CS 112 - Week 9 Lab Exercise - 2022-10-21

Deadline

Due by the end of lab on 2022-10-21.

How to submit

Submit your `.cpp` and `.h` files for the problems below on https://canvas.humboldt.edu.

IF you prefer, you may instead compress your `.cpp` and `.h` files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

(I'll also accept the `.zip` file created when one downloads a folder from the CS50 IDE, as long as it includes all of your lab's `.cpp` and `.h` files -- I suspect it will also contain your resulting executables, but that's OK.)

Purpose

To practice a bit with the C++ `vector` class and with some more formatting options.

Important notes

- Be sure to put BOTH of your names and today's date in each of the files for this lab exercise.
- When you are done, or before you leave lab, the driver/whoever's account has the lab exercise files should e-mail a copy of all of the files to BOTH/ALL of you, and EACH of you should submit these files on Canvas.

Program - a little vector play, and a little numeric formatting!

Recall: in a new-enough C++ compiler (such as the one used by the CS50 IDE), the >> operator and `getline` function happen to return `true` if they successfully read something, and return `false` if the read fails.

So, if you have an `ifstream inny` that has been successfully opened for a file containing all-integers, then:

```
int next_num;

while (inny >> next_num)
{
    ... do what you'd like with next_num ...
}

inny.close();
```

...should read all of the integers from that file, one at a time, and do what is specified for each.

Consider the C++ class **`Point`** from the Week 6 Lab Exercise. (There is a link to an example version of this class from the Canvas assignment link if you are not confident in your Week Lab Exercise version.)

Write a program in a file `file-points.cpp` that does the following:

- Asks the user to enter the name of file that contains x and y coordinates for some collection of points.
- Creates an empty vector able to hold `Point` objects.
- It tries to open that file, and read each pair of x and y coordinates within, and for each pair read in tries to add a `Point` object with those coordinates into your `Point` vector.
  - Your program may be very trusting, and assume the file does indeed contain just pairs of x and y coordinates for a collection of points.
- Once it has read them all in, it should print a message to the screen saying how many points' data have been read in.
- Now use the precision features and `setw` function we discussed to print to the screen your vector's contents' x and y coordinates:
  - one point's coordinates per line,
  - each x and y value to one fixed fractional place,
  - each x and y value right-justified in a field of size 8,
  - each with a label `x:` and `y:` each right-justified in a field whose width is your choice.
  - For example, something like this:

```
x:      3.4  y:     12.3
x:     10.6  y:      2.3
```

  - (Note: you'll need to use `Point`'s accessor methods `get_x()` and `get_y()` here, rather than its methods `display` or `to_string`.)
- Feel free to do more than just the above if you wish, as long as you also do the above. 8-)

Submit your resulting `file-points.cpp`, `Point.h`, and `Point.cpp`

- When you are done, or before you leave lab, use Gmail to
  - MAIL a copy of ALL of the resulting files for these programs to BOTH of you, and
  - EACH of you should SUBMIT the required files on Canvas