CS 202 Iditarod Challenge 3

Kelby Hubbard

April 14, 2020

- Repository Link: `https://github.com/krhubbard2/CS202/tree/master/Iditarod3`
- Git Commits: `https://github.com/krhubbard2/CS202/commits`
- This homework took approximately 8 hours to complete.

1 Design

Design was pretty straight forward following the homework directions as the directions were laid out very cleanly and simply. The biggest design struggle was understanding how the TSP files were orientated and making sure the classes talked to each other.

2 Post Mortem

This part of the homework wasn't too awful. It gave me very good insight on how classes talked to each other and just overall more practice on member variables and member functions. I'd say the biggest struggle was just keeping on track and staying focused on what I was working on at the exact second, not trying to backtrack or skip a step or two ahead.

3 Recursion Problems

3.1 Sample Output

Listing 1: Sample Program Output

```
City/File: fnl4461.tsp Node: 1 Lat: 5639 Lon: 6909
City/File: fnl4461.tsp Node: 2 Lat: 5652 Lon: 6142
City/File: fnl4461.tsp Node: 3 Lat: 5654 Lon: 6101
City/File: fnl4461.tsp Node: 4 Lat: 5659 Lon: 6910
City/File: fnl4461.tsp Node: 5 Lat: 5659 Lon: 6920
City/File: fnl4461.tsp Node: 6 Lat: 5661 Lon: 6182
City/File: fnl4461.tsp Node: 7 Lat: 5663 Lon: 6830
...
City/File: fnl4461.tsp Node: 4456 Lat: 9134 Lon: 6738
City/File: fnl4461.tsp Node: 4457 Lat: 9140 Lon: 6830
City/File: fnl4461.tsp Node: 4458 Lat: 9143 Lon: 6779
City/File: fnl4461.tsp Node: 4459 Lat: 9158 Lon: 6901
City/File: fnl4461.tsp Node: 4460 Lat: 9166 Lon: 6932
City/File: fnl4461.tsp Node: 4461 Lat: 9176 Lon: 6953
```

3.2 Git Commit Messages

Date	Message
2020-03-29	Started Iditarod3
2020-03-31	Write setLat, setLong, setNode, and read file.
2020-03-31	Write CityNode Class (in progress)
2020-04-08	Write CityNode Class
2020-04-08	Write CityList Class
2020-04-08	Return Euclidean distance

3.3 Source Code

```cpp
// Kelby Hubbard
// CS202
// March 29, 2020
// Iditarod Challenge #3

#include <iostream>
using std::cout;
using std::endl;
using std::cin;
#include <fstream>
using std::ifstream;
#include <string>
using std::string;
#include <sstream>
using std::istringstream;
#include "citynode.hpp"
#include "citylist.hpp"

void readTSP(string fileName, CityNode& node, CityList& city)
{
  ifstream ifile(fileName);
  //Throw error if it can't open file
  if (!ifile)
  {
    cout << "Couldn't open file." << endl;
  }
  else
  {
    city.setFileName(fileName);
    string line;
    bool loop = true;

    while (loop)
    {
      //If reading file hits an error or EOF
      if (!ifile)
      {
        if (ifile.eof())
        {
          loop = false;
        }
        else
        {
          loop = true;
        }
      }
      //If file opens correctly
      else
      {
        getline(ifile, line);
        string nodeStart = "NODE_COORD_SECTION";

        //Start of node listings
        if (line == nodeStart)
        {
          bool loop1 = true;
          while(loop1)
          {
            //If reading file hits EOF
            if (line == "EOF")
            {
              loop1 = false;
              loop = false;
            }
            else
            {
              loop1 = true;
            }
            getline(ifile, line);
```

```cpp

            //Ensure line is an int (node / info)
            istringstream iss(line);
            int val;
            iss >> val;
            if(iss)
            {
              istringstream iss1(line);
              //Grab each section of string
              for (int i = 0; i < 3; i++)
              {
                double val1;

                iss1 >> val1;
                //Node number
                if (i == 0)
                {
                  node.setNodeNumber(val1);
                }
                //Latitude
                else if (i == 1)
                {
                  node.setLatitudeY(val1);
                }
                //Longitude
                else if (i == 2)
                {
                  node.setLongitudeX(val1);
                }
              }
              city.setCityNode(node);
            }
          }
        }
      }
    }
  }
}

int main(int argc, char** argv)
{
  CityNode node0(0, 0, 0);

  //Make CityList for BRD14051
  CityList brd;
  readTSP("brd14051.tsp", node0, brd);

  //Make CityList for FL3795
  CityList fl;
  readTSP("fl3795.tsp", node0, fl);

  //Make CityList for FNL4461
  CityList fnl;
  readTSP("fnl4461.tsp", node0, fnl);

  //Make CityList for RL1304
  CityList rl;
  readTSP("rl1304.tsp", node0, rl);

  //Make CityList for U2152
  CityList u;
  readTSP("u2152.tsp", node0, u);

  brd.printAllCityNodes();
  fl.printAllCityNodes();
  fnl.printAllCityNodes();
  rl.printAllCityNodes();
  u.printAllCityNodes();

  rl.printSpecCityNode(1300);

  cout << "Distance from node 1 and node 20 in rl: " << rl.distance(1, 20)
       << endl;

  return 0;
}
```

3.4 CityNode Header

```cpp
// Kelby Hubbard
// CS202
// April 10, 2020
// Iditarod Challenge #3
#ifndef CITYNODE_HPP_
#define CITYNODE_HPP_

#include <string>
using std::string;

class CityNode
{
public:
  //Default Constructor
  CityNode();

  /*This constructor should typically be used. Adds node number,
    the lat, and long of node to the node.  */
  CityNode(unsigned int node, double latY, double lonX);

  //Sets _nodeNumber
  void setNodeNumber(unsigned int node);
  //Sets _latitude
  void setLatitudeY(double lat);
  //Sets _longitude
  void setLongitudeX(double lon);

  //returns _nodeNumber
  unsigned int getNodeNumber();
  //returns _latitude
  double getLatitudeY();
  //returns _longitude
  double getLongitudeX();

private:
  unsigned int _nodeNumber;
  double _latitude;
  double _longitude;
  double _graphX;
  double _graphY;
};



```

```


#endif
```

3.5 CityNode Source

```
// Kelby Hubbard
// CS202
// April 10, 2020
// Iditarod Challenge #3
#ifndef CITYNODE_HPP_
#define CITYNODE_HPP_

#include <string>
using std::string;

class CityNode
{
public:
  //Default Constructor
  CityNode();

  /*This constructor should typically be used. Adds node number,
    the lat, and long of node to the node.  */
  CityNode(unsigned int node, double latY, double lonX);

  //Sets _nodeNumber
  void setNodeNumber(unsigned int node);
  //Sets _latitude
  void setLatitudeY(double lat);
  //Sets _longitude
  void setLongitudeX(double lon);

  //returns _nodeNumber
  unsigned int getNodeNumber();
  //returns _latitude
  double getLatitudeY();
  //returns _longitude
  double getLongitudeX();

private:
  unsigned int _nodeNumber;
  double _latitude;
  double _longitude;
  double _graphX;
  double _graphY;
};





#endif
```

3.6 CityList Header

```cpp
// Kelby Hubbard
// CS202
// April 10, 2020
// Iditarod Challenge #3
#ifndef CITYLIST_HPP_
#define CITYLIST_HPP_

#include <iostream>
using std::cout;
using std::endl;
#include <string>
using std::string;
#include <vector>
using std::vector;
#include "citynode.hpp"
#include <cmath>

class CityList
{
public:
  //Default Constructor
  CityList();

  //Constructor automatically adding node to _cityList vector
  CityList(CityNode node);

  //Adds node to _cityList vector
  void setCityNode(CityNode node);

  //Prints whole _cityList vector
  void printAllCityNodes();

  /*Prints specific vector position of _cityList depending
    on node you want printed*/
  void printSpecCityNode(unsigned int node);

  //Sets file/city name
  void setFileName(string name);

  //returns _fileName
  string getFileName();

  //Returns Euclidean distance between two cities (node 1 & node 2).
  double distance(int first, int second);

  //Returns CityNode
  CityNode getCityNode(int node);

  //returns  _latitude from CityNode
  double getCityLat(int node);

  //returns _longitude from CityNode
  double getCityLon(int node);

  //Returns _cityList vector size (For use in TspSolver)
  int cityListSize();

  //Returns _cityList [n] value as an int
  int listVectorSpecific(int n);

  int getCityNodeInt(int node);

  //Removes node from _cityList (For use in TspSolver)
  void removeCityList(int node);

private:
  vector<CityNode> _cityList;
  string _fileName;
  CityNode node;
};












#endif
```

3.7 CityList Source

```
// Kelby Hubbard
// CS202
// April 10, 2020
// Iditarod Challenge #3
#include "citylist.hpp"

CityList::CityList() {}

CityList::CityList(CityNode node)
{
  _cityList.push_back(node);
}

void CityList::setCityNode(CityNode node)
{
  _cityList.push_back(node);
}

CityNode CityList::getCityNode(int node)
{
  return _cityList[node];
}

int CityList::getCityNodeInt(int node)
{
  return _cityList[node].getNodeNumber();
}

double CityList::getCityLat(int node)
{
  return _cityList[node].getLatitudeY();
}

double CityList::getCityLon(int node)
{
  return _cityList[node].getLongitudeX();
}

void CityList::printAllCityNodes()
{
```

```cpp
  for (auto a : _cityList)
  {
    cout << "City/File: " << getFileName() << " Node: " << a.getNodeNumber()
         << " Lat: " << a.getLatitudeY() << " Lon: " << a.getLongitudeX()
         << endl;
  }
}

void CityList::printSpecCityNode(unsigned int node)
{
  /*Function assumes you typed in the node you want printed NOT the vector
    position. Hence node-1 to take into account vector position [0]*/
  cout << "City/File: " << getFileName() << " Node: "
       << _cityList[node-1].getNodeNumber() << " Lat: "
       << _cityList[node-1].getLatitudeY() << " Lon: "
       << _cityList[node-1].getLongitudeX() << endl;
}

void CityList::setFileName(string name)
{
  _fileName = name;
}

string CityList::getFileName()
{
  return _fileName;
}

  //Returns Euclidean distance between two cities (node 1 & node 2).
double CityList::distance(int first, int second)
{
  //x = long y = lat
  double d, x1, x2, y1, y2;
  x1 = getCityLon(first);
  x2 = getCityLon(second);
  y1 = getCityLat(first);
  y2 = getCityLat(second);

  d = sqrt(pow((x2-x1),2) + pow((y2-y1),2));

  return d;

}

int CityList::cityListSize()
{
  return _cityList.size();
}

int CityList::listVectorSpecific(int n)
{
  return _cityList[n].getNodeNumber();
}

void CityList::removeCityList(int node)
{
  _cityList.erase(_cityList.begin() + node);
}
```