...

}

You would access structure variables in the similar way as you have accessed in the above example:

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

struct Box {
int width;
int height;
int length;
};

int GetVolume(struct Box b);

int main()
{
     struct Box b;

     printf("Enter the box dimensions (width length height): ");
     scanf("%d %d %d", &b.width, &b.length, &b.height);

     printf("Box volume = %d\n", GetVolume(b));
}


int GetVolume(struct Box b)
{
     return b.width * b.height * b.length;
}
```

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

```c
#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;
}
```

Pointers to Structures

- Defining pointers to structures in very similar way as you define pointer to any other variable as follows:

```
struct Books *struct_pointer;
```

- Store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the & operator before the structure's name as follows:

```
struct_pointer = &Book1;
```

- To access the members of a structure using a pointer to that structure, you must use the -> operator as follows:

```
struct_pointer->title;
```

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

struct Books
{
    char    title[50];
    char    author[50];
    char    subject[100];
    int     book_id;
};

/* function declaration */
void printBook( struct Books *book );
int main( )
{
    struct Books Book1;             /* Declare Book1 of type Book*/
    struct Books Book2;             /* Declare Book2 of type Book*/

    /* book 1 specification */
    strcpy( Book1.title, "CProgramming");
    strcpy( Book1.author, "Nuha Ali");
    strcpy( Book1.subject, "C ProgrammingTutorial");
```

```
    Book1.book_id = 6495407;

    /* book 2 specification */
    strcpy( Book2.title, "TelecomBilling");
    strcpy( Book2.author, "Zara Ali");
    strcpy( Book2.subject, "Telecom Billing Tutorial");
    Book2.book_id = 6495700;

    /* print Book1 info by passing address of Book1 */
    printBook( &Book1 );

    /* print Book2 info by passing address of Book2 */
    printBook( &Book2 );

    return 0;
}
void printBook( struct Books *book )
{
    printf( "Book title : %s\n", book->title);
    printf( "Book author : %s\n", book->author);
    printf( "Book subject : %s\n", book->subject);
    printf("Book book_id :%d\n", book->book_id);
}
```

Another Example:

```c
#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;
}
```

LAB TASK

1. Write a program in C that creates structure STUDENT. The STUDENT structure contains the following members: Name, Roll number, Attendance Marks, Test1 marks, Test2 marks and Test3 marks. Initialize two variables to store the record of two students. The program should display the name; roll number and total sessional marks of both the students.

2. Write a program in C that creates structure TIME. Initialize two variables to store the starting and ending time of the race. The program should calculate the elapsed time in the third TIME variable and display it.

3. Write a program in C that creates structure EMPLOYEE. The EMPLOYEE structure contains the following members: Name, ID, Salary, Address (flat/house no, street, Area, city, Country), Age and Designation. Store and Print 10-employee list by using array of Structures.
   **Here Address is inner struct in Employee.

4. Calculate the average Salary of the 10 employees by adding a function in the TASK 3.

5. By using Pointer to Structures, increment +3 in Test 1, +10 in Test 2 and +8 in Test 3 and then display the name; roll number and total sessional marks (After increment) of both the students.

6. Calculate the Sizeof() Struct EMPLOYEE.

7. Write a program to compare two dates entered by user. Make a structure named Date to store the elements day, month and year to store the dates. If the dates are equal, display "Dates are equal" otherwise display "Dates are not equal".

8. Write a structure to store the name, account number and balance of customers (more than 10) and store their information.
   1 - Write a function to print the names of all the customers having balance less than 2000 PKR.
   2 - Add 1000 PKR in the balance of all the customers having more than 2000 in their balance by using Pointers in structure then print the incremented value of their balance.

9. Write a structure to store the roll no., name, age (between 11 to 14) and address of students (more than 10). Store the information of the students.
   1 - Write a function to print the names of all the students having age 14.
   2 - Write another function to print the names of all the students having even roll no.
   3 - Write another function to display the details of the student whose roll no is given (i.e. roll no. Entered by the user).

10. Write a program that calculate Area of square, Volume of Cube, Circumference of Circle, and Hypotenuse of a triangle using structures to function.
    ** Value are entered by User.