

Mapa

Wygenerowano przez Doxygen 1.8.14

Spis treści

1	Indeks klas	1
1.1	Lista klas	1
2	Indeks plików	3
2.1	Lista plików	3
3	Dokumentacja klas	5
3.1	Dokumentacja struktury existindex	5
3.1.1	Opis szczegółowy	5
3.1.2	Dokumentacja atrybutów składowych	5
3.1.2.1	correctdist	5
3.1.2.2	existA	6
3.1.2.3	existB	6
3.1.2.4	locationptrA	6
3.1.2.5	locationptrB	6
3.2	Dokumentacja struktury lcities	6
3.2.1	Opis szczegółowy	7
3.2.2	Dokumentacja atrybutów składowych	7
3.2.2.1	city	7
3.2.2.2	pNextCity	7
3.2.2.3	pNextDist	7
3.3	Dokumentacja struktury ldirect	7
3.3.1	Opis szczegółowy	8
3.3.2	Dokumentacja atrybutów składowych	8

3.3.2.1	lvNext	8
3.3.2.2	vertex	8
3.4	Dokumentacja struktury lcities::ldist	8
3.4.1	Opis szczegółowy	8
3.4.2	Dokumentacja atrybutów składowych	9
3.4.2.1	dist	9
3.4.2.2	pNextCity	9
3.4.2.3	pNextDist	9
3.5	Dokumentacja struktury pathtab	9
3.5.1	Opis szczegółowy	9
3.5.2	Dokumentacja atrybutów składowych	10
3.5.2.1	nextpathtab	10
3.5.2.2	pvertex	10
3.5.2.3	sdist	10
3.5.2.4	vertex	10
3.5.2.5	visited	10
4	Dokumentacja plików	11
4.1	Dokumentacja pliku Cleanse.cpp	11
4.1.1	Opis szczegółowy	11
4.1.2	Dokumentacja funkcji	12
4.1.2.1	clean()	12
4.1.2.2	cleanpath()	12
4.1.2.3	cleanroute()	12
4.1.2.4	complexclean()	13
4.2	Dokumentacja pliku Cleanse.h	13
4.2.1	Opis szczegółowy	14
4.2.2	Dokumentacja funkcji	14
4.2.2.1	clean()	14
4.2.2.2	cleanpath()	14
4.2.2.3	cleanroute()	15

4.2.2.4	complexclean()	15
4.3	Dokumentacja pliku Loading.cpp	16
4.3.1	Opis szczegółowy	16
4.3.2	Dokumentacja funkcji	17
4.3.2.1	addcity()	17
4.3.2.2	addroad()	17
4.3.2.3	existroad()	18
4.3.2.4	loadroad()	18
4.4	Dokumentacja pliku Loading.h	19
4.4.1	Opis szczegółowy	19
4.4.2	Dokumentacja funkcji	20
4.4.2.1	addcity()	20
4.4.2.2	addroad()	20
4.4.2.3	existroad()	21
4.4.2.4	loadroad()	21
4.5	Dokumentacja pliku Mapa.cpp	22
4.5.1	Opis szczegółowy	22
4.5.2	Dokumentacja funkcji	22
4.5.2.1	main()	23
4.6	Dokumentacja pliku Mapheader.h	23
4.6.1	Dokumentacja typów wyliczanych	24
4.6.1.1	errorid	24
4.7	Dokumentacja pliku pathfinding.cpp	24
4.7.1	Opis szczegółowy	25
4.7.2	Dokumentacja funkcji	26
4.7.2.1	allroadexist()	26
4.7.2.2	allvisited()	26
4.7.2.3	existroadAtoB()	27
4.7.2.4	existV()	27
4.7.2.5	findiname()	28

4.7.2.6	findinpath()	28
4.7.2.7	findinvisited()	29
4.7.2.8	findmin()	29
4.7.2.9	findshortAB()	30
4.7.2.10	getweight()	31
4.7.2.11	initialize()	31
4.7.2.12	interpret()	32
4.7.2.13	weightless()	32
4.7.2.14	writeroute()	33
4.8	Dokumentacja pliku pathfinding.h	33
4.8.1	Opis szczegółowy	34
4.8.2	Dokumentacja funkcji	34
4.8.2.1	allroadexist()	34
4.8.2.2	allvisited()	35
4.8.2.3	existroadAtoB()	35
4.8.2.4	existV()	36
4.8.2.5	findiname()	36
4.8.2.6	findinpath()	37
4.8.2.7	findinvisited()	37
4.8.2.8	findmin()	38
4.8.2.9	findshortAB()	39
4.8.2.10	getweight()	39
4.8.2.11	initialize()	40
4.8.2.12	interpret()	40
4.8.2.13	weightless()	41
4.8.2.14	writeroute()	41
4.9	Dokumentacja pliku Service.cpp	42
4.9.1	Opis szczegółowy	42
4.9.2	Dokumentacja funkcji	43
4.9.2.1	allexcep()	43

4.9.2.2	core()	43
4.9.2.3	correctparams()	44
4.9.2.4	doparameters()	44
4.9.2.5	mainexception()	45
4.9.2.6	noroad()	46
4.9.2.7	printhelp()	46
4.10	Dokumentacja pliku Service.h	46
4.10.1	Opis szczegółowy	47
4.10.2	Dokumentacja funkcji	47
4.10.2.1	allexcep()	47
4.10.2.2	core()	48
4.10.2.3	correctparams()	48
4.10.2.4	doparameters()	49
4.10.2.5	mainexception()	50
4.10.2.6	noroad()	50
4.10.2.7	printhelp()	51
4.11	Dokumentacja pliku Valid.cpp	51
4.11.1	Opis szczegółowy	51
4.11.2	Dokumentacja funkcji	51
4.11.2.1	charonly()	51
4.12	Dokumentacja pliku Valid.h	52
4.12.1	Opis szczegółowy	52
4.12.2	Dokumentacja funkcji	52
4.12.2.1	charonly()	52
Indeks		55

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

existindex	Struktura dla funkcji existroad, z informacjami o (nie)istnieniu miast oraz wskaźnikami na nie. Struktura jest uzupełniana funkcją existroad, by stanowić podstawy do uzupełniania listy przy ładowaniu miast lub nie	5
lcities	Struktura mapy złożonej z dróg jednokierunkowych. lcities to lista list. Struktura reprezentuje miasto. Miasta tworzą listę jednokierunkową. Każdemu miastu przypisana jest droga lub lista dróg wychodzących do innych miast	6
ldirect	Struktura drogi z A do B. Struktura wskazuje na elementy tablicy pathtab (pathtab), przez które należy iść aby dotrzeć z miasta A do miasta B, a dokładniej z miasta B do miasta A, ponieważ funkcje poruszają się z vertex do pvertex poczynając od vertex wskazujący na miasto B	7
lcities::ldist	Struktura dystansów w mapie	8
pathtab	Struktura do podawania wag odległościowych i poprawnego wykonania algorytmu Dijkstry, dalej zwana pathtab	9

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Cleanse.cpp	
Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu	11
Cleanse.h	
Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu	13
Loading.cpp	
Plik zawiera funkcje uzytkowe do dzialan na liscie pName, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejscowego	16
Loading.h	
Plik zawiera funkcje uzytkowe do dzialan na liscie pName, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejscowego	19
Mapa.cpp	22
Mapheader.h	23
pathfinding.cpp	
Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje drogi z A do B	24
pathfinding.h	
Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje drogi z A do B	33
Service.cpp	
Zawiera funkcje dla <code>main()</code> , do obsługi wprowadzanych danych, czy to w postaci argumentów linii poleceń, czy danych bezpośrednio z pliku	42
Service.h	
Zawiera funkcje dla <code>main()</code> , do obsługi wprowadzanych danych, czy to w postaci argumentów linii poleceń, czy danych bezpośrednio z pliku	46
Valid.cpp	
Zawiera funkcje to tworzenia stringów tylko ze znakami dozwolonymi	51
Valid.h	
Zawiera funkcje to tworzenia stringów tylko ze znakami dozwolonymi	52

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury existindex

Struktura dla funkcji existroad, z informacjami o (nie)istnieniu miast oraz wskaźnikach na nie. Struktura jest uzupełniana funkcją existroad, by stanowić podstawy do uzupełniania listy przy ładowaniu miast lub nie.

```
#include <Mapheader.h>
```

Atrybuty publiczne

- bool `existA`
- bool `existB`
- int `correctdist`
- `lcities` * `locationptrA` = `nullptr`
- `lcities` * `locationptrB` = `nullptr`

3.1.1 Opis szczegółowy

Struktura dla funkcji existroad, z informacjami o (nie)istnieniu miast oraz wskaźnikach na nie. Struktura jest uzupełniana funkcją existroad, by stanowić podstawy do uzupełniania listy przy ładowaniu miast lub nie.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 `correctdist`

```
int existindex::correctdist
```

Czy dystans między miastami A i B jest poprawny

Ostrzeżenie

Wartość tego elementu jest ściśle określona.

"0" - jeśli taki dystans nie istnieje

"1" - jeśli dystans podany z pliku jest taki jak już był podany wcześniej

"2" - jeśli wcześniej podano inny dystans z A do B

3.1.2.2 existA

```
bool existindex::existA
```

Czy istnieje miasto A

3.1.2.3 existB

```
bool existindex::existB
```

Czy istnieje miasto B

3.1.2.4 locationptrA

```
lcities* existindex::locationptrA = nullptr
```

nodist=0=nie istnieje, okdist=1=taki sam, wasdifferentdist=2=był inny Wskaznik na miasto A jeśli istnieje, nullptr jeśli nie istnieje

3.1.2.5 locationptrB

```
lcities* existindex::locationptrB = nullptr
```

Wskaznik na miasto B jeśli istnieje, nullptr jeśli nie istnieje

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Mapheader.h](#)

3.2 Dokumentacja struktury lcities

Struktura mapy złożonej z dróg jednokierunkowych. Lcities to lista list. Struktura reprezentuje miasto. Miasta tworzą listę jednokierunkową. Każdemu miastu przypisana jest droga lub lista dróg wychodzących do innych miast.

```
#include <Mapheader.h>
```

Komponenty

- struct [ldist](#)

Struktura dystansów w mapie.

Atrybuty publiczne

- string [city](#)
- [lcities](#) * [pNextCity](#) = nullptr
- [ldist](#) * [pNextDist](#) = nullptr

3.2.1 Opis szczegółowy

Struktura mapy złożonej z dróg jednokierunkowych. lcities to lista list. Struktura reprezentuje miasto. Miasta tworzą listę jednokierunkową. Każdemu miastu przypisana jest droga lub lista dróg wychodzących do innych miast.

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 city

```
string lcities::city
```

nazwa miasta

3.2.2.2 pNextCity

```
lcities* lcities::pNextCity = nullptr
```

wskaznik na następne miasto w liście

3.2.2.3 pNextDist

```
ldist* lcities::pNextDist = nullptr
```

wskaznik na pierwszą drogę wychodzącą z listy

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Mapheader.h](#)

3.3 Dokumentacja struktury ldirect

Struktura drogi z A do B. Struktura wskazuje na elementy tablicy pathtab (pathtab), przez które należy iść aby dotrzeć z miasta A do miasta B, a dokładniej z miasta B do miasta A, ponieważ funkcje poruszają się z vertex do pvertex poczynając od vertex wskazującego na miasto B.

```
#include <Mapheader.h>
```

Atrybuty publiczne

- [pathtab](#) * [vertex](#) = nullptr
- [ldirect](#) * [lvNext](#) = nullptr

3.3.1 Opis szczegółowy

Struktura drogi z A do B. Struktura wskazuje na elementy tablicy pathtab (pathtab), przez które należy iść aby dotrzeć z miasta A do miasta B, a dokładniej z miasta B do miasta A, ponieważ funkcje poruszają się z vertex do pvertex poczynając od vertex wskazujący na miasto B.

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 lvNext

```
ldirect* ldirect::lvNext = nullptr
```

Wskaznik na następny element listy

3.3.2.2 vertex

```
pathtab* ldirect::vertex = nullptr
```

Wskaznik na miasto X

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Mapheader.h](#)

3.4 Dokumentacja struktury lcities::ldist

Struktura dystansów w mapie.

```
#include <Mapheader.h>
```

Atrybuty publiczne

- double `dist` = 0
- `lcities` * `pNextCity` = nullptr
- `ldist` * `pNextDist` = nullptr

3.4.1 Opis szczegółowy

Struktura dystansów w mapie.

Ostrzeżenie

Struktura dalej zwana mapą. `ldist` zdefiniowane w `lcities`, to struktura zawierająca wartość, interpretowana jako odległość lub droga o danym dystansie z miasta wskazującego na ten element do miasta, na który element wskazuje. Elementy tworzą listę jednokierunkową, która jest podwieszona do jakiegoś miasta.

W tabeli X oznacza zupełnie dowolne miasto/miasta o n-tym/n-tych indeksie/indeksach.

"...->V.X." to wskaznik na listę w dół tabeli

"...->>>.." to wskaznik na następny element w wierszu tabeli.

Struktura rzeczywista może się znacząco różnić od przykładowej w niektórych przypadkach co do wartości, ale nie co do konceptu.

3.4.2 Dokumentacja atrybutów składowych

3.4.2.1 dist

```
double lcities::ldist::dist = 0
```

odleglosc drogi miedzy miastami

3.4.2.2 pNextCity

```
lcities* lcities::ldist::pNextCity = nullptr
```

wskaznik na miasto w do ktorego prowadzi droga

3.4.2.3 pNextDist

```
ldist* lcities::ldist::pNextDist = nullptr
```

wskaznik na nastepna droge wychodzaca z miasta

Dokumentacja dla tej struktury zostala wygenerowana z pliku:

- [Mapheader.h](#)

3.5 Dokumentacja struktury pathtab

Struktura do podawania wag odleglosciowych i poprawnego wykonania algorytmu Dijkstry, dalej zwana pathtab.

```
#include <Mapheader.h>
```

Atrybuty publiczne

- `lcities * vertex` = nullptr
- `double sdist` = 0
- `lcities * pvertex` = nullptr
- `bool visited` = false
- `pathtab * nextpathtab`

3.5.1 Opis szczegółowy

Struktura do podawania wag odleglosciowych i poprawnego wykonania algorytmu Dijkstry, dalej zwana pathtab.

Ostrzeżenie

tablica pathtab jest tworzona dla wszystkich istniejących wierzchołków niezależnie od jakiegokolwiek zmiennej, gdyż nie wiemy ile wierzchołków (ani które) musi znaleźć się na trasie.

3.5.2 Dokumentacja atrybutów składowych

3.5.2.1 nextpathtab

```
pathtab* pathtab::nextpathtab
```

Wskaźnik na kolejny element listy/tabeli pathtab

3.5.2.2 pvertex

```
lcities* pathtab::pvertex = nullptr
```

Wskaźnik na wierzchołek z którego przychodzi Dijkstra

3.5.2.3 sdist

```
double pathtab::sdist = 0
```

Waga, czyli odległość do miasta X z miasta A, przez miasta Y,Z...

3.5.2.4 vertex

```
lcities* pathtab::vertex = nullptr
```

Wskaźnik na wierzchołek

3.5.2.5 visited

```
bool pathtab::visited = false
```

Czy było odwiedzone

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Mapheader.h](#)

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku Cleanse.cpp

Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu.

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
#include "pathfinding.h"
#include "Loading.h"
```

Funkcje

- void [clean](#) ([lcities](#) *&pName)
Funkcja usuwa liste obiektow typu lcities, z lista podrzedna obiektow ldist wlacznie.
- void [cleanroute](#) ([ldirect](#) *&route)
Funkcja usuwa strukture typu ldirect.
- void [cleanpath](#) ([pathtab](#) *&path)
Funkcja usuwa liste elementow typu pathtab.
- void [complexclean](#) ([lcities](#) *&pName, bool pn, [pathtab](#) *&path, bool pt, [ldirect](#) *&route, bool rt)
Wywołuje usuwanie podanych struktur dwokrotnie, po strukturze nalezy podac 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominięciu dla false.

4.1.1 Opis szczegółowy

Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu.

Autor

Tomasz Skowron

Zobacz również

[Cleanse.h](#)

4.1.2 Dokumentacja funkcji

4.1.2.1 clean()

```
void clean (
    lcities *& pName )
```

Funkcja usuwa liste obiektow typu lcities, z lista podzredna obiektow ldist wlacznie.

Parametry

<i>pName</i>	- wskaznik na strukture do usuniecia
--------------	--------------------------------------

Zwraca

void

4.1.2.2 cleanpath()

```
void cleanpath (
    pathtab *& path )
```

Funkcja usuwa liste elementow typu pathtab.

Parametry

<i>path</i>	- wskaznik na strukture do usuniecia
-------------	--------------------------------------

Zwraca

void

4.1.2.3 cleanroute()

```
void cleanroute (
    ldirect *& route )
```

Funkcja usuwa strukture typu ldirect.

Parametry

<i>route</i>	- wskaznik na strukture do usuniecia
--------------	--------------------------------------

Zwraca

void

4.1.2.4 complexclean()

```
void complexclean (
    lcities *& pName,
    bool pn,
    pathtab *& path,
    bool pt,
    ldirect *& route,
    bool rt )
```

Wywołuje usuwanie podanych struktur dwukrotnie, po strukturze należy podać 1/0 lub true/false co decyduje o usuwaniu struktury dla true i pominięciu dla false.

Ostrzeżenie

Funkcja jest tylko medium do wywołania funkcji czyszczących poszczególne struktury. Wywołuje `clean()`, `cleanroute()`, `cleanpath()` w zależności od parametrów typu bool.

Parametry

<i>pName</i>	- wskaźnik na strukturę <code>lcities</code>
<i>pn</i>	- 1 usun, 0 nie usuwaj
<i>path</i>	- wskaźnik na strukturę <code>lcities</code>
<i>pt</i>	- 1 usun, 0 nie usuwaj
<i>route</i>	- wskaźnik na strukturę <code>lcities</code>
<i>rt</i>	- 1 usun, 0 nie usuwaj

Zwraca

void

4.2 Dokumentacja pliku Cleanse.h

Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu.

```
#include "Mapheader.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- void `clean` (`lcities` *&pName)
Funkcja usuwa liste obiektow typu lcities, z lista podrzedna obiektow ldist wlacznie.
- void `cleanroute` (`ldirect` *&route)
Funkcja usuwa strukture typu ldirect.
- void `cleanpath` (`pathtab` *&path)
Funkcja usuwa liste elementow typu pathtab.
- void `complexclean` (`lcities` *&pName, bool pn, `pathtab` *&path, bool pt, `ldirect` *&route, bool rt)
Wywołuje usuwanie podanych struktur dwukrotnie, po strukturze należy podać 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominięciu dla false.

4.2.1 Opis szczegółowy

Plik zawiera funkcje do usuwania struktur powstałych w wyniku działania programu.

Autor

Tomasz Skowron

Zobacz również

[Cleanse.cpp](#)

4.2.2 Dokumentacja funkcji

4.2.2.1 `clean()`

```
void clean (
    lcities *& pName )
```

Funkcja usuwa liste obiektow typu lcities, z lista podrzedna obiektow ldist wlacznie.

Parametry

<code>pName</code>	- wskaznik na strukture do usuniecia
--------------------	--------------------------------------

Zwraca

void

4.2.2.2 `cleanpath()`

```
void cleanpath (
    pathtab *& path )
```

Funkcja usuwa liste elementow typu pathtab.

Parametry

<i>path</i>	- wskaznik na strukture do usuniecia
-------------	--------------------------------------

Zwraca

void

4.2.2.3 cleanroute()

```
void cleanroute (
    ldirect *& route )
```

Funkcja usuwa strukture typu ldirect.

Parametry

<i>route</i>	- wskaznik na strukture do usuniecia
--------------	--------------------------------------

Zwraca

void

4.2.2.4 complexclean()

```
void complexclean (
    lcities *& pName,
    bool pn,
    pathtab *& path,
    bool pt,
    ldirect *& route,
    bool rt )
```

Wywołuje usuwanie podanych struktur dwukrotnie, po strukturze należy podać 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominięciu dla false.

Ostrzeżenie

Funkcja jest tylko medium do wywołania funkcji czyszczących poszczególne struktury. Wywołuje [clean\(\)](#), [cleanroute\(\)](#), [cleanpath\(\)](#) w zależności od parametrów typu bool.

Parametry

<i>pName</i>	- wskaźnik na strukturę <i>lcities</i>
<i>pn</i>	- 1 usun, 0 nie usuwaj
<i>path</i>	- wskaźnik na strukturę <i>lcities</i>
<i>pt</i>	- 1 usun, 0 nie usuwaj
<i>route</i>	- wskaźnik na strukturę <i>lcities</i>
<i>rt</i>	- 1 usun, 0 nie usuwaj

Zwraca

void

4.3 Dokumentacja pliku Loading.cpp

Plik zawiera funkcje uzytkowe do dzialan na liscie *pName*, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejsciowego.

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
#include "pathfinding.h"
#include "Loading.h"
```

Funkcje

- [existindex existroad](#) (*lcities* **&pName*, const string &cityA, double distance, const string &cityB)
Funkcja odnajduje miasta i zwraca strukture existindex ze wskaźnikami na szukane elementy o ile istniały.
- void [addcity](#) (*lcities* **&pName*, const string &cityA)
*Funkcja dodaje nowe miasto do struktury mapy, tzn *lcities*.*
- void [addroad](#) (*lcities* **&pName*, double dist, [existindex](#) &cityloc)
Funkcja dodaje droge, tzn dystans miedzy miastami A B. Funkcja wymaga podania struktury existindex by uproszczyc proces poszukiwania wskaźnikow na odpowiednie miasta, te zas sa dodawane do struktury existindex przez funkcje existroad.
- void [loadroad](#) (*lcities* **&pName*, const string &infilename)
Funkcja jest medium do tworzenia struktury mapy. Funkcja pobiera linie z pliku i sprawdzając ich poprawność tworzy strukture mapy.

4.3.1 Opis szczegółowy

Plik zawiera funkcje uzytkowe do dzialan na liscie *pName*, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejsciowego.

Autor

Tomasz Skowron

Zobacz również

[Loading.cpp](#)

4.3.2 Dokumentacja funkcji

4.3.2.1 addcity()

```
void addcity (
    lcities *& pName,
    const string & cityA )
```

Funkcja dodaje nowe miasto do struktury mapy, tzn lcities.

Ostrzeżenie

Funkcja nie sprawdza czy miasto istnieje i dla stabilności oraz poprawności działania programu powinno nie istnieć w strukturze.

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>cityA</i>	- nazwa miasta A

Zwraca

void

4.3.2.2 addroad()

```
void addroad (
    lcities *& pName,
    double dist,
    existindex & cityloc )
```

Funkcja dodaje drogę, tzn dystans między miastami A B. Funkcja wymaga podania struktury existindex by uprościć proces poszukiwania wskaźników na odpowiednie miasta, te zaś są dodawane do struktury existindex przez funkcję existroad.

Ostrzeżenie

Funkcja nie sprawdza czy miasta istniały, ani czy drogi istniały i dla stabilności oraz poprawności działania programu muszą one istnieć.

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>dist</i>	- dystans do dodania
<i>cityloc</i>	- struktura zawierająca wskaźniki na miasta, między którymi występuje dystans

Zwraca

void

4.3.2.3 existroad()

```
existindex existroad (
    lcities *& pName,
    const string & cityA,
    double distance,
    const string & cityB )
```

Funkcja odnajduje miasta i zwraca strukture existindex ze wskaznikami na szukane elementy o ile istniały.

Parametry

<i>pName</i>	- wskaznik na strukture mapy
<i>cityA</i>	- nazwa miasta A
<i>distance</i>	- dystans miedzy miastami
<i>cityB</i>	- nazwa miasta B

Zwraca

existindex - struktura z informacjami o informacjach ktore sa juz w strukturze, i ktorych nie ma

Zobacz również

[existindex](#)

4.3.2.4 loadroad()

```
void loadroad (
    lcities *& pName,
    const string & infilename )
```

Funkcja jest medium do tworzenia struktury mapy. Funkcja pobiera linie z pliku i sprawdzając ich poprawność tworzy strukture mapy.

Parametry

<i>pName</i>	- wskaznik na strukture mapy
<i>infilename</i>	- sciezka do pliku wejsciowego z danymi do mapy

Zwraca

void

Wyjątki

<i>nominusd</i>	-throw, gdy był błąd pliku wejściowego
<i>maplineerr</i>	- błąd w linii w pliku
<i>wasdifferentdist</i>	- był już podany inny dystans

4.4 Dokumentacja pliku Loading.h

Plik zawiera funkcje użytkowe do działań na liście pName, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejściowego.

```
#include "Mapheader.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- [existindex](#) [existroad](#) ([lcities](#) *&pName, const string &cityA, double distance, const string &cityB)
Funkcja odnajduje miasta i zwraca struktury existindex ze wskaźnikami na szukane elementy o ile istniały.
- void [addcity](#) ([lcities](#) *&pName, const string &cityA)
Funkcja dodaje nowe miasto do struktury mapy, tzn lcities.
- void [addroad](#) ([lcities](#) *&pName, double dist, [existindex](#) &cityloc)
Funkcja dodaje drogę, tzn dystans między miastami A B. Funkcja wymaga podania struktury existindex by uprościć proces poszukiwania wskaźników na odpowiednie miasta, te zaś są dodawane do struktury existindex przez funkcję existroad.
- void [loadroad](#) ([lcities](#) *&pName, const string &infilename)
Funkcja jest medium do tworzenia struktury mapy. Funkcja pobiera linie z pliku i sprawdzając ich poprawność tworzy strukturę mapy.

4.4.1 Opis szczegółowy

Plik zawiera funkcje użytkowe do działań na liście pName, z naciskiem na stworzenie struktury mapy na podstawie danych z pliku wejściowego.

Autor

Tomasz Skowron

Zobacz również[Loading.cpp](#)

4.4.2 Dokumentacja funkcji

4.4.2.1 addcity()

```
void addcity (
    lcities *& pName,
    const string & cityA )
```

Funkcja dodaje nowe miasto do struktury mapy, tzn lcities.

Ostrzeżenie

Funkcja nie sprawdza czy miasto istnieje i dla stabilności oraz poprawności działania programu powinno nie istnieć w strukturze.

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>cityA</i>	- nazwa miasta A

Zwraca

void

4.4.2.2 addroad()

```
void addroad (
    lcities *& pName,
    double dist,
    existindex & cityloc )
```

Funkcja dodaje drogę, tzn dystans między miastami A B. Funkcja wymaga podania struktury existindex by uprościć proces poszukiwania wskaźników na odpowiednie miasta, te zaś są dodawane do struktury existindex przez funkcję existroad.

Ostrzeżenie

Funkcja nie sprawdza czy miasta istniały, ani czy drogi istniały i dla stabilności oraz poprawności działania programu muszą one istnieć.

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>dist</i>	- dystans do dodania
<i>cityloc</i>	- struktura zawierająca wskaźniki na miasta, między którymi występuje dystans

Zwraca

void

4.4.2.3 existroad()

```
existindex existroad (
    lcities *& pName,
    const string & cityA,
    double distance,
    const string & cityB )
```

Funkcja odnajduje miasta i zwraca strukture existindex ze wskaznikami na szukane elementy o ile istniały.

Parametry

<i>pName</i>	- wskaznik na strukture mapy
<i>cityA</i>	- nazwa miasta A
<i>distance</i>	- dystans miedzy miastami
<i>cityB</i>	- nazwa miasta B

Zwraca

existindex - struktura z informacjami o informacjach ktore sa juz w strukturze, i ktorych nie ma

Zobacz również

[existindex](#)

4.4.2.4 loadroad()

```
void loadroad (
    lcities *& pName,
    const string & infilename )
```

Funkcja jest medium do tworzenia struktury mapy. Funkcja pobiera linie z pliku i sprawdzając ich poprawność tworzy strukture mapy.

Parametry

<i>pName</i>	- wskaznik na strukture mapy
<i>infilename</i>	- sciezka do pliku wejsciowego z danymi do mapy

Zwraca

void

Wyjątki

<i>nominusd</i>	-throw, gdy był błąd pliku wejściowego
<i>maplineerr</i>	- błąd w linii w pliku
<i>wasdifferentdist</i>	- był już podany inny dystans

4.5 Dokumentacja pliku Mapa.cpp

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
#include "pathfinding.h"
#include "Loading.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- int [main](#) (int paramixmax, char *params[])

Główna funkcja programu.

4.5.1 Opis szczegółowy

Plik z główną funkcją programu [main\(\)](#).

Autor

Tomasz Skowron

4.5.2 Dokumentacja funkcji

4.5.2.1 main()

```
int main (
    int paramixmax,
    char * params[ ] )
```

Główna funkcja programu.

Ostrzeżenie

W programie przyjęto nazwę `mapa` dla struktury z miastami i drogami, nie jest to tablica asocjacyjna. Funkcja wywołuje obsługę parametrów. Funkcja wywołuje w odpowiedni sposób funkcje `core`, zastępując brakujące parametry standardowymi wyjściami. Zostanie również wywołane ładowanie struktury mapy z pliku. funkcja może zgłosić wyjątek, ale zostanie on obsługowany w bloku `catch` na końcu `main()`, funkcja wychwytuje również wszystkie wyjątki spadające do niej z podfunkcji.

Parametry

<code>paramixmax</code>	-zmienna z ilością parametrów
<code>params</code>	-tablica parametrów

Zwraca

0

4.6 Dokumentacja pliku Mapheader.h

```
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Komponenty

- struct `lcities`

Struktura mapy złożonej z dróg jednokierunkowych. `lcities` to lista list. Struktura reprezentuje miasto. Miasta tworzą listę jednokierunkową. Każdemu miastu przypisana jest droga lub lista dróg wychodzących do innych miast.

- struct `lcities::ldist`

Struktura dystansów w mapie.

- struct `existindex`

Struktura dla funkcji `existroad`, z informacjami o (nie)istnieniu miast oraz wskaźnikami na nie. Struktura jest uzupełniana funkcją `existroad`, by stanowić podstawy do uzupełniania listy przy ładowaniu miast lub nie.

- struct `pathtab`

Struktura do podawania wag odległościowych i poprawnego wykonania algorytmu Dijkstry, dalej zwana `pathtab`.

- struct `ldirect`

Struktura drogi z `A` do `B`. Struktura wskazuje na elementy tablicy `pathtab` (`pathtab`), przez które należy iść aby dotrzeć z miasta `A` do miasta `B`, a dokładniej z miasta `B` do miasta `A`, ponieważ funkcje poruszają się z vertex do pvertex poczynając od vertex wskazujący na miasto `B`.

Wyliczenia

- enum `errorid` {
`unknown`, `maperr`, `wasdifferentdist`, `noroute`,
`earlyhit`, `maplineerr`, `nominusd`, `nominust`,
`nominuso`, `outferr`, `routeerr`, `noTnoO`,
`donothing`, `nodist`, `okdist` }

Wyliczenie wyjątków programu. Zawiera wszystkie identyfikatory wszystkich wyjątków, które pojawiają się w programie.

4.6.1 Dokumentacja typów wyliczanych

4.6.1.1 errorid

enum `errorid`

Wyliczenie wyjątków programu. Zawiera wszystkie identyfikatory wszystkich wyjątków, które pojawiają się w programie.

Wartości wyliczeń

<code>unknown</code>	0 nieznany
<code>maperr</code>	1 błąd pliku mapy
<code>wasdifferentdist</code>	2 błąd dystansu do existindex
<code>noroute</code>	3 nie ma drogi
<code>earlyhit</code>	4 znaleziono trasę
<code>maplineerr</code>	5 błąd linii w mapie
<code>nominusd</code>	6 brak pliku mapy
<code>nominust</code>	7 brak pliku tras od wyznaczenia
<code>nominuso</code>	8 brak pliku wyjściowego
<code>outferr</code>	9 błąd pliku wyjściowego
<code>routeerr</code>	10 błąd pliku z trasami do wyznaczenia
<code>noTnoO</code>	11 nie podano ani pliku z trasami, ani wyjściowego
<code>donothing</code>	12 nie rob nic, kończ program
<code>nodist</code>	13 nie ma drogi do existindex
<code>okdist</code>	14 droga jest taka sama jak podana

4.7 Dokumentacja pliku pathfinding.cpp

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje droga z A do B.

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
```



```
#include "pathfinding.h"
#include "Loading.h"
```

Funkcje

- `pathtab * initialize (lcities * &pName, pathtab * &path)`
Funkcja inicjalizuje liste (tablice) pathtab na podstawie wierzchołkow struktury mapy.
- `pathtab * findinpath (pathtab * &path, const string &cityA)`
Funkcja odnajduje i zwraca wskaźnik na element z tablicy pathtab, który wskazuje na miasto o zadanej nazwie.
- `lcities * findiname (lcities * &pName, const string &cityA)`
Funkcja odnajduje i zwraca wskaźnik na element ze struktury mapy, który wskazuje na miasto o zadanej nazwie.
- `pathtab * findmin (pathtab * &path, const string &cityA)`
Funkcja znajduje w tablicy pathtab wskaźnik na minimalna wartosc wierzchołka, co jest jednym z kluczowych krokow algorytmu Dijkstry.
- `bool allvisited (pathtab * &path, const string &cityA)`
Funkcja sprawdza czy jest istniejący element tablicy pathtab, który nie był nigdy odwiedzony.
- `pathtab findshortAB (lcities * &pName, const string &cityA, const string &cityB, pathtab * &path)`
Funkcja wypełnia tablice pathtab wagami odległości punktu B od A.
- `bool existV (ldirect * &pVisited, const string &city)`
Sprawdza czy w strukturze ldirect istnieje element wskazujący na miasto o zadanej nazwie.
- `bool existroadAtoB (const string &cityA, const string &cityB, lcities * &pName)`
*Sprawdza czy w strukturze mapy występuje droga z A do B. Ta funkcja różni się od existroad, gdyż sprawdza wyłącznie istnienie i nie zwraca gdzie wystąpiły.
Stworzone dla funkcji allroadexist().*
- `bool allroadexist (ldirect * &pVisited, lcities * &pName, const string &startcity)`
Sprawdza czy drogi między wierzchołkami (vertex), a poprzednimi wierzchołkami (pvertex) w strukturze ldirect istnieją w strukturze mapy.
- `ldirect * interpret (pathtab * &path, const string &cityA, const string &cityB, lcities * &pName)`
Funkcja wypełnia struktury ldirect trasami a punktu A do B na podstawie wartości vertex i pvertex z tablicy pathtab.
- `ldirect * findinvisited (ldirect * &pVisited, const string &city)`
Odnajduje w strukturze ldirect element o zadanej nazwie.
- `void writeroute (ldirect * &route, ostream &stream)`
Funkcja wypisuje do zadanego strumienia drogę z A do B.
- `void weightless (ldirect * &route, const string &endcity, const string &startcity)`
Funkcja przywróci dystanse z $|X-Y|$ usuwając odległość $|A-X|$, powstałe w wyniku algorytmu Dijkstry.
- `double getweight (pathtab * &path, const string &city)`
Funkcja zbiera wagę nadaną przez findshortAB do sdist w elemencie wskazującym na szukana nazwę miasta.

4.7.1 Opis szczegółowy

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje droga z A do B.

Autor

Tomasz Skowron

Zobacz również

[pathfinding.cpp](#)

4.7.2 Dokumentacja funkcji

4.7.2.1 allroadexist()

```
bool allroadexist (
    ldirect *& pVisited,
    lcities *& pName,
    const string & startcity )
```

Sprawdza czy drogi między wierzchołkami (vertex), a poprzednimi wierzchołkami (pvertex) w strukturze ldirect istnieją w strukturze mapy.

Ostrzeżenie

Wystarczy jedna nie istniejąca droga by zwrócono false.

Parametry

<i>pVisited</i>	- wskaźnik na strukturę trasy
<i>pName</i>	- wskaźnik na strukturę mapy
<i>startcity</i>	- wskaźnik na miasto początkowe, z którego drogę szukamy. Pozwala na wcześniejsze przerwanie zakończenia działania funkcji.

Zwraca

true - gdy istnieją wszystkie drogi
false - gdy nie istnieje co najmniej jedno z miast na trasie

4.7.2.2 allvisited()

```
bool allvisited (
    pathtab *& path,
    const string & cityA )
```

Funkcja sprawdza czy jest istniejący element tablicy pathtab, który nie był nigdy odwiedzony.

Parametry

<i>path</i>	- wskaźnik na tablicę pathtab
<i>cityA</i>	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta początkowego nie wracamy), a swoją drogą w prawie wszystkich przypadkach to miasto jest już odwiedzane.

Zwraca

element - wskaźnik na szukany element mapy

Wyjątki

<i>noroute</i>	- gdy nie odnaleziono elementu
----------------	--------------------------------

4.7.2.3 existroadAtoB()

```
bool existroadAtoB (
    const string & cityA,
    const string & cityB,
    lcities *& pName )
```

Sprawdza czy w strukturze mapy występuje droga z A do B. Ta funkcja różni się od `existroad`, gdyż sprawdza wyłącznie istnienie i nie zwraca gdzie wystąpiły. Stworzone dla funkcji `allroadexist()`.

Parametry

<i>cityA</i>	- nazwa szukanego miasta A
<i>cityB</i>	- nazwa szukanego miasta B
<i>pName</i>	- wskaźnik na strukturę mapy

Zwraca

`true` - gdy istnieją oba
`false` - gdy nie istnieje co najmniej jedno z miast

4.7.2.4 existV()

```
bool existV (
    ldirect *& pVisited,
    const string & city )
```

Sprawdza czy w strukturze `ldirect` istnieje element wskazujący na miasto o zadanej nazwie.

Parametry

<i>pVisited</i>	- wskaźnik na strukturę <code>ldirect</code>
<i>city</i>	- nazwa szukanego miasta

Zwraca

true - gdy istnieje
false - gdy nie istnieje

Wyjątki

<i>earlyhit</i>	- gdy droga może być znaleziona wcześniej
-----------------	---

4.7.2.5 findiname()

```
lcities* findiname (
    lcities *& pName,
    const string & cityA )
```

Funkcja odnajduje i zwraca wskaźnik na element ze struktury mapy, który wskazuje na miasto o zadanej nazwie.

Ostrzeżenie

Funkcja może zgłosić wyjątek `noroute`, gdyż jeśli szukamy jakiegos drogi, a miasto nie istnieje to nie można kontynuować poszukiwań. `findinpath` jest zatem niemal bezużyteczna poza [findshortAB\(\)](#).

Parametry

<i>pName</i>	- wskaźnik na tablicę <code>pathtab</code>
<i>cityA</i>	- nazwa szukanego miasta

Zwraca

element - wskaźnik na szukany element mapy

Wyjątki

<i>noroute</i>	- gdy nie odnaleziono elementu
----------------	--------------------------------

4.7.2.6 findinpath()

```
pathtab* findinpath (
    pathtab *& path,
    const string & cityA )
```

Funkcja odnajduje i zwraca wskaźnik na element z tablicy `pathtab`, który wskazuje na miasto o zadanej nazwie.

Ostrzeżenie

Nie wolno szukać elementów które nie istnieją, szczególnie jeśli wywołujemy to z funkcji [findshortAB\(\)](#).

Parametry

<i>path</i>	- wskaźnik na tablice pathtab
<i>cityA</i>	- nazwa szukanego miasta

Zwraca

element - wskaźnik na szukany element tablicy pathtab
nullptr - gdy nie odnaleziono elementu

4.7.2.7 findinvisited()

```
ldirect* findinvisited (
    ldirect *& pVisited,
    const string & city )
```

Odnajduje w strukturze ldirect element o zadanej nazwie.

Ostrzeżenie

Funkcja nie wypisuje dystansu z A do B, tylko poszczególne dystanse z tablicy pathtab. Poleca się użycie weightless na tablicy pathtab by otrzymać poprawne dane, z $|X-Y|$, a nie $|A-X|+|X-Y|$.

Zobacz również

weightless
[findshortAB](#)

Parametry

<i>pVisited</i>	- wskaźnik na strukturę trasy z A do B
<i>city</i>	- nazwa szukanego miasta

Zwraca

pHeadV - wskaźnik na szukany element
nullptr - gdy miasto nie istnieje w strukturze

4.7.2.8 findmin()

```
pathtab* findmin (
    pathtab *& path,
    const string & cityA )
```

Funkcja znajduje w tablicy pathtab wskaźnik na minimalną wartość wierzchołka, co jest jednym z kluczowych kroków algorytmu Dijkstry.

Ostrzeżenie

Minimum może okazać się nieskończonością, czyli 0. Funkcja pomija miasto A oraz miasta odwiedzone, więc jeśli wszystkie były odwiedzone to zwróci wartość początku tablicy `pathtab`.
Findmin porównuje wagi (`value`, czyli `sdist`), a nie dystanse ze struktury mapy!

Parametry

<i>path</i>	- wskaźnik na <code>pathtab</code>
<i>cityA</i>	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta początkowego nie wracamy)

Zwraca

min - wskaźnik na minimalny pod względem `sdist` i zgodny z algorytmem element tablicy `pathtab`

Wyjątki

<i>noroute</i>	gdy miasto nie istnieje
----------------	-------------------------

4.7.2.9 findshortAB()

```
pathtab findshortAB (
    lcities *& pName,
    const string & cityA,
    const string cityB,
    pathtab *& path )
```

Funkcja wypełnia tablicę `pathtab` wagami odległości punktu B od A.

Ostrzeżenie

Wagi nie są równie odległościom $|X-Y|$, tylko $|A-X|+|X-Y|$. Aby mieć w tablicy odległości między poszczególnymi wierzchołkami należy ją podać funkcji `weightless()`

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>cityA</i>	- nazwa miasta A
<i>cityB</i>	- nazwa miasta B
<i>path</i>	- wskaźnik na tablicę <code>pathtab</code>

Zwraca

`path` - uzupełniona wagami tablica `pathtab`

Wyjątki

<i>earlyhit</i>	- jeśli droga została znaleziona wcześniej, niż odwiedzenie wszystkich miast przez Dijkstre
<i>noroute</i>	- jeśli nie istnieje żadna droga z miasta A

Ostrzeżenie

Mozna sobie pozwolić na *earlyhit*, nawet w strukturze mapy rozdzielnej, gdzie nie każda droga można osiągnąć, gdyż przejście do nieosiągalnej części mapy jest ostatecznie ograniczana przez *findmin*, a jeśli tak się stało to *alloradexist* ponownie zabezpiecza niepoprawny wynik.

4.7.2.10 `getweight()`

```
double getweight (
    pathtab *& path,
    const string & city )
```

Funkcja zbiera wagę nadaną przez *findshortAB* do *sdist* w elemencie wskazującym na szukana nazwę miasta.

Ostrzeżenie

Szczególnie dobrym użyciem tej funkcji jest zebranie odległości z A do B przez użyciem *weightless* na tablicy *pathtab*, daje to gotową sumę dystansów.
Droga z A do B musi istnieć.

Parametry

<i>path</i>	- wskaźnik na tablicę <i>pathtab</i>
<i>city</i>	- nazwa miasta o szukanej wadze

Zwraca

value - waga odległości
0 - gdy waga jest nie możliwa do odnalezienia

4.7.2.11 `initialize()`

```
pathtab* initialize (
    lcities *& pName,
    pathtab *& path )
```

Funkcja inicjalizuje listę (tablicę) *pathtab* na podstawie wierzchołków struktury mapy.

Ostrzeżenie

Wartości domyślne elementów listy to: `new pathtab{ pHeadN,0,nullptr,false,nullptr }`

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>path</i>	- wskaźnik na pathtab do utworzenia

Zwraca

path - wskaźnik na zainicjalizowaną tablicę
nullptr - jeśli pName było puste

4.7.2.12 interpret()

```
ldirect* interpret (
    pathtab *& path,
    const string & cityA,
    const string & cityB,
    lcities *& pName )
```

Funkcja wypełnia strukturę ldirect trasami a punktu A do B na podstawie wartości vertex i pvertex z tablicy pathtab.

Parametry

<i>path</i>	- wskaźnik na tablicę pathtab
<i>cityA</i>	- nazwa miasta A
<i>cityB</i>	- nazwa miasta B
<i>pName</i>	- wskaźnik na strukturę mapy

Zwraca

pVisited - struktura zawierająca interesującą nas drogę z A do B
nullptr - w wypadku niepowodzenia/przerwania petli

Wyjątki

<i>noroute</i>	- gdy funkcja otrzyma sygnał, że droga nie istnieje
----------------	---

4.7.2.13 weightless()

```
void weightless (
    ldirect *& route,
    const string & endcity,
    const string & startcity )
```

Funkcja przywróci dystanse z $|X-Y|$ usuwając odległość $|A-X|$, powstałe w wyniku algorytmu Dijkstry.

Parametry

<i>route</i>	- wskaźnik na strukturę trasy z A do B
<i>endcity</i>	- nazwa miasta końcowego
<i>startcity</i>	- wskaźnik na miasto początkowe

Zwraca

void

4.7.2.14 writerroute()

```
void writerroute (
    ldirect *& route,
    ostream & stream )
```

Funkcja wypisuje do zadanego strumienia drogę z A do B.

Parametry

<i>route</i>	- wskaźnik na strukturę trasy
<i>stream</i>	- strumień wyjściowy do zapisu

Zwraca

void

4.8 Dokumentacja pliku pathfinding.h

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje droga z A do B.

```
#include "Mapheader.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- `pathtab * initialize (lcities *&pName, pathtab *&path)`
Funkcja inicjalizuje listę (tablicę) pathtab na podstawie wierzchołków struktury mapy.
- `pathtab * findinpath (pathtab *&path, const string &cityA)`
Funkcja odnajduje i zwraca wskaźnik na element z tablicy pathtab, który wskazuje na miasto o zadanej nazwie.
- `lcities * findiname (lcities *&pName, const string &cityA)`

- Funkcja odnajduje i zwraca wskaźnik na element ze struktury mapy, który wskazuje na miasto o zadanej nazwie.*

• `path` * `findmin` (`path` *`&path`, `const string &cityA`)

Funkcja znajduje w tablicy `path` wskaźnik na minimalna wartosc wierzolka, co jest jednym z kloczowych krokow algorytmu Dijkstry.
- `bool` `allvisited` (`path` *`&path`, `const string &cityA`)

Funkcja sprawdza czy jest istniejący element tablicy `path`, który nie był nigdy odwiedzony.
- `path` * `findshortAB` (`lcity` *`&pName`, `const string &cityA`, `const string cityB`, `path` *`&path`)

Funkcja wypelnia tablice `path` wagami odleglosci punktu B od A.
- `bool` `existV` (`ldirect` *`&pVisited`, `const string &city`)

Sprawdza czy w strukturze `ldirect` istnieje element wskazujący na miasto o zadanej nazwie.
- `bool` `existroadAtoB` (`const string &cityA`, `const string &cityB`, `lcity` *`&pName`)

Sprawdza czy w strukturze mapy występuje droga z A do B. Ta funkcja różni się od `existroad`, gdyż sprawdza wyłącznie istnienie i nie zwraca gdzie wystąpiły.

Stworzone dla funkcji `allroadexist()`.
- `bool` `allroadexist` (`ldirect` *`&pVisited`, `lcity` *`&pName`, `const string &startcity`)

Sprawdza czy drogi między wierzchołkami (vertex), a poprzednimi wierzchołkami (pvertex) w strukturze `ldirect` istnieją w strukturze mapy.
- `ldirect` * `interpret` (`path` *`&path`, `const string &cityA`, `const string &cityB`, `lcity` *`&pName`)

Funkcja wypelnia strukture `ldirect` trasami a punktu A do B na podstawie wartosci vertex i pvertex z tablicy `path`.
- `ldirect` * `findinvisited` (`ldirect` *`&pVisited`, `const string &city`)

Odnajduje w strukturze `ldirect` element o zadanej nazwie.
- `void` `writeroute` (`ldirect` *`&route`, `ostream &stream`)

Funkcja wypisuje do zadanego strumienia droge z A do B.
- `void` `weightless` (`ldirect` *`&route`, `const string &endcity`, `const string &startcity`)

Funkcja przywróci dystanse z |X-Y| usuwając odleglosc |A-X|, powstałe w wyniku algorytmu Dijkstry.
- `double` `getweight` (`path` *`&path`, `const string &city`)

Funkcja zbiera wage nadana przez `findshortAB` do `sdist` w elemencie wskazującym na szukana nazwe miasta.

4.8.1 Opis szczegółowy

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje droga z A do B.

Autor

Tomasz Skowron

Zobacz również

[pathfinding.cpp](#)

4.8.2 Dokumentacja funkcji

4.8.2.1 `allroadexist()`

```
bool allroadexist (
    ldirect *& pVisited,
    lcity *& pName,
    const string & startcity )
```

Sprawdza czy drogi między wierzchołkami (vertex), a poprzednimi wierzchołkami (pvertex) w strukturze `ldirect` istnieją w strukturze mapy.

Ostrzeżenie

Wystarczy jedna nie istniejąca droga by zwrócono false.

Parametry

<i>pVisited</i>	- wskaźnik na strukturę trasy
<i>pName</i>	- wskaźnik na strukturę mapy
<i>startcity</i>	- wskaźnik na miasto początkowe, z którego drogę szukamy. Pozwala na wcześniejsze przerwanie zakończenia działania funkcji.

Zwraca

true - gdy istnieją wszystkie drogi
false - gdy nie istnieje co najmniej jedno z miast na trasie

4.8.2.2 allvisited()

```
bool allvisited (
    pathtab *& path,
    const string & cityA )
```

Funkcja sprawdza czy jest istniejący element tablicy pathtab, który nie był nigdy odwiedzony.

Parametry

<i>path</i>	- wskaźnik na tablicę pathtab
<i>cityA</i>	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta początkowego nie wracamy), a swoją drogą w prawie wszystkich przypadkach to miasto jest już odwiedzane.

Zwraca

element - wskaźnik na szukany element mapy

Wyjątki

<i>noroute</i>	- gdy nie odnaleziono elementu
----------------	--------------------------------

4.8.2.3 existroadAtoB()

```
bool existroadAtoB (
    const string & cityA,
    const string & cityB,
    lcities *& pName )
```

Sprawdza czy w strukturze mapy występuje droga z A do B. Ta funkcja różni się od existroad, gdyż sprawdza wyłącznie istnienie i nie zwraca gdzie wystąpiły. Stworzone dla funkcji [allroadexist\(\)](#).

Parametry

<i>cityA</i>	- nazwa szukanego miasta A
<i>cityB</i>	- nazwa szukanego miasta B
<i>pName</i>	- wskaźnik na strukturę mapy

Zwraca

true - gdy istnieją oba
false - gdy nie istnieje co najmniej jedno z miast

4.8.2.4 existV()

```
bool existV (
    ldirect *& pVisited,
    const string & city )
```

Sprawdza czy w strukturze ldirect istnieje element wskazujący na miasto o zadanej nazwie.

Parametry

<i>pVisited</i>	- wskaźnik na strukturę ldirect
<i>city</i>	- nazwa szukanego miasta

Zwraca

true - gdy istnieje
false - gdy nie istnieje

Wyjątki

<i>earlyhit</i>	- gdy droga może być znaleziona wcześniej
-----------------	---

4.8.2.5 findiname()

```
lcities* findiname (
    lcities *& pName,
    const string & cityA )
```

Funkcja odnajduje i zwraca wskaźnik na element ze struktury mapy, który wskazuje na miasto o zadanej nazwie.

Ostrzeżenie

Funkcja może zgłosić wyjątek noroute, gdyż jeśli szukamy jakiejś drogi, a miasto nie istnieje to nie można kontynuować poszukiwań. findinpath jest zatem niemal bezużyteczna poza [findshortAB\(\)](#).

Parametry

<i>pName</i>	- wskaźnik na tablice pathtab
<i>cityA</i>	- nazwa szukanego miasta

Zwraca

element - wskaźnik na szukany element mapy

Wyjątki

<i>noroute</i>	- gdy nie odnaleziono elementu
----------------	--------------------------------

4.8.2.6 findinpath()

```
pathtab* findinpath (
    pathtab *& path,
    const string & cityA )
```

Funkcja odnajduje i zwraca wskaźnik na element z tablicy pathtab, który wskazuje na miasto o zadanej nazwie.

Ostrzeżenie

Nie wolno szukać elementów które nie istnieją, szczególnie jeśli wywołujemy to z funkcji [findshortAB\(\)](#).

Parametry

<i>path</i>	- wskaźnik na tablice pathtab
<i>cityA</i>	- nazwa szukanego miasta

Zwraca

element - wskaźnik na szukany element tablicy pathtab
nullptr - gdy nie odnaleziono elementu

4.8.2.7 findinvisited()

```
ldirect* findinvisited (
    ldirect *& pVisited,
    const string & city )
```

Odnajduje w strukturze ldirect element o zadanej nazwie.

Ostrzeżenie

Funkcja nie wypisuje dystansu z A do B, tylko poszczególne dystanse z tablicy pathtab. Poleca się użycie `weightless` na tablicy pathtab by otrzymać poprawne dane, z $|X-Y|$, a nie $|A-X|+|X-Y|$.

Zobacz również

`weightless`
[findshortAB](#)

Parametry

<i>pVisited</i>	- wskaźnik na strukturę trasy z A do B
<i>city</i>	- nazwa szukanego miasta

Zwraca

`pHeadV` - wskaźnik na szukany element
`nullptr` - gdy miasto nie istnieje w strukturze

4.8.2.8 findmin()

```
pathtab* findmin (
    pathtab *& path,
    const string & cityA )
```

Funkcja znajduje w tablicy pathtab wskaźnik na minimalną wartość wierzchołka, co jest jednym z kluczowych kroków algorytmu Dijkstry.

Ostrzeżenie

Minimum może okazać się nieskończonością, czyli 0. Funkcja pomija miasto A oraz miasta odwiedzone, więc jeśli wszystkie były odwiedzone to zwróci wartość początku tablicy pathtab. Findmin porównuje wagi (`value`, czyli `sdist`), a nie dystanse ze struktury mapy!

Parametry

<i>path</i>	- wskaźnik na pathtab
<i>cityA</i>	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta początkowego nie wracamy)

Zwraca

`min` - wskaźnik na minimalny pod względem `sdist` i zgodny z algorytmem element tablicy pathtab

Wyjątki

<i>noroute</i>	gdy miasto nie istnieje
----------------	-------------------------

4.8.2.9 findshortAB()

```
pathtab findshortAB (
    lcities *& pName,
    const string & cityA,
    const string cityB,
    pathtab *& path )
```

Funkcja wypelnia tablice pathtab wagami odleglosci punktu B od A.

Ostrzeżenie

Wagi nie sa rownie odleglosciom $|X-Y|$, tylko $|A-X|+|X-Y|$. Aby miec w tablicy odleglosci miedzy poszczegolnymi wierzchołkami nalezy ja podac funkcji [weightless\(\)](#)

Parametry

<i>pName</i>	- wskaznik na strukture mapy
<i>cityA</i>	- nazwa miasta A
<i>cityB</i>	- nazwa miasta B
<i>path</i>	- wkaznik na tablice pathtab

Zwraca

path - uzupełniona wagami tablica pathtab

Wyjątki

<i>earlyhit</i>	- jesli droga zostala znaleziona wczesniej, niz odwiedzenie wszystkich miast przez Dijkstre
<i>noroute</i>	- jesli nie istnieje zadna droga z miata A

Ostrzeżenie

Mozna sobie pozwolic na earlyhit, nawet w strukturze mapy rozdzielnej, gdzie nie kazda droge mozna osiagnac, gdyz przejscie do nieosiagalnej czesci mapy jest ostatecznoscia ograniczana przez findmin, a jesli tak sie stalo to alloradexist ponownie zabezpiecza niepoprawny wynik.

4.8.2.10 getweight()

```
double getweight (
    pathtab *& path,
    const string & city )
```

Funkcja zbiera wage nadana przez findshortAB do sdist w elemencie wskazujacym na szukana nazwe miasta.

Ostrzeżenie

Szczególnie dobrym użyciem tej funkcji jest zebranie odległości z A do B przez użyciem `weightless` na tablicy `pathtab`, daje to gotową sumę dystansów.
Droga z A do B musi istnieć.

Parametry

<i>path</i>	- wskaźnik na tablicę <code>pathtab</code>
<i>city</i>	- nazwa miasta o szukanej wadze

Zwraca

`value` - waga odległości
0 - gdy waga jest nie możliwa do odnalezienia

4.8.2.11 initialize()

```
pathtab* initialize (
    lcities *& pName,
    pathtab *& path )
```

Funkcja inicjalizuje listę (tablicę) `pathtab` na podstawie wierzchołków struktury mapy.

Ostrzeżenie

Wartości domyślne elementów listy to: `new pathtab{ pHeadN,0,nullptr,false,nullptr }`

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
<i>path</i>	- wskaźnik na <code>pathtab</code> do utworzenia

Zwraca

`path` - wskaźnik na zainicjalizowaną tablicę
`nullptr` - jeśli `pName` było puste

4.8.2.12 interpret()

```
ldirect* interpret (
    pathtab *& path,
    const string & cityA,
    const string & cityB,
    lcities *& pName )
```

Funkcja wypełnia strukturę `ldirect` trasami z punktu A do B na podstawie wartości `vertex` i `pvertex` z tablicy `pathtab`.

Parametry

<i>path</i>	- wskaźnik na tablice pathtab
<i>cityA</i>	- nazwa miasta A
<i>cityB</i>	- nazwa miasta B
<i>pName</i>	- wskaźnik na strukturę mapy

Zwraca

pVisited - struktura zawierająca interesującą nas drogę z A do B
nullptr - w przypadku niepowodzenia/przerwania petli

Wyjątki

<i>noroute</i>	- gdy funkcja otrzyma sygnał, że droga nie istnieje
----------------	---

4.8.2.13 weightless()

```
void weightless (
    ldirect *& route,
    const string & endcity,
    const string & startcity )
```

Funkcja przywróci dystanse z $|X-Y|$ usuwając odległość $|A-X|$, powstałe w wyniku algorytmu Dijkstry.

Parametry

<i>route</i>	- wskaźnik na strukturę trasy z A do B
<i>endcity</i>	- nazwa miasta końcowego
<i>startcity</i>	- wskaźnik na miasto początkowe

Zwraca

void

4.8.2.14 writeroute()

```
void writeroute (
    ldirect *& route,
    ostream & stream )
```

Funkcja wypisuje do zadanego strumienia drogę z A do B.

Parametry

<i>route</i>	- wskaźnik na strukturę trasy
<i>stream</i>	- strumień wyjściowy do zapisu

Zwraca

void

4.9 Dokumentacja pliku Service.cpp

Zawiera funkcje dla `main()`, do obsługi wprowadzanych danych, czy to w postaci argumentów linii poleceń, czy danych bezpośrednio z pliku.

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
#include "pathfinding.h"
#include "Loading.h"
```

Funkcje

- void `printhelp` ()
Funkcja wypisuje na ekranie okno pomocy.
- bool `correctparams` (const string &ifpath, const string &ofpath, const string &rfilepath)
Funkcja sprawdza poprawność parametrów.
- void `doparameters` (int paramixmax, char *params[], string &mfilepath, string &ofpath, string &rfilepath, bool &wasrf, bool &wasof)
Funkcja obsługuje podane argumenty z linii poleceń, ustawia podane parametry na stan adekwatny do sytuacji.
- void `noroad` (ostream &outfile, const string &cityA, const string cityB)
Funkcja zaisuje do podanego strumienia wyjściowego informacje o nieistniejącej drodze z A do B.
- void `core` (lcities *&pName, pathtab *&path, ldirect *&route, ostream &outfile, istream &rfile)
Funkcja obsługuje podane argumenty z linii poleceń, ustawia podane parametry na stan adekwatny do sytuacji.
- string `allexcep` (errorid excep)
Funkcja obsługuje wszystkie wyjątki które mogą wystąpić w programie.
- string `mainexception` (errorid excep, bool &wasrf, bool &wasof)
Funkcja obsługuje tylko część wyjątków specjalnie dla funkcji main.

4.9.1 Opis szczegółowy

Zawiera funkcje dla `main()`, do obsługi wprowadzanych danych, czy to w postaci argumentów linii poleceń, czy danych bezpośrednio z pliku.

Autor

Tomasz Skowron

Zobacz również

[Service.h](#)

4.9.2 Dokumentacja funkcji

4.9.2.1 allexcep()

```
string allexcep (
    errorid excep )
```

Funkcja obsługuje wszystkie wyjątki które mogą wystąpić w programie.

Zobacz również

[errorid](#)

Parametry

<i>excep</i>	- wyjątek (na podstawie wyliczenia errorid)
--------------	---

Zwraca

- "nieznany błąd \n"
- "błąd pliku wejściowego z mapą \n"
- "inny dystans między miastami został już podany \n"
- "droga nie istnieje \n"
- "znaleziono trasę"
- "błąd w linii w pliku z połączeniami między miastami \n"
- "nie podano pliku z drogami \n"
- "nie podano pliku z trasami do wyznaczenia \n"
- "nie podano pliku wyjściowego \n"
- "błąd pliku wyjściowego \n"
- "błąd pliku z drogami do wyznaczenia \n"
- "nie podano pliku z trasami, ani wyjściowego \n"
- ""

4.9.2.2 core()

```
void core (
    lcities *& pName,
    pathTab *& path,
    ldirect *& route,
    ostream & outfile,
    istream & rfile )
```

Funkcja obsługuje podane argumenty z linii poleceń, ustawia podane parametry na stan adekwatny do sytuacji.

Parametry

<i>pName</i>	- wskaźnik na strukturę mapy
--------------	------------------------------

Parametry

<i>path</i>	- wskaźnik na strukturę pathtab
<i>route</i>	- wskaźnik na strukturę ldirect, droge z A do B
<i>outfile</i>	- strumień wyjściowy
<i>rfile</i>	- strumień wejściowy z którego czytane są poszukiwane trasy

Zwraca

void

4.9.2.3 correctparams()

```
bool correctparams (
    const string & ifpath,
    const string & ofpath,
    const string & rfilepath )
```

Funkcja sprawdza poprawność parametrów.

Parametry

<i>ifpath</i>	- ścieżka do pliku wejściowego
<i>ofpath</i>	- ścieżka do pliku wyjściowego
<i>rfilepath</i>	- ścieżka do pliku z trasami

Zwraca

true - zawsze true, o ile wcześniej nie było błędu krytycznego (brak pliku mapy, niepoprawnie otwarty/nie otwarty plik)

Wyjątki

<i>nominusd</i>	gdy nie ma pliku z mapą
<i>noTnoO</i>	gdy nie ma ani pliku z trasami do wyznaczenia, ani pliku wyjściowego
<i>nominuso</i>	gdy nie ma pliku wyjściowego
<i>nominust</i>	gdy nie ma pliku z trasami do wyznaczenia

4.9.2.4 doparameters()

```
void doparameters (
    int paramixmax,
    char * params[],
```

```

    string & mfp1ath,
    string & of2path,
    string & rf3path,
    bool & was4rf,
    bool & was5of )

```

Funkcja obsługuje podane argumenty z linii polece¹n, ustawia podane parametry na stan adekwatny do sytuacji.

Parametry

	<i>paramixmax</i>	-zmienna z iloscia parametrow
	<i>params</i>	- tablica parametrow
out	<i>mfp¹ath</i>	- sciezka do pliku z mapa
out	<i>of²path</i>	- sciezka do pliku wyjsciowego
	<i>rf³path</i>	- sciezka do pliku z trasami do wyznaczenia
	<i>was⁴rf</i>	- (nie)byl plik z trasami
	<i>was⁵of</i>	- (nie)byl plik wyjsciowy

Zwraca

void

4.9.2.5 mainexception()

```

string mainexception (
    errorid excep,
    bool & was4rf,
    bool & was5of )

```

Funkcja obsługuje tylko czesc wyjat¹kow specjalnie dla funkcji main.

Funkcja moze zmienic was⁴rf lub/i was⁵of na podstawie braku podania ktoregos z parametrow do main.

Obslugiwane wyjatki: noTnoO, nominust, nominuso

Parametry

<i>excep</i>	- wyjatek (na podstawie wyliczenia errorid)
<i>was⁴rf</i>	- czy byl plik z trasami
<i>was⁵of</i>	- czy byl plik wyjsciowy

Zwraca

- "nie podano pliku z trasami ani wyjsciowego"
- "nie podano pliku z trasami do wyznaczenia"
- "nie podano pliku wyjsciowego "

Zobacz również

[main](#)

4.9.2.6 noroad()

```
void noroad (
    ostream & outfile,
    const string & cityA,
    const string cityB )
```

Funkcja zapisuje do podanego strumienia wyjściowego informacje o nieistniejącej drodze z A do B.

Parametry

<i>outfile</i>	- strumień wyjściowy
<i>cityA</i>	- miasto początkowe a miasta A
<i>cityB</i>	- nazwa miasta B

Zwraca

void

4.9.2.7 printhelp()

```
void printhelp ( )
```

Funkcja wypisuje na ekranie okno pomocy.

Zwraca

void

4.10 Dokumentacja pliku Service.h

Zawiera funkcje dla [main\(\)](#), do obsługi wprowadzanych danych, czy to w postaci argumentów linii poleceń, czy danych bezpośrednio z pliku.

```
#include "Mapheader.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- void [printhelp](#) ()
Funkcja wypisuje na ekranie okno pomocy.
- bool [correctparams](#) (const string &ifpath, const string &ofpath, const string &rftpath)
Funkcja sprawdza poprawnosc parametrow.
- void [doparameters](#) (int paramixmax, char *params[], string &mftpath, string &ofpath, string &rftpath, bool &wasrf, bool &wasof)
Funkcja obsluguje podane argumenty z linii polecen, ustawia podane parametry na stan adekwatny do sytuacji.
- void [noroad](#) (ostream &outfile, const string &cityA, const string cityB)
Funkcja zaisuje do podanego strumienia wyjsciowego informacje o nieistniejacej drodze z A do B.
- void [core](#) (lcities *pName, [pathtab](#) *&path, [ldirect](#) *&route, ostream &outfile, istream &rfile)
Funkcja obsluguje podane argumenty z linii polecen, ustawia podane parametry na stan adekwatny do sytuacji.
- string [allexcep](#) ([errorid](#) excep)
Funkcja obsluguje wszystkie wyjatki ktore moga wystapic w programie.
- string [mainexception](#) ([errorid](#) excep, bool &wasrf, bool &wasof)
Funkcja obsluguje tylko czesc wyjatkow specjalnie dla funkcji main.

4.10.1 Opis szczegółowy

Zawiera funkcje dla [main\(\)](#), do obslugi wprowadzanych danych, czy to w postaci argumentow linii polecen, czy danych bezposrednio z pliku.

Autor

Tomasz Skowron

Zobacz również

[Service.cpp](#)

4.10.2 Dokumentacja funkcji

4.10.2.1 allexcep()

```
string allexcep (  
    errorid excep )
```

Funkcja obsluguje wszystkie wyjatki ktore moga wystapic w programie.

Zobacz również

[errorid](#)

Parametry

<code>except</code>	- wyjątek (na podstawie wyliczenia errorid)
---------------------	---

Zwraca

- "nieznany blad \n"
- "blad pliku wejscowego z mapa \n"
- "inny dystans miedzy miastami zostal juz podany \n"
- "droga nie istnieje \n"
- "znaleziono trase"
- "blad w linii w pliku z polaczeniami miedzy miastami \n"
- "nie podano pliku z drogami \n"
- "nie podano pliku z trasami do wyznaczenia \n"
- "nie podano pliku wyjscowego \n"
- "blad pliku wyjscowego \n"
- "blad pliku z drogami do wyznaczenia \n"
- "nie podano pliku z trasami, ani wyjscowego \n"
- ""

4.10.2.2 `core()`

```
void core (
    lcities *& pName,
    pathtab *& path,
    ldirect *& route,
    ostream & outfile,
    istream & rfile )
```

Funkcja obsluhuje podane argumenty z linii polecen, ustawia podane parametry na stan adekwatny do sytuacji.

Parametry

<code>pName</code>	- wskaznik na strukture mapy
<code>path</code>	- wskaznik na strukture pathtab
<code>route</code>	- wskaznik na strukture ldirect, droge z A do B
<code>outfile</code>	- strumien wyjscowy
<code>rfile</code>	- strumien wejscowy z ktorego czytane sa poszukiwane trasy

Zwraca

`void`

4.10.2.3 `correctparams()`

```
bool correctparams (
    const string & ifpath,
```



```
const string & ofpath,
const string & rfp_path )
```

Funkcja sprawdza poprawność parametrów.

Parametry

<i>ifpath</i>	- ścieżka do pliku wejściowego
<i>ofpath</i>	- ścieżka do pliku wyjściowego
<i>rfpath</i>	- ścieżka do pliku z trasami

Zwraca

true - zawsze true, o ile wcześniej nie było błędu krytycznego (brak pliku mapy, niepoprawnie otwarty/nie otwarty plik)

Wyjątki

<i>nominusd</i>	gdy nie ma pliku z mapą
<i>noTnoO</i>	gdy nie ma ani pliku z trasami do wyznaczenia, ani pliku wyjściowego
<i>nominuso</i>	gdy nie ma pliku wyjściowego
<i>nominust</i>	gdy nie ma pliku z trasami do wyznaczenia

4.10.2.4 doparameters()

```
void doparameters (
    int paramixmax,
    char * params[],
    string & mfp_path,
    string & ofpath,
    string & rfp_path,
    bool & wasrf,
    bool & wasof )
```

Funkcja obsługuje podane argumenty z linii poleceń, ustawia podane parametry na stan adekwatny do sytuacji.

Parametry

	<i>paramixmax</i>	- zmienna z ilością parametrów
	<i>params</i>	- tablica parametrów
out	<i>mfp_path</i>	- ścieżka do pliku z mapą
out	<i>ofpath</i>	- ścieżka do pliku wyjściowego
	<i>rfpath</i>	- ścieżka do pliku z trasami do wyznaczenia
	<i>wasrf</i>	- (nie)był plik z trasami
	<i>wasof</i>	- (nie)był plik wyjściowy

Zwraca

void

4.10.2.5 mainexception()

```
string mainexception (
    errorid excep,
    bool & wasrf,
    bool & wasof )
```

Funkcja obsługuje tylko czesc wyjatkov specjalnie dla funkcji main.

Funkcja moze zmienic wasrf lub/i wasof na podstawie braku podania ktoregos z parametrow do main.

Obslugiwane wyjatki: noTnoO, nominust, nominuso

Parametry

<i>excep</i>	- wyjatek (na podstawie wyliczenia errorid)
<i>wasrf</i>	- czy byl plik z trasami
<i>wasof</i>	- czy byl plik wyjsciowy

Zwraca

- "nie podano pliku z trasami ani wyjsciowego"
- "nie podano pliku z trasami do wyznaczenia"
- "nie podano pliku wyjsciowego "

Zobacz również

[main](#)

4.10.2.6 noroad()

```
void noroad (
    ostream & outfile,
    const string & cityA,
    const string cityB )
```

Funkcja zaisuje do podanego strumienia wyjsciowego informacje o nieistniejacej drodze z A do B.

Parametry

<i>outfile</i>	- strumien wyjsciowy
<i>cityA</i>	- miasto poczatkowe a miasta A
<i>cityB</i>	- nazwa miasta B

Zwraca

void

4.10.2.7 printhelp()

```
void printhelp ( )
```

Funkcja wypisuje na ekranie okno pomocy.

Zwraca

void

4.11 Dokumentacja pliku Valid.cpp

Zawiera funkcje to tworzenia stringow tylko ze znakami dozwolonymi.

```
#include "Mapheader.h"
#include "Valid.h"
#include "Service.h"
#include "Cleanse.h"
#include "pathfinding.h"
#include "Loading.h"
```

Funkcje

- string [charonly](#) (const string input)
funkcja tworzy nowy string na podstawie podanego, ale tylko z dopuszczalnych znakow, tzn [0-9a-zA-Z]

4.11.1 Opis szczegółowy

Zawiera funkcje to tworzenia stringow tylko ze znakami dozwolonymi.

Autor

Tomasz Skowron

Zobacz również

[Valid.h](#)

4.11.2 Dokumentacja funkcji

4.11.2.1 charonly()

```
string charonly (
    const string input )
```

funkcja tworzy nowy string na podstawie podanego, ale tylko z dopuszczalnych znakow, tzn [0-9a-zA-Z]

Parametry

<i>input</i>	- lancuch do przetworzeina
--------------	----------------------------

Zwraca

tempinputinternal - nowo utworzony string

4.12 Dokumentacja pliku Valid.h

Zawiera funkcje to tworzenia stringow tylko ze znakami dozwolonymi.

```
#include "Mapheader.h"
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
```

Funkcje

- string [charonly](#) (const string input)
funkcja tworzy nowy string na podstawie podanego, ale tylko z dopuszczalnych znakow, tzn [0-9a-zA-Z]

4.12.1 Opis szczegółowy

Zawiera funkcje to tworzenia stringow tylko ze znakami dozwolonymi.

Autor

Tomasz Skowron

Zobacz również

[Valid.cpp](#)

4.12.2 Dokumentacja funkcji

4.12.2.1 charonly()

```
string charonly (
    const string input )
```

funkcja tworzy nowy string na podstawie podanego, ale tylko z dopuszczalnych znakow, tzn [0-9a-zA-Z]

Parametry

<i>input</i>	- lancuch do przetworzeina
--------------	----------------------------

Zwraca

tempinputinternal - nowo utworzony string

Skorowidz

- addcity
 - Loading.cpp, [17](#)
 - Loading.h, [20](#)
- addroad
 - Loading.cpp, [17](#)
 - Loading.h, [20](#)
- allexcep
 - Service.cpp, [43](#)
 - Service.h, [47](#)
- allroadexist
 - pathfinding.cpp, [26](#)
 - pathfinding.h, [34](#)
- allvisited
 - pathfinding.cpp, [26](#)
 - pathfinding.h, [35](#)
- charonly
 - Valid.cpp, [51](#)
 - Valid.h, [52](#)
- city
 - lcities, [7](#)
- clean
 - Cleanse.cpp, [12](#)
 - Cleanse.h, [14](#)
- cleanpath
 - Cleanse.cpp, [12](#)
 - Cleanse.h, [14](#)
- cleanroute
 - Cleanse.cpp, [12](#)
 - Cleanse.h, [15](#)
- Cleanse.cpp, [11](#)
 - clean, [12](#)
 - cleanpath, [12](#)
 - cleanroute, [12](#)
 - complexclean, [13](#)
- Cleanse.h, [13](#)
 - clean, [14](#)
 - cleanpath, [14](#)
 - cleanroute, [15](#)
 - complexclean, [15](#)
- complexclean
 - Cleanse.cpp, [13](#)
 - Cleanse.h, [15](#)
- core
 - Service.cpp, [43](#)
 - Service.h, [48](#)
- correctdist
 - existindex, [5](#)
- correctparams
 - Service.cpp, [44](#)
- Service.h, [48](#)
- dist
 - lcities::ldist, [9](#)
- doparameters
 - Service.cpp, [44](#)
 - Service.h, [49](#)
- errorid
 - Mapheader.h, [24](#)
- existA
 - existindex, [5](#)
- existB
 - existindex, [6](#)
- existindex, [5](#)
 - correctdist, [5](#)
 - existA, [5](#)
 - existB, [6](#)
 - locationptrA, [6](#)
 - locationptrB, [6](#)
- existroad
 - Loading.cpp, [18](#)
 - Loading.h, [21](#)
- existroadAtoB
 - pathfinding.cpp, [27](#)
 - pathfinding.h, [35](#)
- existV
 - pathfinding.cpp, [27](#)
 - pathfinding.h, [36](#)
- findiname
 - pathfinding.cpp, [28](#)
 - pathfinding.h, [36](#)
- findinpath
 - pathfinding.cpp, [28](#)
 - pathfinding.h, [37](#)
- findinvisited
 - pathfinding.cpp, [29](#)
 - pathfinding.h, [37](#)
- findmin
 - pathfinding.cpp, [29](#)
 - pathfinding.h, [38](#)
- findshortAB
 - pathfinding.cpp, [30](#)
 - pathfinding.h, [39](#)
- getweight
 - pathfinding.cpp, [31](#)
 - pathfinding.h, [39](#)
- initialize

- pathfinding.cpp, 31
 - pathfinding.h, 40
- interpret
 - pathfinding.cpp, 32
 - pathfinding.h, 40
- lcities, 6
 - city, 7
 - pNextCity, 7
 - pNextDist, 7
- lcities::ldist, 8
 - dist, 9
 - pNextCity, 9
 - pNextDist, 9
- ldirect, 7
 - lvNext, 8
 - vertex, 8
- Loading.cpp, 16
 - addcity, 17
 - addroad, 17
 - existroad, 18
 - loadroad, 18
- Loading.h, 19
 - addcity, 20
 - addroad, 20
 - existroad, 21
 - loadroad, 21
- loadroad
 - Loading.cpp, 18
 - Loading.h, 21
- locationptrA
 - existindex, 6
- locationptrB
 - existindex, 6
- lvNext
 - ldirect, 8
- main
 - Mapa.cpp, 22
- mainexception
 - Service.cpp, 45
 - Service.h, 50
- Mapa.cpp, 22
 - main, 22
- Mapheader.h, 23
 - errorid, 24
- nextpathtab
 - pathtab, 10
- noroad
 - Service.cpp, 46
 - Service.h, 50
- pNextCity
 - lcities, 7
 - lcities::ldist, 9
- pNextDist
 - lcities, 7
 - lcities::ldist, 9
- pathfinding.cpp, 24
 - allroadexist, 26
 - allvisited, 26
 - existroadAtoB, 27
 - existV, 27
 - findiname, 28
 - findinpath, 28
 - findinvisited, 29
 - findmin, 29
 - findshortAB, 30
 - getweight, 31
 - initialize, 31
 - interpret, 32
 - weightless, 32
 - writeroute, 33
- pathfinding.h, 33
 - allroadexist, 34
 - allvisited, 35
 - existroadAtoB, 35
 - existV, 36
 - findiname, 36
 - findinpath, 37
 - findinvisited, 37
 - findmin, 38
 - findshortAB, 39
 - getweight, 39
 - initialize, 40
 - interpret, 40
 - weightless, 41
 - writeroute, 41
- pathtab, 9
 - nextpathtab, 10
 - pvertex, 10
 - sdist, 10
 - vertex, 10
 - visited, 10
- printhelp
 - Service.cpp, 46
 - Service.h, 51
- pvertex
 - pathtab, 10
- sdist
 - pathtab, 10
- Service.cpp, 42
 - allexcep, 43
 - core, 43
 - correctparams, 44
 - doparameters, 44
 - mainexception, 45
 - noroad, 46
 - printhelp, 46
- Service.h, 46
 - allexcep, 47
 - core, 48
 - correctparams, 48
 - doparameters, 49
 - mainexception, 50
 - noroad, 50

- printhelp, [51](#)
- Valid.cpp, [51](#)
 - charonly, [51](#)
- Valid.h, [52](#)
 - charonly, [52](#)
- vertex
 - ldirect, [8](#)
 - pathtab, [10](#)
- visited
 - pathtab, [10](#)
- weightless
 - pathfinding.cpp, [32](#)
 - pathfinding.h, [41](#)
- writeroute
 - pathfinding.cpp, [33](#)
 - pathfinding.h, [41](#)