Lab 6 – Standard Template Library (STL)

CPRG4202 – T. MacDonald

In this lab you will implement a std::map[1] to store data read from a text file for processing. The data itself is a number of labelled (or *'keyed'*) Cartesian points, e.g. the first point is (0, 0) and is labelled "AA". Subsequent points are also labelled with a two-letter uppercase *'key'*. Your first task is to read all the labels and points from the data file, then report to the user how many points there are, as well as the total distance between all of the points in order (i.e. the distance between AA and AB + the distance between AB and AC + the distance between AC and AD, etc.). Your second task is prompt the user to enter a label (*key*) and, if the key is in the map, report the distance between that point and the start point, i.e. AA (0, 0). You will continue to allow the user to enter in labels until they want to end.

The following files are provided:

- *CartesianPoint.h* – this header file declares and defines the CartesianPoint class. Note that a CartesianPoint object does not store a label, only x and y values. The class features overloaded friend operators >> and << for inputting and outputting CartesianPoint objects. You should not need to modify this class.
- *PointData.dat* – this is the file that contains the labels and Cartesian points that the final version of your program will read into the map and process.
- *MockDataForTesting.txt* – this file contains some labels and Cartesian points that you can use for testing/debugging your program. The difference between the data in this file and PointsData.dat is that the labels/points are in order and are more easily calculated manually. The final version of your program should not read from this file.

Reading Labels/Points into the Map:

Attempt to open the data file. If the data file opens, read a label, followed by a point and insert them into the map using the label as the key. Repeat until all the data is read and close the file. If the data file did not open, tell the user and remind them to check that the file exists. If the file opened but the map is empty after the input loop, report that to the user and remind them to check that the file contains valid data in the correct format. **Only continue processing if the file was opened and the map is not empty**.

Determine the Total Distance Between All Points in Order:

Use an iterator and a loop to traverse each label/point in the map. For each label/point, determine the distance from that point to the previous point (or next point depending on how you implement this) and add that distance to a total. Note that the CartesianPoint class overloads the subtraction operator to determine the distance between two CartesianPoint objects, so you should not need to use any complicated math here. Report to the user how many points the map contains and what the total distance is.

Determine the Distance Between the Start Point and a User Selected Point:

Prompt the user to enter a label or to enter "quit" to end. If the user entered anything other than "quit", attempt to find the label they entered in the map. If it was found, report the distance between the point for the label they entered and the start point (i.e. the point labelled "AA"). Otherwise, tell the user that the label they entered is not in the map. Repeat these steps until the user enters "quit".

Exception Handling:

Catch any std::exception thrown. Report to the user that a run-time error occurred and show what exception was thrown.

[1] http://www.cplusplus.com/reference/map/map/

Lab 6 – Standard Template Library (STL)

CPRG4202 – T. MacDonald

Example Outputs:

The data file did not open (file name misspelled):

```
PointsData.dat could not be opened for input. Check that the file exists.

--------------------------------
Press any key to continue . . .
```

The data file opened but nothing was read into the map successfully (read from an empty file):

```
The map is empty. Check that the file contains valid data in the correct format.

--------------------------------
Press any key to continue . . .
```

An exception gets thrown (example, not likely to happen):

```
An error occurred at run-time: map::at

--------------------------------
Press any key to continue . . .
```

The map was successfully filled, various labels attempted (user input is shown in yellow):

```
The map contains 78 points for a total distance of 4063.4.

Enter the label of the point you wish to go to ("quit" to end): chicken

        There is no point labelled "chicken" in the map.

Enter the label of the next point ("quit" to end): ab

        There is no point labelled "ab" in the map.

Enter the label of the next point ("quit" to end): zz

        There is no point labelled "zz" in the map.

Enter the label of the next point ("quit" to end): AB

        The distance between AA (0, 0) and AB (8, 31) is 32.0156

Enter the label of the next point ("quit" to end): AC

        The distance between AA (0, 0) and AC (81, 54) is 97.3499

Enter the label of the next point ("quit" to end): quit

--------------------------------
Press any key to continue . . .
```