EECE.2160: ECE Application Programming

Spring 2018

Lecture 25: Key Questions
April 4, 2018

1. **Review:** Show how variables of a given structure type can be declared and initialized.

2. Show how elements within a structure can be accessed.

3. **Example:** What does the following program print?

```
#include <stdio.h>

typedef struct {
     double real;
     double imag;
} Complex;

int main() {
     Complex a = {1, 2};
     Complex b = {3.4, 5.6};
     Complex c, d, e;

     printf("A = %.2lf + %.2lfi\n", a.real, a.imag);
     printf("B = %.2lf + %.2lfi\n", b.real, b.imag);

     c = a;
     d.real = a.real + b.real;
     d.imag = a.imag + b.imag;
     e.real = a.real - b.real;
     e.imag = a.imag - b.imag;

     printf("C = %.2lf + %.2lfi\n", c.real, c.imag);
     printf("D = %.2lf + %.2lfi\n", d.real, d.imag);
     printf("E = %.2lf + %.2lfi\n", e.real, e.imag);

     return 0;
}
```

4. **Example:** Write the following functions that use the `StudentInfo` structure

- Given a pointer to a single `StudentInfo` variable, print all of the student info to the screen using the following format:
  - `Michael J. Geiger`
  - `ID #12345678`
  - `GPA: 1.23`

- Given an array of `StudentInfo` variables, compute and return the average GPA of all students in the list

- Prompt the user to enter 3 lines of input (using the format below), read the appropriate values into `StudentInfo` elements, and return a value of type `StudentInfo`
  - Format (user input underlined)
  - `Enter name: Michael J. Geiger`
  - `Enter ID #: 12345678`
  - `Enter GPA: 1.23`

5. Explain how structures can be nested inside one another.

For today's exercise, you will complete the following functions that work with the structures `Name` and `StudentInfo`. The structure definitions are listed below:

```
typedef struct {
    char first[50];
    char middle;
    char last[50];
} Name;

typedef struct {
    Name sname;
    unsigned int ID;
    double GPA;
} SINew;
```

The function descriptions are as follows:

For the `Name` structure:

- **`void printName(Name *n)`**: Print the name pointed to by `n`, using format `<first> <middle>. <last>`
- **`void readName(Name *n)`**: Prompt for and read a first, middle, and last name, and store them in the structure pointed to by `n`

For the `StudentInfo` structure:

- **`void printStudent(SINew *s)`**: Print information about the student pointed to by `s`
- **`void readStudent(SINew *s)`:** Prompt for and read information into the student pointed to by `s`
- **`void printList(SINew list[], int n)`:** Print the contents of an array list that contains `n` `StudentInfo` structures
- **`int findByLName(SINew list[], int n, char lname[])`:** Search for the student with last name `lname` in the array `list`. Return the index of the structure containing that last name, or -1 if not found
- **`int findByID(SINew list[], int n, unsigned int sID)`**: Search for the student with ID # `sID` in the array `list`. Return the index of the structure containing that last name, or -1 if not found

From Name.c:

```
// Print contents of Name struct
void printName(Name *n) {




}

// Read information into existing Name
void readName(Name *n) {







}
```

From SINew.c:

```
// Print information about student
void printStudent(SINew *s) {




}

// Reads student information into existing structure
void readStudent(SINew *s) {









}
```

From SINew.c (continued):

```
// Print list of students
void printList(SINew list[], int n) {









}

// Find student in list, based on last name
// Returns index if student found, -1 otherwise
int findByLName(SINew list[], int n, char lname[]) {









}
```

From SINew.c (continued):

```
// Find student in list, based on ID #
// Returns index if student found, -1 otherwise
int findByID(SINew list[], int n, unsigned int sID) {




}
```