lab04 Ye Oldey Strings

Liam Strand

October 5th, 2022

Structure

We begin, as always, with a check in on the most recent homework, office hours, and a new skill that they've learned.

Then, we'll review pointers, which they should have heard about in lecture. I'm going to avoid talking about the heap because that's more complicated than what they need to fully understand right now.

We'll preview the Word Play assignment with a brief digression into programming language history to justify the introduction of C-style strings. Then we'll talk about what those are and how we can work with them.

Finally, we'll go through the lab spec quickly and remind everyone to use the 4 steps of problem solving.

Notes

Intro and Goals

This document is hyperlinked from the title slide.

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

The third slide is an acknowlegement that I've been out for a while and it's nice to be back.

Slides 4-8 go through the lab's goals.

Check-In

Ask about `tradecraft`. Students may be on a second token, so remind them that that's okay. Also worth checking in on office hour wait times, and if they've been debugging on their own yet.

Review Pointers

We're going to do a conceptual rundown on what pointers actually are, with some sort of justified example. . . . . maybe?? Then get to the buisness of walking through the syntactic example we're going to do later.

Concepts

We have a function that reads three integers from a file and indirectly "returns" those values through references to three integers in the calling scope.

Soooo. . . stack diagram!

`main()` has 3 local uninitilized integers, then calls `read_three_ints()` with a filename and pointers to those three integers.

`read_three_ints()` opens the file at the filename, reads into the 3 referenced integers, then closes the file.

Implementation

Now we have to write the darn thing. It's probably a good idea to reference the solution code as we do this.

1. Start in `main()`, declare the three variables that we want to read into.
2. Get three pointers that contain the respective addresses of those integers.
3. Write the function call to `read_three_ints()` with the filename "numbers.txt"
4. Print out the three numbers after the function call
5. Write the function body, be sure to note that we are dereferencing the pointers to store at the place they point to.
6. Compile and run
7. Write the function call to `mutate_an_int()` along with another print after the function call
8. Write the function body to mutate the integer pointed to by the parameter to be some constant value.
9. Compile and run

Preview

A little history is fun so we are going to hop into our delorian and go back to the beginning of the future when they were inventing modern programming languages.

Bjarne invented C++ and still plays an active role in its development and maintenence.

Ken and Dennis built Unix, and Dennis built C to help with that development... It turned out to be a hit! The slide lists some of the major influencees of C, the link at the bottom has a mroe complete list.

C doesn't have a `string` type :(a logical choice is to use arrays of characters instead.

Got a couple pictures so we poke at those for a bit. Notice the vertical examples of accessor syntax. The second picture shows "actual" memory addresses, which is kinda nice because it links back to the pointer discussion earlier.

Finally we have an example of `char` array syntax (can initialize with a string literal or a `char[]` literal), and if you click again the array sizes disappear because they are optional. Compilers are so smart!

Then we remind everyone that you actually really do ***need*** to think about how you are going to solve a problem before you start writing code :)

Quick reminder of the normal things that help CS homework go smoothly.

Lab

The meat of the lab is basically starting the homework which is kinda fun. The gist is that they are implementing the `is_equal()` function with a partner. In case they missed it the first time, another reminder to discuss, try, and pseudocode first before writing C++.

Then we leave the reminders slide up before starting the jamz (linked from slide) and unleashing them for work. Probably worth noting that the image might be helpful.