Dates

Assigned: April 21st, 2016

Due: April 21st, 2016, by end-of-day

Instructions

This is a group assignment dealing with pointers and memory addresses. Teams will be assigned in class at random.

You can either both submit the files under each student's D2L, or under just one student's D2L – just make sure you list the name of your teammate(s) in the submission either way.

Make sure to keep it organized – zip up your projects with names like “ex12-part1.zip” so I can tell which files belong to which project.

Table of Contents

Quick Reference

- A pointer is a type of variable that stores a memory address. It is declared with a * after the data-type, and must be assigned to the address of another variable.
- To get the memory address of any variable (non-array), you prefix the variable name with &.
- If you output the pointer variable with just the variable name, you will get a memory address.
- If you output the pointer variable prefixed with the dereference operator * you will get the value at that memory address.

```
string name = "Singh";
string* ptrCurrent = &name;

// Outputs the address of the "name" variable
cout << ptrCurrent << endl;

// Outputs the value of the "name" variable
cout << *ptrCurrent << endl;

// Updates the value of the "name" variable
*ptrCurrent = "Rai";
```

Address-of Operator:	`&`	Example:	`cout << &cost << endl;`
Dereference Operator:	`*`	Example:	`cout << *ptrPrice << endl;`
Member-of Operator:	`->`	Example:	`cout << ptrStudent->name << endl;`

Used for classes, where with a non-pointer it would be `student.name` .

Declaring a pointer	`string* ptrStringVal;`
Getting the address of a variable	`&variableName`
Assigning an address to a pointer	`ptrCurrent = &studentName;`
De-referencing a pointer to get value	`*ptrCurrent`

Use this to get the value, or set the value at that address.

Exercise: Students and Colleges

You will create two objects for this project. They can be either classes or structs, but structs might be the simplest.

College struct

- name, a string (public)
- state, a string (public)

Student struct

- name, a string (public)
- ptrSchool, a College pointer (public)

For the "ptrSchool" pointer, you will create a pointer that will point to a College object. It would be declared as: `College* ptrSchool;`

main() function

Create an array of 3 colleges and fill in the information:

Name	**JCCC**	**MCCKC**	**KCKCC**
State	**KS**	**MO**	**KS**

Then, you will create an array of 5 students.

Part 1: Student Setup

First in main, you will have the user set up each of the 5 students. Use a for-loop to iterate through each student for the setup:

1. Output the student # (the index from the for-loop)
2. Have the user input the student name, with either a cin or a getline. Store it in the **name** variable of the student object.
3. Display a list of all 3 colleges (this will be an internal for-loop). Display each college's index (the loop index), state, and name.

4. After all colleges have been displayed, have the user enter the index of the college they wish to assign this student to.
5. Assign the **`ptrSchool`** pointer the address of that college from the array.

Part 2: Student Roster

Once all of the students have been set up, we need to loop back through everything and display their information.

Use another for-loop to iterate through all students.

1. Display the student's name.
2. Display the student's ID (the index from the for-loop)
3. Display the student's college via the **`ptrSchool`** pointer. You will need to either **dereference the ptrSchool pointer, or use the member-of operator**.
4. Display the student's state via the **`ptrSchool`** pointer. **You will need to either dereference the ptrSchool pointer, or use the member-of operator.**