Карта на град

Структурата **„Карта на град“** е реализирана като граф, където кръстовищата са върхове, а улиците – ребра. Графът е представен като хеш таблица, където ключът е име на връх, а стойността – самият връх.

Има 3 класа – Vertex, Edge, Graph

class Vertex (върховете):

Всеки връх има **string name** – името на кръстовището, **vector<Edge> inStreets** – улиците, които влизат в кръстовището, **vector<Edge> outStreets** - улиците, които излизат от кръстовището, **bool closed** – дали кръстовището е затворено, поради ремонт. Освен това в този клас има функция **void addInStreet**, която добавя улица, влизаща в кръстовището, за което се извиква, и функция **void addOutStreet,** която добавя улица, излизаща от кръстовището, за което се извиква. Има имплементирани get функции за **name, inStreets, outStreets, closed**(**bool isClosed).** Имплементирана е функция **void changeStatusOfCrossroad**, която променя текущия статус на кръстовището, за което е извикана.

class Edge (ребрата):

Всяко ребро има **Vertex* start** – от кое кръстовище започва **, Vertex* end** – в кое кръстовище завършва**, unsigned length** – колко е дълга улицата. Имплементирани са get функции за **start end** и **length.**

class Graph(картата):

Картата е представена като хеш таблица (**unordered_map <string, Vertex> cityCrossroads).**

Имплементирани са следните функции:

- **void addCrossroad (string& crossroad)**
  Тази функция добавя връх (кръстовище) в графа, като първо проверява дали вече не сме го добавили.
- **void addStreet (string& start, string& end, unsigned length);**
  Тази функция добавя ребро (улица) в графа, като ако не съществуват кръстовищата start end, извиква **addCrossroad; length** е дължината на улицата.
- **void loadFile (string& url);**
  Това е функцията, четяща от файла. Тук имаме проверка, ако файлът не може да бъде намерен.
- **void closeCrossroad (string& crossroad);** Тази функция служи за затваряне на кръстовище. Използваме я при интерактивния режим на работа. В нея се извиква функцията **changeStatusOfCrossroad()** от класа **Vertex.**
- **void openCrossroad (string& crossroad);** Тази функция служи за отваряне на кръстовище. Използваме я при интерактивния режим на работа. В нея се извиква функцията **changeStatusOfCrossroad()** от класа **Vertex.**
- **vector<string> findClosed();** Отново функция, която се използва при интерактивния режим на работа. Тя връща всички затворени кръстовища

- **bool hasCycleHelper (string& start , string& cursor, vector<string>& visited);** Помощна рекурсивна функция, използвана при **bool hasCycle (string& start)** и **bool areAllAvailable (string& start).**

Функциите от изискванията за проекта:

1. Проверка дали има път между две зададени кръстовища,

- **bool hasRoute (string& start, string& end);** В тази функция сме имплементирали BFS алгоритъм.

2. Намиране на най-кратък път между две кръстовища

3. При наличието на затворени кръстовища се намира алтернативен най-кратък път между двете кръстовища. Точки 2 и 3 са обединени във функцията:

- **queue<string> shortestRoute (string& start, string& end);** Използваме алгоритъм на Дийкстра

4. Проверка дали при дадено кръстовище за начална точка е възможно да обиколим част от града и да се върнем там, от където сме тръгнали

- **bool hasCycle (string& start);** използваме малко изменено DFS(ако намерим един цикъл DFS-то спира). Тази функция използва помощната **bool hasCycleHelper (string& start , string& cursor, vector<string>& visited)**, която споменахме малко по-нагоре.
-

5. Проверка дали можем да направим пълна туристическа обиколка на всички улици без да минаваме по една и съща улица два пъти

- **queue<string> tour (string& start);** Тук проверяваме дали графът е Ойлеров

6. Проверка дали е възможно да стигнем от дадено кръстовище до всички останали

- **bool areAllAvailable (string& start);** И тук използваме помощната **bool hasCycleHelper (string& start , string& cursor, vector<string>& visited),** но и подаваме да търси цикъл към нещо, за което сме сигурни, че го няма. По този начин функцията връща 0, но нас ни интересува страничния ефект(vector с кръстовища; ако векторър си графът съдържат еднакъв брой кръстовища, значи имаме път към всички и функцията връща 1. В противен случай връща 0.)

7. Намиране на всички задънени улици

- **vector<string> deadends () const;** Връща се вектор със всички задъдени улици.

Реализиран е и бонуса с интерактивния режим на работа.