ESS201: C++ Programming

Jaya Sreevalsan Nair *
International Institute of Information Technology, Bangalore

Term I: 2017-18 (Lab on 2017-10-17)

In this Lab, we will learn the concepts of association and composition using the `vector3d` class we have built so far.
Tasks:

1. Rewrite `struct point`, `struct line` and `struct triangle` from Lab01b to class interfaces with (private) data members. Rename them as `Point`, `Line`, and `Triangle`, respectively.

2. For this assignment, write a single C++ file with class interfaces and implementation for the aforementioned classes, and the `main` function.

3. Use your class interface and implementation for `vector3d` here. You may rename the class as `Vector3D`. Different from Lab01b, substitute `vector3d` as `Point*` for data members of `Line` and `Triangle`.

4. Add a new class called `Graph`, which is for implementing the graph data structure. It must have private data members for nodes and edges. Use STL vectors to manage the nodes and edges.
```
#include <vector>
class Graph {
private:
   std :: vector<Point*> nodes ;
   std :: vector<Line*> edges ;
```

5. Read the input for a graph and queries on the graph from a file, of the following format:
```
#number_of_points(N)
point-0-x  point-0-y  point-0-z
point-1-x  point-1-y  point-1-z
...
point-(N-1)-x point-(N-1)-y point-(N-1)-z
#number_of_lines(M)
point-index-0-for-line-0 point-index-1-for-line-0
point-index-0-for-line-1 point-index-1-for-line-1
...
point-index-0-for-line-(M-1) point-index-1-for-line-(M-1)
#number_of_queries (Q)
query-point-0-x query-point-0-y query-point-0-z
query-point-0-x query-point-0-y query-point-0-z
...
query-point-0-x query-point-0-y query-point-0-z
###end-of-file
```

*(jnair@iiitb.ac.in)

- There are three segments in the input file: data about points on the graph, data about connectivity of the points, and data about query points. The input file will contain data about N points, M lines, and Q queries. The values for these numbers are available in the file itself.
- Graph data: The inputs in line 1 and $(N+2)$ will be an integer per line prefixed by `#` symbol (the integers are values for N and M, respectively); in lines $2--(N+1)$ will be three real numbers (i.e. point coordinates) per line; and in lines $(N+2)--(N+M+2)$ will be two integers (i.e. point indices) per line. The point indices for lines will be a valid index between 0 and $(N-1)$.
- Query data: The input in line $(N+M+3)$ will be an integer prefixed by `#` symbol (the integer is the value for Q); and in lines $(N+M+4)$–$(N+M+Q+3)$ will be three real numbers (i.e. point coordinates for query points).
- Line $(N+M+Q+4)$ is the end of file indicated by `###`.

6. Construct all points as `Point*` and construct lines using existing `Point*`.

7. Store the points and lines in the `Graph`, in its private data members *nodes* and *lines*, respectively. The data in an input file will be stored in an instantiation of the `Graph`. Thus, the `Graph` can be viewed as the manifestation of the file.

8. Write methods by which you can query the following:

   - Given coordinates of a query point, return the `Point*` in `nodes` in the `Graph` that is the closest to the point.
   - Given a `Point*` present in the `Graph`, return all `Line*` in `edges` in the `Graph` that have the `Point*` as one of its data members.
   - Given a `Point*` present in the `Graph`, return the sum of lengths of all `Line*` in `edges` in the `Graph` that have the `Point*` as one of its data members.
   - Given a `Point*` present in the `Graph`, return a boolean if it is connected (i.e. if it is a member in `Line*` or not).

9. The output to your program must be Q lines with a real number followed by an integer per line. The real number in the i-th line should give the sum of the lengths of all `Line*` containing the `Point*` closest to the query point. The integer value in the i-th line should be the number of `Line*` containing the `Point*` closest to the query point.

   - In case of tie for the `Point*` closest to the query point, consider the `Point*` with minimum index in the vector `node` in the `Graph`.

- Contents in a sample input .txt file:

```
# 10
3.4 2.1 4.2
5.6 9.3 2.2
0.4 8.2 2.3
6.2 0.2 4.2
3.4 0.2 1.3
4.1 4.2 1.4
8.3 9.8 5.2
2.4 0.2 6.8
0.5 9.2 0.1
0.5 0.2 0.1
# 15
0 3
0 5
3 6
4 2
2 0
7 9
4 8
4 9
3 2
7 2
3 5
6 2
5 7
4 7
5 9
# 2
7.4 9.5 4.5
4.3 0.4 1.6
###
```