Programming Fundamentals CT-175

Dynamic Memory Allocation

Objectives

The objective of this lab is to understand and implement structures in C and to utilize them in an efficient and suitable manner.

Tools Required

DevC++ IDE

Course Coordinator –
Course Instructor –
Lab Instructor –
Department of Computer Science and Information Technology
NED University of Engineering and Technology

Structures in C

If we want a group of same data type we use an array. If we want a group of elements of different data types we use **structures**. For Example: To store the names, prices and number of pages of a book you can declare three variables. To store this information for more than one book three separate arrays may be declared. Another option is to make a structure.

No memory is allocated when a structure is declared. It simply defines the form of structure. When a variable of that structure is made then memory is allocated. This is equivalent to saying that there is no memory for, but when we declare an integer i.e.
int var;
only then memory is allocated.

The struct keyword defines a structure type followed by an identifier (name of the structure). Then inside the curly braces, you can declare one or more members (declare variables inside curly braces) of that structure. For example:

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

Once a structure person is declared a structure variable is defined as:

Person bill;

A structure variable bill is of type structure Person. When structure variable is defined, only then the required memory is allocated by the compiler. Considering either 32-bit or 64-bit system, the memory of float is 4 bytes, memory of int is 4 bytes and memory of char is 1 byte. Hence, 58 bytes of memory is allocated for structure variable bill.
The members of structure variable are accessed using a **dot (.)** operator. Suppose, you want to access *age* of structure variable *bill* and assign it 50 to it. You can perform this task by using following code below:
Bill.age = 50;
Example

```
#include <stdio.h>
struct Person{
    char name[50];
    int age;
    float salary; };
int main(){
    Person p1;
    printf ( "Enter Full name: ");
    gets(p1.name);
    printf ( "Enter age: ");
    scanf("%d",&p1.age);
    printf ( "Enter salary: ");
    scanf ("%d",&p1.salary);
    printf ( "\nDisplaying Information. \n" );
    printf ( "Name: %s \n" ,p1.name);
    printf ("Age: %d \n", p1.age );
    printf ( "Salary: %f \n", p1.salary);
```

Exercise

1. Declare a structure named employee that stores the employee id, name, salary and department.

2. Take data input from user after declaring a variable of employee type and show the data in proper format on output screen.

3. A phone number, such as (212) 767-8900, can be thought of as having three parts: e.g., the area code (212), the exchange (767), and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone. Initialize one, and have the user input a number for the other one. Then display both numbers.

   The interchange might look like this:

   Enter area code: 415

   Enter exchange: 555

   Enter number: 1212

   Then display like below:

   My number is (212) 767-8900

   Your number is (415) 555-1212

4. Define a structure to store the following student data: CGPA, courses (course name, GPA), address (consisting of street address, city, state, zip). Input 2 student records, compare and display which student have highest GPA in which course also Display which student has the highest CGPA . HINT: define another structure to hold the courses and address.

5. Write a C program that uses functions to perform the following operations:

   i) Reading a complex number
   ii) Writing a complex number
   iii) Addition of two complex numbers
   iv) Multiplication of two complex numbers
   (Note: represent complex numbers using a structure.)