Spedycja

Generated by Doxygen 1.9.6

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 functions.cpp File Reference

```
#include "main.h"
```

Functions

- Graph LoadFromFile (const std::string &fileName)

  Funkcja odpowiedzialna za wczytanie grafu z pliku tekstowego. Funkcja otwiera plik o nazwie podanej w argumencie, następnie odczytuje każdy wiersz i za pomocą operatora ">>" wczytuje informacje o wierzchołkach i długości krawędzi, które następnie są dodawane do grafu.
- std::vector< std::string > ConstructShortestPath (const std::map< std::string, std::string > &previous, const std::string &start, const std::string &end)

  Funkcja odpowiedzialna za tworzenie wektora zawierającego najkrótszą ścieżkę między dwoma wierzchołkami w grafie. Funkcja działa poprzez przeglądanie mapy "previous" i tworzenie wektora zawierającego wierzchołki od końcowego do startowego. Następnie wektor jest odwracany i zwracany jako wynik funkcji.
- std::pair< double, std::vector< std::string > > dijkstra (const Graph &graph, const std::string &start, const std::string &end)

  Implementacja algorytmu Dijkstry, która pozwala na wyznaczenie najkrótszej drogi między dwoma wierzchołkami w grafie skierowanym lub nieskierowanym. Algorytm działa poprzez przeglądanie wszystkich wierzchołków w grafie i aktualizację odległości od wierzchołka startowego.
- void saveToFile (const std::string &fileName, const Graph &graph, const std::string &start)

  Funkcja odpowiedzialna za zapis wyznaczonej najkrótszej drogi z danego miasta. Funkcja tworzy plik, następnie dla każdego miasta poza miastem początkowym (start), wyznaczana jest najkrótsza droga za pomocą algorytmu Dijkstry, a następnie zapisywana jest do pliku w formie ciągu tekstowego.
- std::vector< std::string > userData (int argc, char *argv[])

  Funkcja odpowiedzialna za przetwarzanie argumentów podanych przez użytkownika podczas uruchomienia programu. Funkcja iteruje przez argumenty i sprawdza, czy są one odpowiedniej postaci (np. "-i" i kolejny argument jest nazwą pliku wejściowego). Jeśli brakuje jakiegoś argumentu lub są one niepoprawne, funkcja wyświetla informację o poprawnym użyciu i zwraca pusty wektor.
- bool correctCity (Graph &graph, std::string &city)

  Funkcja odpowiedzialna za sprawdzenie poprawności miasta podanego przez użytkownika. Funkcja zamienia pierwszą literę miasta na wielką literę, następnie sprawdza czy miasto znajduje się w grafie za pomocą metody count().

2.1.1 Function Documentation

2.1.1.1 ConstructShortestPath()

```
std::vector< std::string > ConstructShortestPath (
            const std::map< std::string, std::string > & previous,
            const std::string & start,
            const std::string & end )
```

Funkcja odpowiedzialna za tworzenie wektora zawierającego najkrótszą ścieżkę między dwoma wierzchołkami w grafie. Funkcja działa poprzez przeglądanie mapy "previous" i tworzenie wektora zawierającego wierzchołki od końcowego do startowego. Następnie wektor jest odwracany i zwracany jako wynik funkcji.

Parameters

previous	Mapa która zawiera informacje o poprzedniku dla każdego wierzchołka w grafie
start	Nazwa początkowego wierzchołka w grafie
end	Nazwa końcowego wierzchołka w grafie

Returns

Vector z miastami w dobrej kolejności.

2.1.1.2 correctCity()

```
bool correctCity (
            Graph & graph,
            std::string & city )
```

Funkcja odpowiedzialna za sprawdzenie poprawności miasta podanego przez użytkownika. Funkcja zamienia pierwszą literę miasta na wielką literę, następnie sprawdza czy miasto znajduje się w grafie za pomocą metody count().

Parameters

graph	Graf reprezentujący połączenie miast
city	Miasto wpisane przez uzytkownika

Returns

Prawde jeśli miasto zostało poprawnie wpisane

2.1.1.3 dijkstra()

```
std::pair< double, std::vector< std::string > > dijkstra (
            const Graph & graph,
            const std::string & start,
            const std::string & end )
```

Implementacja algorytmu Dijkstry, która pozwala na wyznaczenie najkrótszej drogi między dwoma wierzchołkami w grafie skierowanym lub nieskierowanym. Algorytm działa poprzez przeglądanie wszystkich wierzchołków w grafie i aktualizację odległości od wierzchołka startowego.

Parameters

graph	Graf reprezentujący połączenie miast
start	Nazwa początkowego wierzchołka w grafie
end	Nazwa końcowego wierzchołka w grafie

Returns

Najkrótsza trasa i jej dystans, w postaci pary odległości i vectora z ułożonymi miastami.

2.1.1.4 LoadFromFile()

```
Graph LoadFromFile (
            const std::string & fileName )
```

Funkcja odpowiedzialna za wczytanie grafu z pliku tekstowego. Funkcja otwiera plik o nazwie podanej w argumencie, następnie odczytuje każdy wiersz i za pomocą operatora ">>" wczytuje informacje o wierzchołkach i długości krawędzi, które następnie są dodawane do grafu.

Parameters

fileName	Nazwa pliku wejściowego

Returns

Załadowany graf danymi z pliku wejściowego.

2.1.1.5 saveToFile()

```
void saveToFile (
            const std::string & fileName,
            const Graph & graph,
            const std::string & start )
```

Funkcja odpowiedzialna za zapis wyznaczonej najkrótszej drogi z danego miasta. Funkcja tworzy plik, następnie dla każdego miasta poza miastem początkowym (start), wyznaczana jest najkrótsza droga za pomocą algorytmu Dijkstry, a następnie zapisywana jest do pliku w formie ciągu tekstowego.

Parameters

fileName	Nazwa pliku wyjśicowego
graph	Graf reprezentujący połączenie miast
start	Nazwa początkowego wierzchołka w grafie.

2.1.1.6 userData()

```
std::vector< std::string > userData (
            int argc,
            char * argv[] )
```

Funkcja odpowiedzialna za przetwarzanie argumentów podanych przez użytkownika podczas uruchomienia programu. Funkcja iteruje przez argumenty i sprawdza, czy są one odpowiedniej postaci (np. "-i" i kolejny argument jest nazwą pliku wejściowego). Jeśli brakuje jakiegoś argumentu lub są one niepoprawne, funkcja wyświetla informację o poprawnym użyciu i zwraca pusty wektor.

Parameters

argc	Liczba argumentów przekazanych do programu
argv	Tablica łańcuchów znaków, reprezentujących argumenty przekazane do programu

Returns

Vector z danymi wyprowadzonymi przez użytkownika

2.2 main.cpp File Reference

```
#include "main.h"
```

Functions

- int main (int argc, char ∗argv[ ])

2.2.1 Function Documentation

2.2.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

2.3 main.h File Reference

```
#include <iostream>
#include <map>
#include <set>
#include <fstream>
#include <list>
#include <string>
#include <sstream>
#include <algorithm>
#include <vector>
#include <limits>
#include <queue>
#include <cctype>
```

Typedefs

- typedef std::map< std::string, std::set< std::pair< std::string, double > > > Graph

  Definicja Grafu. Składa się on z mapy, której kluczem jest nazwa miasta. Kolejną częścią jest set składający się z pary, która przechowuje wszystkie połączenia wraz z jej odległościami do danego miasta.

Functions

- Graph LoadFromFile (const std::string &fileName)

  Funkcja odpowiedzialna za wczytanie grafu z pliku tekstowego. Funkcja otwiera plik o nazwie podanej w argumencie, następnie odczytuje każdy wiersz i za pomocą operatora ">>" wczytuje informacje o wierzchołkach i długości krawędzi, które następnie są dodawane do grafu.
- std::vector< std::string > ConstructShortestPath (const std::map< std::string, std::string > &previous, const std::string &start, const std::string &end)

  Funkcja odpowiedzialna za tworzenie wektora zawierającego najkrótszą ścieżkę między dwoma wierzchołkami w grafie. Funkcja działa poprzez przeglądanie mapy "previous" i tworzenie wektora zawierającego wierzchołki od końcowego do startowego. Następnie wektor jest odwracany i zwracany jako wynik funkcji.
- std::pair< double, std::vector< std::string > > dijkstra (const Graph &graph, const std::string &start, const std::string &end)

  Implementacja algorytmu Dijkstry, która pozwala na wyznaczenie najkrótszej drogi między dwoma wierzchołkami w grafie skierowanym lub nieskierowanym. Algorytm działa poprzez przeglądanie wszystkich wierzchołków w grafie i aktualizację odległości od wierzchołka startowego.
- void saveToFile (const std::string &fileName, const Graph &graph, const std::string &start)

  Funkcja odpowiedzialna za zapis wyznaczonej najkrótszej drogi z danego miasta. Funkcja tworzy plik, następnie dla każdego miasta poza miastem początkowym (start), wyznaczana jest najkrótsza droga za pomocą algorytmu Dijkstry, a następnie zapisywana jest do pliku w formie ciągu tekstowego.
- std::vector< std::string > userData (int argc, char *argv[])

  Funkcja odpowiedzialna za przetwarzanie argumentów podanych przez użytkownika podczas uruchomienia programu. Funkcja iteruje przez argumenty i sprawdza, czy są one odpowiedniej postaci (np. "-i" i kolejny argument jest nazwą pliku wejściowego). Jeśli brakuje jakiegoś argumentu lub są one niepoprawne, funkcja wyświetla informację o poprawnym użyciu i zwraca pusty wektor.
- bool correctCity (Graph &graph, std::string &city)

  Funkcja odpowiedzialna za sprawdzenie poprawności miasta podanego przez użytkownika. Funkcja zamienia pierwszą literę miasta na wielką literę, następnie sprawdza czy miasto znajduje się w grafie za pomocą metody count().

2.3.1 Typedef Documentation

2.3.1.1 Graph

```
typedef std::map<std::string, std::set<std::pair<std::string, double> > > Graph
```

Definicja Grafu. Składa się on z mapy, której kluczem jest nazwa miasta. Kolejną częścią jest set składający się z pary, która przechowuje wszystkie połączenia wraz z jej odległościami do danego miasta.

2.3.2 Function Documentation

2.3.2.1 ConstructShortestPath()

```
std::vector< std::string > ConstructShortestPath (
            const std::map< std::string, std::string > & previous,
            const std::string & start,
            const std::string & end )
```

Funkcja odpowiedzialna za tworzenie wektora zawierającego najkrótszą ścieżkę między dwoma wierzchołkami w grafie. Funkcja działa poprzez przeglądanie mapy "previous" i tworzenie wektora zawierającego wierzchołki od końcowego do startowego. Następnie wektor jest odwracany i zwracany jako wynik funkcji.

Parameters

previous	Mapa która zawiera informacje o poprzedniku dla każdego wierzchołka w grafie
start	Nazwa początkowego wierzchołka w grafie
end	Nazwa końcowego wierzchołka w grafie

Returns

Vector z miastami w dobrej kolejności.

2.3.2.2 correctCity()

```
bool correctCity (
            Graph & graph,
            std::string & city )
```

Funkcja odpowiedzialna za sprawdzenie poprawności miasta podanego przez użytkownika. Funkcja zamienia pierwszą literę miasta na wielką literę, następnie sprawdza czy miasto znajduje się w grafie za pomocą metody count().

Parameters

graph	Graf reprezentujący połączenie miast
city	Miasto wpisane przez uzytkownika

Returns

Prawde jeśli miasto zostało poprawnie wpisane

2.3.2.3 dijkstra()

```
std::pair< double, std::vector< std::string > > dijkstra (
            const Graph & graph,
            const std::string & start,
            const std::string & end )
```

Implementacja algorytmu Dijkstry, która pozwala na wyznaczenie najkrótszej drogi między dwoma wierzchołkami w grafie skierowanym lub nieskierowanym. Algorytm działa poprzez przeglądanie wszystkich wierzchołków w grafie i aktualizację odległości od wierzchołka startowego.

Parameters

graph	Graf reprezentujący połączenie miast
start	Nazwa początkowego wierzchołka w grafie
end	Nazwa końcowego wierzchołka w grafie

Returns

Najkrótsza trasa i jej dystans, w postaci pary odległości i vectora z ułożonymi miastami.

2.3.2.4 LoadFromFile()

```
Graph LoadFromFile (
            const std::string & fileName )
```

Funkcja odpowiedzialna za wczytanie grafu z pliku tekstowego. Funkcja otwiera plik o nazwie podanej w argumencie, następnie odczytuje każdy wiersz i za pomocą operatora ">>" wczytuje informacje o wierzchołkach i długości krawędzi, które następnie są dodawane do grafu.

Parameters

fileName	Nazwa pliku wejściowego

Returns

Załadowany graf danymi z pliku wejściowego.

2.3.2.5 saveToFile()

```
void saveToFile (
            const std::string & fileName,
            const Graph & graph,
            const std::string & start )
```

Funkcja odpowiedzialna za zapis wyznaczonej najkrótszej drogi z danego miasta. Funkcja tworzy plik, następnie dla każdego miasta poza miastem początkowym (start), wyznaczana jest najkrótsza droga za pomocą algorytmu Dijkstry, a następnie zapisywana jest do pliku w formie ciągu tekstowego.

Parameters

fileName	Nazwa pliku wyjściowego
graph	Graf reprezentujący połączenie miast
start	Nazwa początkowego wierzchołka w grafie.

2.3.2.6 userData()

```
std::vector< std::string > userData (
            int argc,
            char * argv[] )
```

Funkcja odpowiedzialna za przetwarzanie argumentów podanych przez użytkownika podczas uruchomienia programu. Funkcja iteruje przez argumenty i sprawdza, czy są one odpowiedniej postaci (np. "-i" i kolejny argument jest nazwą pliku wejściowego). Jeśli brakuje jakiegoś argumentu lub są one niepoprawne, funkcja wyświetla informację o poprawnym użyciu i zwraca pusty wektor.

Parameters

argc	Liczba argumentów przekazanych do programu
argv	Tablica łańcuchów znaków, reprezentujących argumenty przekazane do programu

Returns

Vector z danymi wyprowadzonymi przez użytkownika

2.4 main.h

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include <map>
00003 #include <set>
00004 #include <fstream>
00005 #include <list>
00006 #include <string>
00007 #include <sstream>
00008 #include <algorithm>
00009 #include <vector>
00010 #include <limits>
00011 #include <queue>
```

```
00012 #include <cctype>
00013 #pragma once
00014
00019 typedef std::map<std::string, std::set<std::pair<std::string, double»> Graph;
00020
00026 Graph LoadFromFile(const std::string& fileName);
00027
00035 std::vector<std::string> ConstructShortestPath(
00036         const std::map<std::string, std::string>& previous,
00037         const std::string& start,
00038         const std::string& end);
00039
00047 std::pair<double, std::vector<std::string» dijkstra(
00048         const Graph& graph,
00049         const std::string& start,
00050         const std::string& end);
00051
00058 void saveToFile(
00059         const std::string& fileName,
00060         const Graph& graph,
00061         const std::string& start);
00062
00069 std::vector<std::string> userData (
00070         int argc,
00071         char* argv[]);
00072
00079 bool correctCity(
00080         Graph& graph,
00081         std::string& city);
```

Index