

16.216: ECE Application Programming

Summer 2014

Lecture 10: Key Questions

June 19, 2014

1. Describe what a structure is in C, and how structures can be useful.
2. Explain how we can essentially declare our own types using structures.

3. Show how variables of a given structure type can be declared and initialized.

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

5. **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

8. **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

9. Explain the reasons for using dynamic memory allocation.

10. Explain the `malloc()` function.

11. Explain the use of type casting, and why it is necessary with the allocation functions.

12. Explain the `calloc()` function.

13. Explain the `realloc()` function.

14. What are the common pitfalls of dynamic memory allocation?

15. Explain how to use dynamic memory allocation with strings.

16. Explain how to use dynamic memory allocation with two-dimensional arrays.

17. **Example:** Write each of the following functions:

- a. **`char *readLine()`**: Read a line of data from the standard input, store that data in a dynamically allocated string, and return the string (as a **`char *`**)

Hint: Read the data one character at a time and repeatedly reallocate space in string

- b. **int **make2DArray(int total, int nR):** Given the total number of values and number of rows to be stored in a two-dimensional array, determine the appropriate number of columns, allocate the array, and return its starting address
Note: if **nR** does not divide evenly into **total**, round up. In other words, an array with 30 values and 4 rows should have 8 columns, even though $30 / 4 = 7.5$