lab02 snowcrash

Liam Strand

September 21, 2022

Structure

The lab is broken up into three sections.

The first section is a check-in with the students about the recent homework, how lecture is going, and their thoughts on the course. The primary purpose of this section is to help dispel impostor syndrome by reminding the students that any level of struggling is okay, and we are here to support them.

The second section is a rapid-fire review of the topics that have been covered recently.

The third section is a walkthrough of the `snowcrash` assignment.

Notes

Intro and Goals

This document is hyperlinked from my name on the title slide

The second slide is the weekly reminder that students are worthy of this course and we are here to support them.

Slides 3-6 walk through the lab's goals. It is good to state them up-front so the students know where we are going

Check-In

Ask about `hw01`, experiences in office hours, and lecture strategies. This is a good time to remind students to start assignments early and come to office hours on Wednesday, Thursday, or Friday for more one-on-one time with TAs.

Review

Loops

We open with a `while` loop example. What is this loop doing? What will it print?

Then, how can we convert this `while` loop to a `for` loop? The next slide shows the conversion. Notice that the variable declaration and initialization goes first, followed by the same condition that was in the `while` loop, followed by the increment operation.

The next slide shows a similar problem, but instead we are now asking how many times line 5 will run.

The next slide is the same problem with slightly more fun math.

2D Arrays

We're going to begin on the board with some examples of 2D arrays. We'll practice accessing both 1D rows/cols of the 2D array, and indexing all the way to individual elements.

We then go to an example of a 2D array of integers in code. Ask what this code does, then the next slide swaps rows and cols. See if the students spot the swap.

Functions

Ask for some reasons we might want to use functions.

Then we have a super basic function declaration, definition, and call. The call is hidden, ask students how we might call the function in `main()`. Advance to reveal the call. Then advance to go through the declaration, definition, and call. It might be good to note that the parameter and argument don't need to have the same name.

The function definition is missing something important! Advance to hide `main()`. Then advance to an empty function contract. Ask students what kinds of things might go in the function contract. Advance to reveal my function contract.

Pass By Value / Pass By Reference

Tell the students some reasons we might want to pass things by reference:

- We might want to pass around really big collections of data, we don't want to spend lots of time copying that stuff around.
- We don't know how big an array might be when we pass it. The compiler needs to know how the size of the stack changes with various function calls.
- Even if we wanted to pass a copy of some big data collection by value, the function call stack has limited space.

Advance for an empty table with some C++ types. Which of these types are pass-by-value and which are pass-by-reference. Advance to fill it in.

The next slide shows an example of passing an array to a function. The slide highlights the array definition, then the function calls (mention that we list the array name, no square brackets), then the function definition (note how we include the square brackets).

Finally, we move to the live-coding, where we will implement an array-printing function. The solution code is below:

```
void print_array(int arr[], size_t len) {
    if (len == 0) {
        cout << "{}" << endl;
    } else {
        cout << "{" << arr[0];
        for (size_t i = 1; i < len; i++) {
            cout << ", " << arr[i];
        }
        cout << "}" << endl;
    }
}
```

`snowcrash`

Invite students to open the lab spec (it points to the homework spec).

Discuss the 30,000ft view of the assignment, you are decoding letters which form a password, but we provide the code that encrypted them. You have to reverse-engineer the decryption functions.

Mention `pull-crash` and `diff-crash` as the mechanism to get and test the work.

Point out the part in the spec where they describe how to run `snowcrash`.

Remind everyone that they should **only** work on phase 4.