Algorithmen und Datenstrukturen

Übung 4 (Advanced Level 3)

Sebastian Stricker – If16b038  
Johannes Fessler – If16b042

# Funktionen & Variablen

## Global

#### int main (int argc, char \*argv[]);

#### int validate (int &argc, char\*\* argv);

#### bool fexists (const string& filename);

#### void printHelp ();

#### bool validateUserQuery(string start, string end, const map< string, Vertex\* > &m\_stations);

## Manager

### Private:

#### Algorithm \*alg;

#### map< string, Vertex\* > m\_stations;

#### vector<string> m\_lineNames;

#### Vertex\* newStation (string name);

#### void printRoute (vector<Edge\*> &route) const;

### Public:

#### Manager();

#### ~Manager();

#### void deleteDatastructure();

#### void readFile (string path);

#### void getQuery (string &start, string &end);

#### void findRoute (string &start, string &end);

## Algorithm

### Private:

#### Manager\* manager;

#### Vertex\* origin;

#### Vertex\* destination;

#### vector <Vertex\* > visitedStations;

#### list <Vertex\* > queue;

#### void checkPathLength(Vertex\* originVertex, Vertex\* vertexToCheck, int newDistance);

#### vector <Edge\* > finish (Vertex\* reachedGoal) const;

#### void addToQueue (Vertex\* addMeToQueue);

#### Edge\* turnPrevious(Vertex\* originVertex) const;

#### void initializeQuery (Vertex\* orig, Vertex\* dest);

### Public:

#### Algorithm (Manager \*man);

#### ~Algorithm();

#### vector <Edge\* > runQuery(Vertex\* orig, Vertex\* dest);

#### void reset ();

## Vertex

### Public:

#### Vertex (string name);

#### ~Vertex ();

#### vector<Edge\*> m\_edges;

#### Edge\* previous;

#### int pathLength;

#### string m\_stationName;

#### void changePrevious (Vertex\* originVertex);

## Edge

### Public:

#### Edge(int distance, int lineId, Vertex\* target);

#### ~Edge();

#### int m\_distance;

#### int m\_lineId;

#### Vertex\* m\_target = nullptr;

#### int getDistance (int currLineId) const;

## Validation

### Private:

### Public: