GJU
الجامعة الألمانية الأردنية
German Jordanian University

CS1160
Lab Assignment 11: Structures

Date: 27 December 2020 **Online Lab Session**

I. Overview

In this lab, you will learn about structures. Declare, initialize, and the usage of them in many different ways, by applying them directly in main function, other user defined functions and using arrays of structures to represent real life records.

II. Definition

Structures are collections of related variables under one name. In contrast, to arrays that contain only elements of the same data type. Structures may contain variables of many different data types.

III. Review

1. **Declare a structure:**

General Expression:

```
struct structName {
        data_type varName1;
            .
            .
            .
};
```

General Expression for struct inside a stucut:

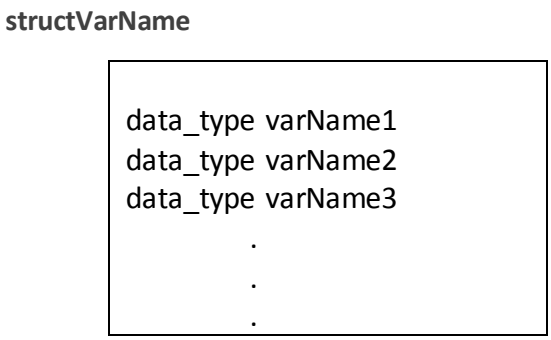
```
struct structName1 {
        data_type varName1;
        struct structName structVarName;
            .
            .
            .
};
```

2. **Declare a variable(s)** from that structure(either, inside the main, inside a user defined function or outside the main as global variable) in order to store in it some values as the following:

   A. Declare a single variable of the structure

   struct *structName* **structVarName**;

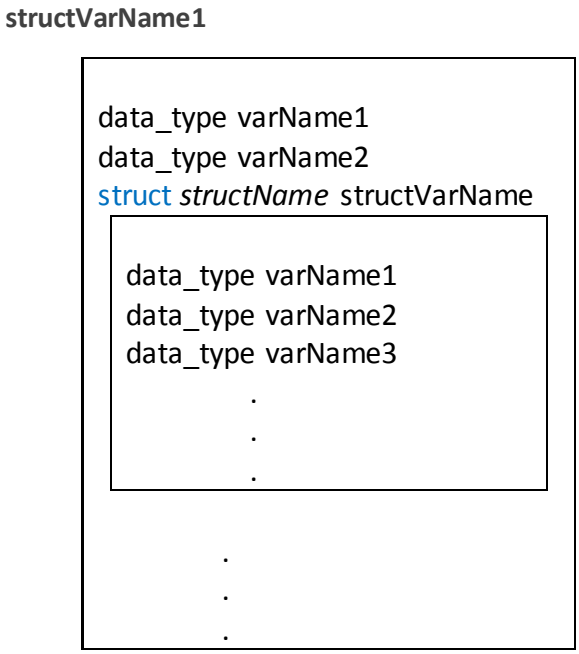
   The following is the representation in memory:



   B. Declare a single variable of the structure that has inside it another variable of structure:

   struct *structName1* **structVarName1**;

   The following is the representation in memory:



C. Declare an array of structures:

struct *structName* **structArrayName[size]**;

The following is the representation in memory:

structArrayName

0	1	2	3	size - 1
data_type varName1 *data_type* varName2 *data_type* varName3 . . .	*data_type* varName1 *data_type* varName2 *data_type* varName3 . . .	*data_type* varName1 *data_type* varName2 *data_type* varName3 . . .	*data_type* varName1 *data_type* varName2 *data_type* varName3 . . .	*data_type* varName1 *data_type* varName2 *data_type* varName3 . . .

3. **Accessing the variables**: from main function or user defined functions:

A. Single variable of the structure:

structVarName**.**varName1 = value; // the dot is called the member access operator

B. Single variable of the structure that has inside it another variable of structure

*structVarName1***.***structVarName***.**varName1 = value;

C. Array of structures:

structArrayName[pos].varName1 = value;//where *pos* is the index of the element inside the array

III. Example

```
struct Contact {
        char phoneNumber[11];
        char address[50];
        char city[15];
        char state[3];
        char zipCode[6];
};
struct Customer {
        char lastName[15];
        char firstName[15];
        int customerNumber;
        struct Contact personal;
} customerRecord;
```

IV. Tasks

1. Write a C program with the following specifications:
   A. Declare a structure called **EngineeringStudent** that has the following:
      1. Integer variable called ***studentID*** to store the id of the student.
      2. String variable (array of characters) of size 30 called **studentName** to store the student name.
      3. float variable called **mathGrade** that stores the math grade of the student.
      4. float variable called **englishGrade** that stores the english grade of the student.
      5. float variable called **CSGrade** that stores the CS grade of the student.
      6. float variable called **studentGPA** that stores the gpa of the student.
   B. In the main function declare two variables called ***student1*** and ***student2*** of type struct **EngineeringStudent** and read all of their information (except **studentGPA**) from the user.
   C. Calculate and print the gpa of each student.
   D. Find and print the maximum grade of the first student.
   E. Find and print all the student information that has the **least amount of gpa.**

2. Complex numbers are numbers that consist of a real part and an imaginary part. A complex number is expressed as: **r + b*i***, where r is the real part, b is the imaginary part, and *i* is the square root of -1. The C programming language does not include complex numbers as a native data type, i.e. no data type definition and also no operations to add, subtract, multiply, and divide complex number. Therefore, a programmer needs to write his/her own programs to deal with complex numbers. We will define a complex number as a structure with two floating point attributes as follows:

```
struct complex {
      float re; // The real part
      float im; // The imaginary part
};
```

- The C program below use the above mentioned complex number definition and asks the user to enter two complex numbers, adds the two numbers, and displays the sum.

```
#include <stdio.h>

struct complex {
      float re; // The real part
      float im; // The imaginary part
};
int main()
{

struct complex complex1, complex2, sum;

printf( "Enter the first complex number \n");
printf( "Enter the real part " );
scanf( "%f", &complex1.re );
printf( "Enter the imaginary part " );
scanf( "%f", &complex1.im );

printf( "Enter the second complex number \n");
printf( "Enter the real part " );
scanf( "%f", &complex2.re);
printf( "Enter the imaginary part " );
scanf( "%f", &complex2.im );

sum.re = complex1.re + complex2.re;
sum.im = complex1.im + complex2.im;

//show the sum
printf("Sum is (%f, %fi)\n", sum.re, sum.im);

return 0;
}
```

A. Modify the above program so that the addition of two complex numbers will be done in a function instead of doing it in the main function. Bellow the function frame:

```
struct complex add(struct complex z1, struct complex z2)
{
        struct sum;

        //Add your code here

        return sum;
}
```

B. Modify your code by adding separate functions to compute the difference between the two complex numbers, the product of the two complex numbers, and the result of dividing the first complex number by the second complex number.

- Hint: consider the two complex numbers

- The difference between the two numbers is (real1 - real2) + (imag1 - imag2)i

- The product of the two numbers is (real1*real2 –imag1*imag2) + (real1*imag2 + real2*imag1) i

- The division of the first complex number by the second complex number is given by:

$$\left(\frac{real1 * real2 + imag1 * imag2}{real2^2 + imag2^2}\right) + \left(\frac{real2 * imag1 - real1 * imag2}{real2^2 + imag2^2}\right)i$$

3. Write a C program with the following specifications :
   - Write the definition of a structure named employee that contains character array members for an employee's first and last names of size 20, an int member for the employee's age, a char member that contains 'M' or 'F' for the employee's gender, and a double member for the employee's hourly salary.
   - Using the above-mentioned definition, write a C program that asks a user to enter the values of 20 employees and save them in an array of employees.

   Hint: To define an array that contains 20 elements of type 'struct employee', you just need to write:

   ```
   struct employee employees[20];
   ```

   - Modify your program to include a function, which calculates the average salary of all employees. The function takes as input the array of all employees and returns the average salary as float value.
   - Modify you program to include a function that take the array of all employees as input and return the employee record with the youngest age. If there is more than one employee with the same youngest age, the program should return one of them.
   - Modify your program to include a function that take the array of all employees as input and sort it in an ascending way according to the employee age. There is no need for a return value to the function, but somehow the result must be assigned to the caller.
   - Modify your program to include a function, which prints all the content of the employees array. The array of all employees acts as input and there is no return type of this function.