CS115 Exercise: C++ Pointers

Introduction

In this exercise you will expand on a program by using the reference, and dereference operators to pass a structure instance by reference to a function and manipulate it. You will also create a dynamic array for associated data.

Instructions

Part 1 - Passing pointers to functions

- In `pointers.cpp` there is a prototype, and a call to a function to input the values into a `Student` instance. Note that the parameter to this function is the address of a structure instance.
  - Write the definition for this function at the bottom of the file.
  - Don't forget to use the member selector operator for pointers (`->`) when you reference instance members in the function. e.g.:

```
cin >> stuPtr->id;
```

  - Don't worry about being fancy, this is just a simple exercise for you to play with pointers.
  - You should enter student marks between 0 and 100 (no need to error check).
  - Compile and run this C++ program.
- Write a function that takes a pointer to an instance of the `Student` structure and displays the contents.
  - Call it from the main routine.
  - Don't forget to pass the function the ***address*** of the structure instance.
  - After you've done that, compile, link, and run again.

At this point, your output should look something like this:

```
Please enter a name:  Sam
Please enter an id:  195556
Please enter a mark: 87
Please enter a mark: 88
Please enter a mark: 89
Student info:
Name: Sam
Id: 195556
Mark 0: 87
Mark 1: 88
Mark 2: 89
```

Part 2 - Making and Using Dynamically Allocated Arrays.

Let us say that you want to choose how many marks to enter for the student and have the array change size accordingly. This gives you the opportunity to work with dynamic pointers.

You will have to make the following changes to your code:

- Change `mark` within the `Student` struct to an integer pointer.
- Ask the user from `main` how many marks they would like to enter
- Change both the functions to have an additional parameter which is the number of marks.
- For the `inputStudent` function, dynamically allocate the marks according to the number required
- Add a destructor for the `Student` structure (`~Student`). The purpose of this will be to `delete` any memory inside this structure allocated using `new`

The run might look like the following:

```
 How many marks are there? 5
Please enter a name:  Sam
Please enter an id:  195556
Please enter a mark: 87
Please enter a mark: 98
Please enter a mark: 77
Please enter a mark: 88
Please enter a mark: 78
Student info:
Name: Sam
Id: 195556
Mark 0: 87
Mark 1: 98
Mark 2: 77
Mark 3: 88
Mark 4: 78
```

Part 3 (Just for Fun) - Dynamically Allocated Arrays of Structures

- Ask the user from `main` how many students there are in the class.
- In `main`, dynamically allocate an array of `Student`'s with the input size
- In one `for` loop, call the input function for each student.
- In a second `for` loop, call the print function for each student.
- Deallocate the dynamic array of students.
  - Hint 1: You will use the `delete` operator for this step. You will not write any function call to the destructor. Recall that the destructor is automatically called for us.
  - Hint 2: You can comment out the `Student` object that was declared in Part 2 since we will not be using it anymore. If you are getting a core dump, it might be fixed by eliminating the extra `Student` object.

This page last modified:
Saturday, 05-Nov-2022 12:42:41 CST

CS Dept Home Page	CS Dept Class Files	CS115 Lab Files