Assignment 8

100 points

Purpose

The purpose of this assignment is to give you some experience in different ways of manipulating vectors, along with some additional I/O experience.

Assignment

This assignment reads in a collection of data about physical amenities in the United States and generates some reports. The data for this assigment comes from the USDA's Natural Amenities Scale and largely duplicates the work done there. That work can be found at http://www.ers.usda.gov/data-products/natural-amenities-scale.aspx.

The data in this set represents physical amenities, i.e, things that make a place nice to live. The data is given for each county in the United States. Data elements include average January temperature (high is good), total amount of January sunshine (high is good), average July temperature (low is good), average July humidity (low is good), a landform type based on elevation variations (high is good), and the amount of terrain that is next to water (high is good).

Your program will read the data file into a vector and then process it, generating reports of the data in sorted order, based on several sorting criteria.

Program

You will need to write two classes and many functions for this assignment.

`class Amenity`

This class is designed to contain and manipulate all amenity values for one county The class contains at least data members for the 6 amenities. All of these values should be floating point except for the landform, which can be an integer. The class should also contain an integer ID value. The class should also contain two C++ strings, one for the county name, and one for the two letter state abbreviation. The class should also contain a floating point value which will contain a composite amenity value, which will be computed instead of found in the data file.

All data members must be `private`.

Methods for constructors and accessors are left to your design.

`Amenity` helper functions

You will need some helper functions (*NOT* methods) to manipulate `Amenity`s. There should be 6 functions that each compare two `Amenity`s, one for comparing each of the principal amenity values in the `Amenity`. For example, comparing the January temperature, comparing the January sunshine, etc. Each of these functions should take two `Amenity`s and return a `bool`. The function should return true if the associated value of the first argument is considered "better" than the value of the second argument.

These functions are tightly tied to the `Amenity` class. Their declarations and implementations should be in the same header and source code files as the `Amenity` class.

Math utility functions

You will need two utility functions that will process `vector`s of floating point values. One should take a `vector` of floats and return the average of all the values. The second should take the vector and an average and return the standard deviation.

If you're not familiar with the formula for the standard deviation, use the following approach: Take the difference of each value and the average, square it, and sum the squares together. Divide the sum of squared differences by the number of values minus 1. This is the variance. The standard deviation is the square root of the variance.

(Yes, there are different approaches to computing the standard deviation, but this one has some nice numerical properties.)

Implementation Point: Each of the math utility functions should be implemented using a C++ STL `algorithm` and a lambda function. This is an assignment requirement.

Helper `class`

This class is primarily to be used as a function object in other parts of the program. It's purpose is largely to do selective printing of an `Amenity`, but it is also used to compare two `Amenity`s (in a different way than mentioned above). You may call the class whatever you wish. Just don't call it "Helper" as implied by this section title.

The class should have a single, private, integer data member. This data member is simply set with an integer indicating one of the data fields of an `Amenity`. All six basic amenity attributes should be allowed, as well as the composite amenity value.

There should be a way to set the data member, but the details are up to you. A single accessor with a passed in value is one way. Several accessors with no passed in values is another way.
Implementation Point: Symbolic values of some kind should be used. No magic numbers, please.

The class should also overload the function operator twice. One of these should take an `Amenity` as an operand and return nothing. Its purpose is to print out the selected contents of the `Amenity` in the following way:

On a single row, print the county name and the state followed by the data field that has been pre-selected and stored in the function object. For example, if July humidity has been selected and stored in this object, then calling this overloaded function operator will print the county, the state, and the July humidity value from the `Amenity` passed in. The row should be formatted so that the values fall in nice columns.

The second overloaded function operator is for comparisons. It takes two `Amenity`s and returns a `bool`. It should return true if the composite amenity value of the first argument is greater than the composite amenity value of the second argument.

Strongly associated with the helper class is a function you must write called `print_top_bottom_n()`. It takes a `vector` of `Amenity`s, an integer *n*, and an instance of the Amenity helper class. The idea is for the function to print out the vector using the helper class instance to print each individual row. However, because the vector will contain several thousand items, only the first and last n elements of the vector are printed, along with an ellipsis (row of dots) separating the two sections.

Implementation point: This function is to be implemented using STL algorithms and the helper class as a function object.

Main program

The main program requires two additional function which will be described here. This will be followed by a description of the main program itself.

Input function

One function reads data from a file and stores it in an STL vector of `Amenity`s. Function arguments are at your design. The name of the data file comes originally from the command line. The input file format is described below.

Composite amenity function

The second function takes a `vector` of `Amenity`s and computes and stores the composite amenity value for each one. This requires some explanation.

The composite amenity value depends on knowning the average and standard deviation of the individual amenities over all instances. That is, the average and standard deviation of all January temperatures is needed, as is the average and standard deviation of January sunshine, average and standard deviation of July temperature and so forth.

For each (and that is not a hint) of the six prinicipal amenity attributes, use an STL algorithm and a lambda to extract the attribute value from the vector of `Amenity`s into another vector of `float`s. **Implementation Point:** The algorithm and lambda are assignment requirements. Pass the `float` vector to the average routine of the math utility functions. Pass the vector and the average to the standard deviation routine.

Care must be taken in transferring the water area attribute from the `Amenity` vector to the `float` vector. Rather than simply copy the value, it must be multiplied by 100 and then have the natural log taken. The resulting log value should be stored in the `float` vector and then eventually sent to the average and standard deviation functions.

The composite attribute score for an `Amenity` is calculated by combining the z-scores of its individual attributes. For those who are not familiar with statistics, the z score is found by taking a value, subtracting its average and then dividing the result by the standard deviation. Z-scores tend to be in the range of -3 to 3, with most values between -1 and 1.

When computing the composite amenity from the z-scores, loop over the vector of `Amenity`s with a standard `for` loop.

Main program

- Read in the amenities from the data file. The name of the data file comes from the command line. Take appropriate actions for an incorrect or missing file name.
- Print out the number of records read.
- Compute the composite attributes on the records
- Print out several reports, ranking the `Amenity`s by each of the 6 original `Amenity` attributes. Sort the `Amenity`s before printing using an STL algorithm and a function pointer (**Implementation Point:** This is a program requirement.) Print only the top and bottom 12 entries of each sorted report.
- Finally, sort the `Amenity`s using an STL algorithm and an `Amenity` helper object. This should result in a sort by composite amenity value. Print the top and bottom 10 entries of this list.

Input Format

A link to a data set can be found on the web site. Each line of the data file represents the amenity data for a single county. The file is in CSV format, i.e. comma separated values. Each line consists of, in order

- an identifying number
- a two character state abbreviation
- a county name
- Average January temperature (1941-1970)
- Average hours of total January sunlight (1941-1970)
- Average July temperature (1941-1970)
- Average July humidity (1941-1970)
- Average landform topography code (1-21)
- Percent of area next to water

If you are working on turing/hopper, you can also make a copy or link directly to your directory from `/home/turing/duffin/courses/cs689/data/amenity.dat`

Output

Partial sample output from this program on `turing/hopper` is found below.

```
z123456@turing$ assign8 amenity.dat

3073 records input.
```

```
Ranking by January Temperature
================================================
          DADE  FL        67.2
       BROWARD  FL        66.8
        MONROE  FL        65.6
    PALM BEACH  FL        65.5
        MARTIN  FL        65.4
       COLLIER  FL        65.2
     CHARLOTTE  FL          64
           LEE  FL        63.5
     ST. LUCIE  FL        63.5
        HENDRY  FL        63.4
     HIGHLANDS  FL        62.9
        GLADES  FL        62.7
...
          EDDY  ND         3.8
   GRAND FORKS  ND         3.5
       PEMBINA  ND         3.3
      BELTRAMI  MN         3.1
        TOWNER  ND         3.1
    PENNINGTON  MN         2.7
   LAKE OF THE  MN         2.7
     BOTTINEAU  ND         1.7
      MARSHALL  MN         1.6
        ROSEAU  MN         1.4
      CAVALIER  ND         1.3
       KITTSON  MN         1.1

... other reports here

Ranking by Water Area
================================================
         ALGER  MI     75.0256
         DUKES  MA     75.0256
      LEELANAU  MI     75.0256
      KEWEENAW  MI     75.0256
  SAN FRANCISC  CA     75.0256
          DARE  NC     75.0256
     NANTUCKET  MA     75.0256
          DOOR  WI     75.0256
          LAKE  OH     75.0256
     MILWAUKEE  WI     75.0256
       OZAUKEE  WI     75.0256
  PRESQUE ISLE  MI     74.3534
...
       SIMPSON  KY        0.01
       WALLACE  KS        0.01
       TERRELL  TX        0.01
    SCHLEICHER  TX        0.01
        GILMER  WV        0.01
         CRANE  TX        0.01
        TIPTON  IN        0.01
       GREELEY  KS        0.01
       WICHITA  KS        0.01
      CROCKETT  TX        0.01
      JIM HOGG  TX        0.01
      CHEYENNE  NE        0.01
```

```
Composite Ranking
===============================================
       VENTURA  CA     8.47795
  SANTA BARBAR  CA     8.45805
      HUMBOLDT  CA     8.27907
          LAKE  CO      8.1369
     DEL NORTE  CA     8.09901
     MENDOCINO  CA     8.02001
   LOS ANGELES  CA     7.85669
        SUMMIT  CO      7.8036
          MONO  CA     7.62963
  SAN FRANCISC  CA     7.56279
...
        GRUNDY  IA    -3.97747
         DODGE  MN    -3.98063
         MOWER  MN    -4.06759
       CLINTON  IN    -4.09268
       SIMPSON  KY    -4.21432
     CHAMPAIGN  IL    -4.21514
        BENTON  IN    -4.35746
        WILKIN  MN    -4.50897
      RED LAKE  MN    -4.63628
        TIPTON  IN    -5.13703
```

Implementation Hints

- Get the data input section working first. Don't write any other code until this part is working. Although not a part of the assignment, you may want to add a print method to the `Amenity` class to print out an entire instance. Make sure you get rid of all this printing before you submit your assignment.
- Then work on getting various data elements extracted to a float vector. Do lots of printing to verify your work
- Although they are not to be printed out, here are the average and standard deviation of some of the values to aid in debugging.

  ```
  Jan. Temperature (F)
  average:    32.8652    standard deviation:    12.0908

  Jan. sunshine (cum. hours)
  average:    151.566    standard deviation:    33.2946

  ...

  Water_Area
  average:    4.55058    standard deviation:    1.81058
  ```

Other Points

- You should have a source code file for `main()` and all the non-member functions. You should have a source code file for the implementation of all classes. There may be other source code files.
- Your header files are required to have header guards.
- Use `const` as appropriate, both on methods and function arguments.

- A `Makefile` is required for this assignment. Makefile variables are required.
- Symbolic constants should be used to avoid magic numbers. You should use `const` to make your symbolic constants.
- The name of your source code file containing main should be `assign8.cc`.
- Function prototypes are required for all functions you write (except `main()` of course)
- Programs that do not compile on `turing/hopper` automatically receive 0 points.
- Submit your program using the electronic submission guidelines.