Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektrotechniki i Informatyki

Podstawy Programowania Komputerów

Mapa

Autor	Tomasz Skowron
Prowadzący	dr inż. Krzysztof Simiński
Rok akademicki	2017/2018
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	1
Termin laboratorium/ćwiczeń	Wtorek, 13:45-15:15
Grupa	3
Sekcja	7
Termin oddania sprawozdania	Czwartek, 25 stycznia 2018, 23:55
Data oddania sprawozdania	Czwartek, 25 stycznia 2018

1. Treść zadania

Napisać program, który umożliwia znalezienie najkrótszej trasy między dwoma miastami. Miasta połączone są drogami o pewnej długości. Drogi są jednokierunkowe. Plik mapy dróg ma następującą postać.
<miasto początkowe> <miasto końcowe> <odległość>

Przykładowy plik dróg (liczba dróg nie jest ograniczona)
Katowice Kraków 70
Kraków Tarnów 70
Tarnów Jasło 50 ….
…….
Drugim plikiem wejściowym jest plik z trasami do wyznaczenia. Każda linia zawiera jedną trasę w postaci:
<miasto początkowe> <miasto końcowe>

Przykładowo:
Katowice Toruń
Kraków Poznań

Wynikiem działania programu jest plik wyjściowy z wyznaczonymi trasami, tzn. podana jest nazwa trasy, całkowita długość , a potem poszczególne odcinki z długościami, np.

trasa: Katowice → Toruń (355 km):
Katowice → Częstochowa 70
Częstochowa →Łódź 120
Łódź → Toruń

trasa: Tarnów→Wrocław: nie możliwa do wyznaczenia

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):
-d plik wejściowy z drogami
-t plik wejściowy z trasami do wyznaczenia
-o plik wynikowy z wyznaczonymi trasami

2. Analiza zadania

a. Analiza ogólna

Zadanie dotyczy problemu wyszukiwania najkrótszej drogi z punktu A do punktu B, na podstawie znanych połączeń między węzłami (miastami, punktami). W programie przyjęto nazwę **„mapa" dla struktury danych** zawierającą informacje o drogach i miastach, **nie jest to tablica asocjacyjna.**

b. Struktury danych

W programie wykorzystanie listy jednokierunkowe do przechowywania informacji o miastach oraz drogach je łączących. Miasta ułożone są w liście elementów **lcities**. Każde miasto posiada podwieszoną listę obiektów **lcities::ldist** z drogami z niego wychodzącymi. Każda droga posiada wartość (odległość) oraz miasto docelowe. Taka forma przechowywania informacji o drogach jest bardzo logiczna i łatwa w zarówno implementacji, jak i interpretacji. Struktura ta dalej zwana jest mapą. Jest ona uniwersalna, gdyż nie posiada przypisanych zmiennych, ani wartości dla użytego algorytmu (zostały one przypisane do struktury poniżej).

Do odnalezienia najkrótszej drogi z A do B wykorzystano również listę elementów **pathtab** jednokierunkową zawierającą informacje o odległości każdego odwiedzonego miasta od miasta A, przez jakie miasto poprzedzające jest ta odległość oraz informację o odwiedzeniu tego miasta przez algorytm lub też nie. Główną zaletą tej struktury jest umożliwienie odseparowania wartości edytowanych przez algorytm od struktury mapy. Dzięki niej struktura mapy staje się uniwersalna.

Gdy algorytm zakończy poszukiwania wykorzystywana jest lista **ldirect**, której elementy wskazują na wszystkie konieczne ,do dotarcia z miasta A do B, wierzchołki, tym samym pomijając te nieodwiedzone lub niepotrzebne. Struktura pozwala uprościć proces usuwania nadmiaru odległościowego, który powstaje w liście informacyjnej po zastosowaniu algorytmu Dijkstry.

c. Algorytmy

Program wykorzystuje algorytm Dijkstry na strukturach wymienionych powyżej.

d. Złożoność algorytmu

Algorytm jest określony złożonością $O((n^2)+m)$, gdzie n oznacza ilość miast, a m oznacza ilość dróg.

3. Specyfikacja zewnętrzna

a. Program uruchamiany jest z linii poleceń. Do pracy z programem wymagane jest podanie pliku (po przełączniku **-d**) z drogami w postaci takiej jak w treści zadania. Bez tego program nie wykona się i wypisze stosowny komunikat. Przełączniki **–t** oraz **–o** nie są konieczne. Ich brak będzie zastąpiony komunikacją z użytkownikiem przez standardowe wyjścia.

b. Przykładowo program można uruchomić w następujący sposób:
program –d mapa.txt –t szukam.txt –o wynik.txt
Kolejność parametrów nie ma znaczenia.
Dodatkowo można skorzystać z przełącznika **–h**, który spowoduje wyświetlenie krótkiego menu pomocy.

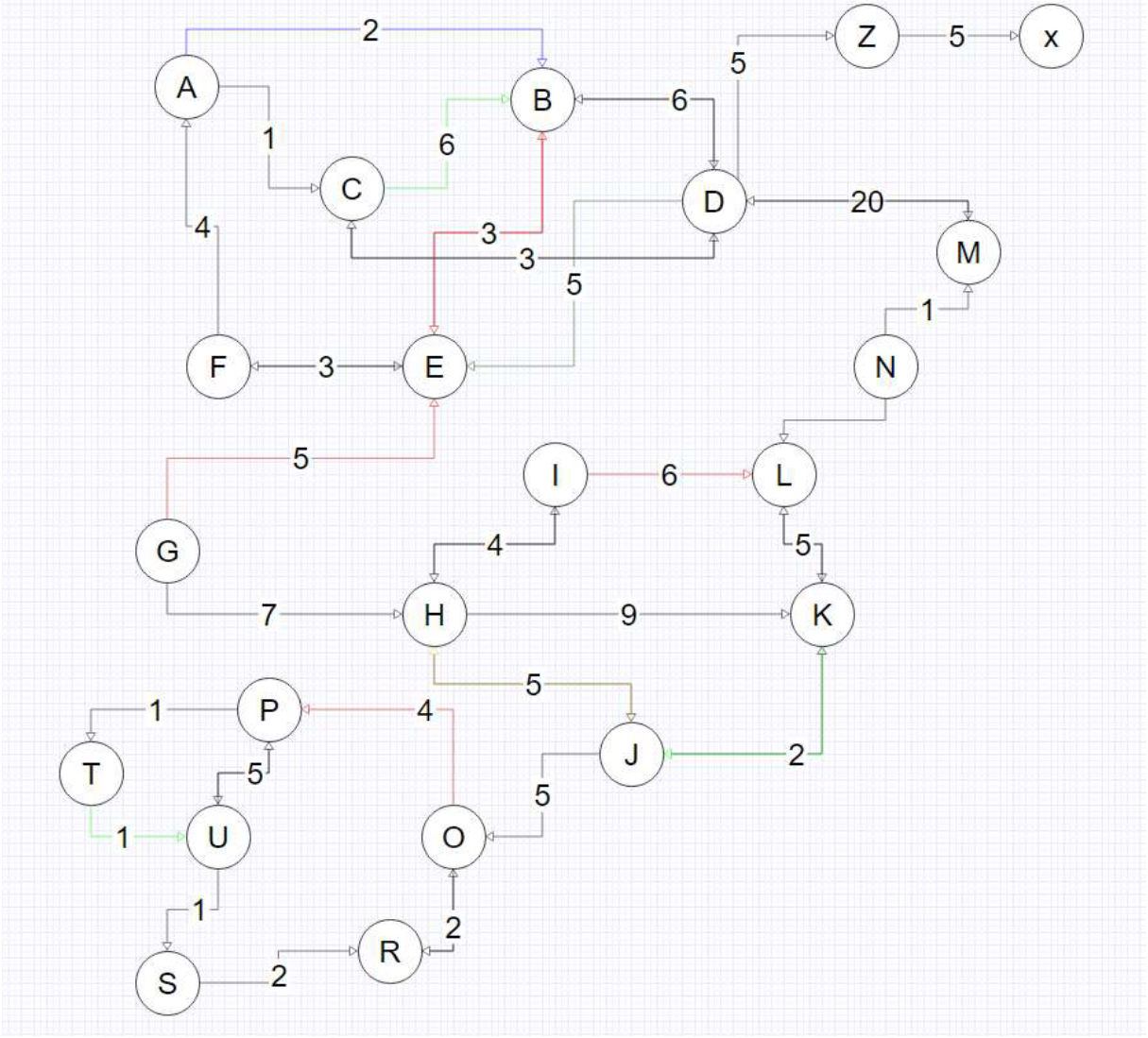
4. Testowanie

Program został przetestowany na przykładowej prostej mapie, zawartej w załączonym pliku data.txt oraz na mapie złożonej z ślepych uliczek, a także miast nieosiągalnych. W każdym z przypadków zwrócona została poprawna trasa. Program posiada ograniczenia poprawności działania dla dystansów z zakresu zmiennej typu „double”. Algorytmy zastosowane w programie nie powinny stawiać górnej wartości maksymalnej wielkości plików wejściowych.

5. Wnioski.

Program można zaliczyć do średnio skomplikowanych. Implementacja algorytmu była wymagająca, ale trudności wynikały głównie z faktu istnienia skrajnych przypadków. Najtrudniejszym elementem było odnalezienie wycieków pamięci, które występowały nie zawsze, a tylko po wprowadzeniu w konkretnej kolejności konkretnych danych poszukiwanych tras. Jak się okazało, wynikało to z zastosowania optymalizacji w algorytmie, który przerywał się zanim zostały odwiedzone wszystkie miasta.

6. Dane przykładowe – wizualizacja



Dane te zostały załączone w pliku data2.txt

Mapa

Wygenerowano przez Doxygen 1.8.14

Spis treści

Rozdział 1

Indeks klas

1.1 Lista klas

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

Rozdział 2

Indeks plików

2.1 Lista plików

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

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury existindex

Struktura dla funkcji existroad, z informacjami o (nie)istnieniu miast oraz wskaznikach na nie. Struktura jest uzupelniana funkcja existroad, by stanowic podstawy do uzupelniania listy przy ladowaniu 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 wskaznikach na nie. Struktura jest uzupelniana funkcja existroad, by stanowic podstawy do uzupelniania listy przy ladowaniu miast lub nie.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 correctdist

`int existindex::correctdist`

Czy dystans miedzy miastami A i B jest poprawny

Ostrzeżenie

Wartosc tego elementu jest scisle okreslona.
"0" - jesli taki dystans nie istnieje
"1" - jesli dystans podany z pliku jest taki jak juz byl podany wczesniej
"2" - jesli wczesniej 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, wasdiffrentdist=2=byl inny Wskaznik na miasto A jesli istnieje, nullptr jesli nie istnieje

3.1.2.5 locationptrB

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

Wskaznik na miasto B jesli istnieje, nullptr jesli nie istnieje

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Mapheader.h

3.2 Dokumentacja struktury lcities

Struktura mapy zlozonej z drog jednokierunkowych. lcities to element skladajacy sie na liste list. Struktura reprezentuje miasto. Miasta tworza liste jednokierunkowa. Kazdemu miastu przypisana jest droga lub lista drog wychodzacych do innych miast.

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

Komponenty

- struct ldist

  Struktura dystansow w mapie.

Atrybuty publiczne

- string city
- lcities * pNextCity = nullptr
- ldist * pNextDist = nullptr

3.2.1 Opis szczegółowy

Struktura mapy zlozonej z drog jednokierunkowych. lcities to element skladajacy sie na liste list. Struktura reprezentuje miasto. Miasta tworza liste jednokierunkowa. Kazdemu miastu przypisana jest droga lub lista drog wychodzacych 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 nastepne miasto w liscie

3.2.2.3 pNextDist

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

wskaznik na pierwsza droge wychodzaca 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 ktore nalezy isc aby dotrzec z miasta A do miasta B, a dokladniej z miasta B do miasta A,
poniewaz funkcje poruszaja sie z vertex do pvertex poczynajac od vertex wskazujacy 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 ktore nalezy isc aby dotrzec z miasta A do miasta B, a dokladniej z miasta B do miasta A,
poniewaz funkcje poruszaja sie z vertex do pvertex poczynajac od vertex wskazujacy na miasto B.

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 lvNext

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

Wskaznik na nastepny 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 dystansow w mapie.

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

Atrybuty publiczne

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

3.4.1 Opis szczegółowy

Struktura dystansow w mapie.

Ostrzeżenie

Struktura dalej zwana mapa. ldist zdefiniowane w lcities, to struktura zawierajaca wartosc, interpretowana jako odleglosc lub droga o danym dystansie z miasta wskazujacego na ten element
do miasta, na ktory element wskazuje. Elementy tworza liste jednokierunkowa, ktora jest podwieszona do jakiegos miasta.
W tabeli X oznacza zupelnie dowolne miasto/miasta o n-tym/n-tych indeksie/indeksach.
"...->V.X." to wskaznik na liste w dol tabeli
"...->>>.." to wskaznik na nastepny element w wierszu tabeli.
Struktura rzeczywista moze sie znaczaco roznic od przykladowej w niektorych przypadkach co do wartosci, 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 została 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 twrzona dla wszystkich istniejacych wierzcholkow niezaleznie od jakiejkolwiek zmiennej, gdyz nie wiemy ile wierzcholkow (ani ktore) musi znalezc sie na trasie.

3.5.2 Dokumentacja atrybutów składowych

3.5.2.1 nextpathtab

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

Wskaznik na kolejny element listy/tabeli pathtab

3.5.2.2 pvertex

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

Wskaznik na wiercholek z ktorego przychodzi Dijkstra

3.5.2.3 sdist

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

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

3.5.2.4 vertex

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

Wskaznik na wiercholek

3.5.2.5 visited

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

Czy bylo 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 powstalych w wyniku dzialania 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)

  Wywoluje usuwanie podanych struktur dwokrotnie, po strukturze nalezy podac 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominieciu dla false.

4.1.1 Opis szczegółowy

Plik zawiera funkcje do usuwania struktur powstalych w wyniku dzialania 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 podrzedna obiektow ldist wlacznie.

Parametry

pName	- wskaznik na strukture do usuniecia

Zwraca

void

4.1.2.2 cleanpath()

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

Funkcja usuwa liste elementow typu pathtab.

Parametry

path	- wskaznik na strukture do usuniecia

Zwraca

void

4.1.2.3 cleanroute()

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

Funkcja usuwa strukture typu ldirect.

Parametry

route	- 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 )
```

Wywoluje usuwanie podanych struktur dwokrotnie, po strukturze nalezy podac 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominieciu dla false.

Ostrzeżenie

Funkcja jest tylko medium do wywolania funkcji czyszczacych poszczegolne struktury. Wywoluje clean(), cleanroute(), cleanpath() w zaleznosci od parametrow typu bool.

Parametry

pName	- wskaznik na strukture lcities
pn	- 1 usun, 0 nie usuwaj
path	- wskaznik na strukture lcities
pt	- 1 usun, 0 nie usuwaj
route	- wskaznik na strukture lcities
rt	- 1 usun, 0 nie usuwaj

Zwraca

void

4.2 Dokumentacja pliku Cleanse.h

Plik zawiera funkcje do usuwania struktur powstalych w wyniku dzialania 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)

  Wywoluje usuwanie podanych struktur dwokrotnie, po strukturze nalezy podac 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominieciu dla false.

4.2.1 Opis szczegółowy

Plik zawiera funkcje do usuwania struktur powstalych w wyniku dzialania 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

pName	- wskaznik na strukture do usuniecia

Zwraca

void

4.2.2.2 cleanpath()

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

Funkcja usuwa liste elementow typu pathtab.

Parametry

path	- wskaznik na strukture do usuniecia

Zwraca

void

4.2.2.3 cleanroute()

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

Funkcja usuwa strukture typu ldirect.

Parametry

route	- 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 )
```

Wywoluje usuwanie podanych struktur dwokrotnie, po strukturze nalezy podac 1/0 lub true false co decyduje o usuwaniu struktury dla true i pominieciu dla false.

Ostrzeżenie

Funkcja jest tylko medium do wywolania funkcji czyszczacych poszczegolne struktury. Wywoluje clean(), cleanroute(), cleanpath() w zaleznosci od parametrow typu bool.

Parametry

pName	- wskaznik na strukture lcities
pn	- 1 usun, 0 nie usuwaj
path	- wskaznik na strukture lcities
pt	- 1 usun, 0 nie usuwaj
route	- wskaznik na strukture lcities
rt	- 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 wskaznikami na szukane elementy o ile istnialy.
- 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 uproscic proces poszukuwania wskaznikow 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 sprawdzajac ich poprawnosc 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 stabilnosci oraz poprawnosci dzialania programu powinno nie istniec w strukturze.

Parametry

pName	- wskaznik na strukture mapy
cityA	- nazwa miasta A

Zwraca

void

4.3.2.2 addroad()

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

Funkcja dodaje droge, tzn dystans miedzy miastami A B. Funkcja wymaga podania struktury existindex by uproscic proces poszukuwania wskaznikow na odpowiednie miasta, te zas sa dodawane do struktury existindex przez funkcje existroad.

Ostrzeżenie

Funkcja nie sprawdza czy miasta istnialy, ani czy drogi istnialy i dla stabilnosci oraz poprawnosci dzialania programu musza one istniec.

Parametry

pName	- wskaznik na strukture mapy
dist	- dystans do dodania
cityloc	- struktura zawierajaca wskazniki na miasta, miedzy ktorymi wystepuje 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 istnialy.

Parametry

pName	- wskaznik na strukture mapy
cityA	- nazwa miasta A
distance	- dystans miedzy miastami
cityB	- 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 sprawdzajac ich poprawnosc tworzy strukture mapy.

Parametry

pName	- wskaznik na strukture mapy
infilename	- sciezka do pliku wejsciowego z danymi do mapy

Zwraca

void

Wyjątki

nominusd	-throw, gdy byl blad pliku wejsciowego
maplineerr	- blad w linii w pliku
wasdiffrentdist	- byl juz podany inny dystans

4.4 Dokumentacja pliku Loading.h

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 <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 strukture existindex ze wskaznikami na szukane elementy o ile istnialy.
- 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 uproscic proces poszukuwania wskaznikow 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 sprawdzajac ich poprawnosc tworzy strukture mapy.

4.4.1 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

4.4.2 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 stabilnosci oraz poprawnosci dzialania programu powinno nie istniec w strukturze.

Parametry

pName	- wskaznik na strukture mapy
cityA	- nazwa miasta A

Zwraca

void

4.4.2.2 addroad()

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

Funkcja dodaje droge, tzn dystans miedzy miastami A B. Funkcja wymaga podania struktury existindex by uproscic proces poszukuwania wskaznikow na odpowiednie miasta, te zas sa dodawane do struktury existindex przez funkcje existroad.

Ostrzeżenie

Funkcja nie sprawdza czy miasta istnialy, ani czy drogi istnialy i dla stabilnosci oraz poprawnosci dzialania programu musza one istniec.

Parametry

pName	- wskaznik na strukture mapy
dist	- dystans do dodania
cityloc	- struktura zawierajaca wskazniki na miasta, miedzy ktorymi wystepuje 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 istnialy.

Parametry

pName	- wskaznik na strukture mapy
cityA	- nazwa miasta A
distance	- dystans miedzy miastami
cityB	- 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 sprawdzajac ich poprawnosc tworzy strukture mapy.

Parametry

pName	- wskaznik na strukture mapy
infilename	- sciezka do pliku wejsciowego z danymi do mapy

Zwraca

void

Wyjątki

nominusd	-throw, gdy byl blad pliku wejsciowego
maplineerr	- blad w linii w pliku
wasdiffrentdist	- byl juz 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[ ])

  Glowna funkcja programu.

4.5.1 Opis szczegółowy

Plik z glowna funkcja programu main().

Autor

Tomasz Skowron

4.5.2 Dokumentacja funkcji

4.5.2.1 main()

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

Glowna funkcja programu.

Ostrzeżenie

W programie przyjeto nazwe mapa dla struktury z miastami i drogami, nie jest to tablica asocjacyjna. Funkcja wywoluje obsluge parametrow. Funkcja wywola w odpowiedni sposob funkcje core, zastepujac brakujace parametry standardowymi wyjsciami. Zostanie rowniez wywolane ladowanie struktury mapy z pliku.
funkcja moze zglosic wyjatek, ale zostanie on obsluzony w bloku catch na koncu main(),
funkcja wylapuje rowniez wszystkie wyjatki spadajace do niej z podfunkcji.

Parametry

paramixmax	-zmienna z iloscia parametrow
params	-tablica parametrow

Zwraca

0

4.6 Dokumentacja pliku Mapheader.h

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

Komponenty

- struct lcities

  Struktura mapy zlozonej z drog jednokierunkowych. lcities to element skladajacy sie na liste list. Struktura reprezentuje miasto. Miasta tworza liste jednokierunkowa. Kazdemu miastu przypisana jest droga lub lista drog wychodzacych do innych miast.
- struct lcities::ldist

  Struktura dystansow w mapie.
- struct existindex

  Struktura dla funkcji existroad, z informacjami o (nie)istnieniu miast oraz wskaznikach na nie. Struktura jest uzupelniana funkcja existroad, by stanowic podstawy do uzupelniania listy przy ladowaniu miast lub nie.
- struct pathtab

  Struktura do podawania wag odlegloscIowych i poprawnego wykonania algorytmu Dijkstry, dalej zwana pathtab.
- struct ldirect

  Struktura drogi z A do B. Struktura wskazuje na elementy tablicy pathtab (pathtab), przez ktore nalezy isc aby dotrzec z miasta A do miasta B, a dokladniej z miasta B do miasta A,
  poniewaz funkcje poruszaja sie z vertex do pvertex poczynajac od vertex wskazujacy na miasto B.

Wyliczenia

- enum errorid {
  unknown, maperr, wasdiffrentdist, noroute,
  earlyhit, maplineerr, nominusd, nominust,
  nominuso, outferr, routeferr, noTnoO,
  donothing, nodist, okdist }

  Wyliczenie wyjatkow programu. Zawiera wszystkie identyfikatory wszystkich wyjatkow, ktore pojawiaja sie w programie.

4.6.1 Dokumentacja typów wyliczanych

4.6.1.1 errorid

```
enum errorid
```

Wyliczenie wyjatkow programu. Zawiera wszystkie identyfikatory wszystkich wyjatkow, ktore pojawiaja sie w programie.

Wartości wyliczeń

unknown	0 nieznany
maperr	1 blad pliku mapy
wasdiffrentdist	2 blad dystansu do existindex
noroute	3 nie ma drogi
earlyhit	4 znaleziono trase
maplineerr	5 blad linii w mapie
nominusd	6 brak pliku mapy
nominust	7 brak pliku tras od wyznaczenia
nominuso	8 brak pliku wyjsciowego
outferr	9 blad pliku wyjsciowego
routeferr	10 blad pliku z trasami do wyznaczenia
noTnoO	11 nie podano ani pliku z trasami, ani wyjsciowego
donothing	12 nie rob nic, koncz program
nodist	13 nie ma drogi do existindex
okdist	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 drogi 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 wierzcholkow struktuy mapy.
- pathtab ∗ findinpath (pathtab ∗&path, const string &cityA)

  Funkcja odnajduje i zwraca wskaznik na element z tablicy pathtab, ktory wskazuje na miasto o zadanej nazwie.
- lcities ∗ findiname (lcities ∗&pName, const string &cityA)

  Funkcja odnajduje i zwraca wskaznik na element ze struktury mapy, ktory wskazuje na miasto o zadanej nazwie.
- pathtab ∗ findmin (pathtab ∗&path, const string &cityA)

  Funkcja znajduje w tablicy pathtab wskaznik na minimalna wartosc wiercholka, co jest jednym z kloczowych krokow algorytmu Dijksty.
- bool allvisited (pathtab ∗&path, const string &cityA)

  Funkcja sprawdza czy jest istniejacy elemeny tablicy pathtab, ktory nie byl nigdy odwiedzony.
- pathtab findshortAB (lcities ∗&pName, const string &cityA, const string cityB, pathtab ∗&path)

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

  Sprawdza czy w strukturze ldirect istnieje element wskazujacy na miasto o zadanej nazwie.
- bool existroadAtoB (const string &cityA, const string &cityB, lcities ∗&pName)

  Sprawdza czy w strukturze mapy wystepuje droga z A do B. Ta funkcja rozni sie od existroad, gdyz sprawdza wylacznie istnienie i nie zwraca gdzie wystapily.
  Stworzone dla funkcji allroadexist().
- bool allroadexist (ldirect ∗&pVisited, lcities ∗&pName, const string &startcity)

  Sprawdza czy drogi miedzy wierzcholkami (vertex), a poprzednimi wierzcholkami (pvertex) w strukturze ldirect istnieja w strukturze mapy.
- ldirect ∗ interpret (pathtab ∗&path, const string &cityA, const string &cityB, lcities ∗&pName)

  Funkcja wypelnia strukture ldirect trasami a punktu A do B na podstawie wartosci 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 droge z A do B.
- void weightless (ldirect ∗&route, const string &endcity, const string &startcity)

  Funkcja przywroci dystanse z |X-Y| usuwajac odleglosc |A-X|, powstale w wyniku algorytmu Dijkstry.
- double getweight (pathtab ∗&path, const string &city)

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

4.7.1 Opis szczegółowy

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje drogi 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 miedzy wierzcholkami (vertex), a poprzednimi wierzcholkami (pvertex) w strukturze ldirect istnieja w strukturze mapy.

Ostrzeżenie

Wystarcza jedna nie isniejaca droga by zwrocono false.

Parametry

pVisited	- wskaznik na strukture trasy
pName	- wskaznik na strukture mapy
startcity	- wskaznik na miasto poczatkowe, z ktorego droge szukamy. Pozwala na wczesniejsze przerwanie zakonczenia dzialania funkcji.

Zwraca

true - gdy istnieja wszystkie drogi
false - gdy nie istnieje conajmniej jedno z miast na trasie

4.7.2.2 allvisited()

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

Funkcja sprawdza czy jest istniejacy elemeny tablicy pathtab, ktory nie byl nigdy odwiedzony.

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta pczatkowego nie wracamy), a swoja droga w prawie wszystkich przypadkach to miasto jest juz odwiedzone.

Zwraca

element - wskaznik na szukany element mapy

Wyjątki

noroute	- gdy nie odnaleziono elementu

4.7.2.3 existroadAtoB()

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

Sprawdza czy w strukturze mapy wystepuje droga z A do B. Ta funkcja rozni sie od existroad, gdyz sprawdza wylacznie istnienie i nie zwraca gdzie wystapily.
Stworzone dla funkcji allroadexist().

Parametry

cityA	- nazwa szukanego miasta A
cityB	- nazwa szukanego miasta B
pName	- wskaznik na strukture mapy

Zwraca

true - gdy istnieja oba
false - gdy nie istnieje conajmniej jedno z miast

4.7.2.4 existV()

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

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

Parametry

pVisited	- wskaznik na strukture ldirect
city	- nazwa szukanego miasta

Zwraca

true - gdy istnieje
false - gdy nie istnieje

Wyjątki

earlyhit	- gdy droga moze byc znaleziona wczesniej

4.7.2.5 findiname()

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

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

Ostrzeżenie

Funkcja moze zglosic wyjatek noroute, gdyz jesli szukamy jakiejs drogi, a miasto nie isnieje to nie mozna kontynuowac poszukiwan. findinpath jest zatem niemal bezuzyteczna poza findshortAB().

Parametry

pName	- wskaznik na tablice pathtab
cityA	- nazwa szukanego miasta

Zwraca

element - wskaznik na szukany element mapy

Wyjątki

noroute	- gdy nie odnaleziono elementu

4.7.2.6 findinpath()

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

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

Ostrzeżenie

Nie wolno szukac elementow ktore nie istnieja, szczegolnie jesli wywolujemy to z funkcji findshortAB().

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa szukanego miasta

Zwraca

element - wskaznik 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 poszczegolne dystanse z tablicy pathtab. Poleca sie uzycie weightless na tablicy pathtab by otrzymac poprawne dane, z |X-Y|, a nie |A-X|+|X-Y|.

Zobacz również

wieghtless
findshortAB

Parametry

pVisited	- wskaznik na strukture trasy z A do B
city	- nazwa szukanego miasta

Zwraca

pHeadV - wskaznik 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 wskaznik na minimalna wartosc wiercholka, co jest jednym z kloczowych krokow algorytmu Dijksty.

Ostrzeżenie

Minimimum moze okazac sie nieskonczonoscia, czyli 0. Funkcja pomija miasto A oraz miasta odwiedzone, wiec jesli wszystkie byly odwiedzone to zwroci wartosc poczatku tablicy pathtab.
Findmin porownoje wagi (value, czyli sdist), a nie dystanse ze struktury mapy!

Parametry

path	- wskaznik na pathtab
cityA	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta pczatkowego nie wracamy)

Zwraca

min - wskaznik na minimalny pod wzgledem sdist i zgodny z algorytmem element tablicy pathtab

Wyjątki

noroute	gdy miasto nie istnieje

4.7.2.9 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 odlegloscio m |X-Y|, tylko |A-X|+|X-Y|. Aby miec w tablicy odleglosci miedzy poszczegolnymi wierzcholkami nalezy ja podac funkcji weightless()

Parametry

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

Zwraca

path - uzupelniona wagami tablica pathtab

Wyjątki

earlyhit	- jesli droga zostala znaleziona wczesniej, niz odwiedzenie wszystkich miast przez Dijkstre
noroute	- 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 ostatecznocia ograniczana przez findmin, a jesli tak sie stalo to alloradexist ponownie zabezpiecza niepoprawny wynik.

4.7.2.10 getweight()

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

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

Ostrzeżenie

Szczegolnie dobym uzyciem tej funkcji jest zebranie odleglosci z A do B przez uzyciem weightless na tablicy pathtab, daje to gotowa sume dystansow.
Droga z A do B musi istniec.

Parametry

path	- wskaznik na tablice pathtab
city	- nazwa miasta o szukanej wadze

Zwraca

value - waga odleglosci
0 - gdy waga jest nie mozliwa do odnalezienia

4.7.2.11 initialize()

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

Funkcja inicjalizuje liste (tablice) pathtab na podstawie wierzcholkow struktuy mapy.

Ostrzeżenie

Wartosci domyslne elelentow listy to: new pathtab{ pHeadN,0,nullptr,false,nullptr }

Parametry

pName	- wskaznik na strukture mapy
path	- wskaznik na pathtab do utworzenia

Zwraca

path - wskaznik na zainicjalizowana tablice
nullptr -jesli pName bylo puste

4.7.2.12 interpret()

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

Funkcja wypelnia strukture ldirect trasami a punktu A do B na podstawie wartosci vertex i pvertex z tablicy pathtab.

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa miasta A
cityB	- nazwa miasta B
pName	- wskaznik na strukture mapy

Zwraca

pVisited - struktura zawierajaca intersujaca nas droge z A do B
nullptr - w wypadku niepowodzenia/przerwania petli

Wyjątki

noroute	- gdy funkcja otrzyma sygnal, ze droga nie istnieje

4.7.2.13 weightless()

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

Funkcja przywroci dystanse z |X-Y| usuwajac odleglosc |A-X|, powstale w wyniku algorytmu Dijkstry.

Parametry

route	- wskaznik na strukture trasy z A do B
endcity	- nazwa miasta koncowego
startcity	- wskaznik na miasto poczatkowe

Zwraca

void

4.7.2.14 writeroute()

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

Funkcja wypisuje do zadanego strumienia droge z A do B.

Parametry

route	- wskaznik na strukture trasy
stream	- strumien wyjsciowy do zapisu

Zwraca

void

4.8 Dokumentacja pliku pathfinding.h

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

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

Funkcje

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

  Funkcja inicjalizuje liste (tablice) pathtab na podstawie wierzcholkow struktuy mapy.
- pathtab * findinpath (pathtab *&path, const string &cityA)

  Funkcja odnajduje i zwraca wskaznik na element z tablicy pathtab, ktory wskazuje na miasto o zadanej nazwie.
- lcities * findiname (lcities *&pName, const string &cityA)

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

- pathtab ∗ findmin (pathtab ∗&path, const string &cityA)

  Funkcja znajduje w tablicy pathtab wskaznik na minimalna wartosc wiercholka, co jest jednym z kloczowych krokow algorytmu Dijksty.
- bool allvisited (pathtab ∗&path, const string &cityA)

  Funkcja sprawdza czy jest istniejacy elemeny tablicy pathtab, ktory nie byl nigdy odwiedzony.
- pathtab findshortAB (lcities ∗&pName, const string &cityA, const string cityB, pathtab ∗&path)

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

  Sprawdza czy w strukturze ldirect istnieje element wskazujacy na miasto o zadanej nazwie.
- bool existroadAtoB (const string &cityA, const string &cityB, lcities ∗&pName)

  Sprawdza czy w strukturze mapy wystepuje droga z A do B. Ta funkcja rozni sie od existroad, gdyz sprawdza wylacznie istnienie i nie zwraca gdzie wystapily.
  Stworzone dla funkcji allroadexist().
- bool allroadexist (ldirect ∗&pVisited, lcities ∗&pName, const string &startcity)

  Sprawdza czy drogi miedzy wierzcholkami (vertex), a poprzednimi wierzcholkami (pvertex) w strukturze ldirect istnieja w strukturze mapy.
- ldirect ∗ interpret (pathtab ∗&path, const string &cityA, const string &cityB, lcities ∗&pName)

  Funkcja wypelnia strukture ldirect trasami a punktu A do B na podstawie wartosci 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 droge z A do B.
- void weightless (ldirect ∗&route, const string &endcity, const string &startcity)

  Funkcja przywroci dystanse z |X-Y| usuwajac odleglosc |A-X|, powstale w wyniku algorytmu Dijkstry.
- double getweight (pathtab ∗&path, const string &city)

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

4.8.1 Opis szczegółowy

Plik zawiera funkcje kluczowe do wykonania algorytmu Dijkstry i znalezienia o ile istnieje drogi 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,
            lcities *& pName,
            const string & startcity )
```

Sprawdza czy drogi miedzy wierzcholkami (vertex), a poprzednimi wierzcholkami (pvertex) w strukturze ldirect istnieja w strukturze mapy.

Ostrzeżenie

Wystarcza jedna nie isniejaca droga by zwrocono false.

Parametry

pVisited	- wskaznik na strukture trasy
pName	- wskaznik na strukture mapy
startcity	- wskaznik na miasto poczatkowe, z ktorego droge szukamy. Pozwala na wczesniejsze przerwanie zakonczenia dzialania funkcji.

Zwraca

true - gdy istnieja wszystkie drogi
false - gdy nie istnieje conajmniej jedno z miast na trasie

4.8.2.2 allvisited()

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

Funkcja sprawdza czy jest istniejacy elemeny tablicy pathtab, ktory nie byl nigdy odwiedzony.

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta pczatkowego nie wracamy), a swoja droga w prawie wszystkich przypadkach to miasto jest juz odwiedzone.

Zwraca

element - wskaznik na szukany element mapy

Wyjątki

noroute	- gdy nie odnaleziono elementu

4.8.2.3 existroadAtoB()

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

Sprawdza czy w strukturze mapy wystepuje droga z A do B. Ta funkcja rozni sie od existroad, gdyz sprawdza wylacznie istnienie i nie zwraca gdzie wystapily.
Stworzone dla funkcji allroadexist().

Parametry

cityA	- nazwa szukanego miasta A
cityB	- nazwa szukanego miasta B
pName	- wskaznik na strukture mapy

Zwraca

true - gdy istnieja oba
false - gdy nie istnieje conajmniej jedno z miast

4.8.2.4 existV()

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

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

Parametry

pVisited	- wskaznik na strukture ldirect
city	- nazwa szukanego miasta

Zwraca

true - gdy istnieje
false - gdy nie istnieje

Wyjątki

earlyhit	- gdy droga moze byc znaleziona wczesniej

4.8.2.5 findiname()

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

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

Ostrzeżenie

Funkcja moze zglosic wyjatek noroute, gdyz jesli szukamy jakiejs drogi, a miasto nie isnieje to nie mozna kontynuowac poszukiwan. findinpath jest zatem niemal bezuzyteczna poza findshortAB().

Parametry

pName	- wskaznik na tablice pathtab
cityA	- nazwa szukanego miasta

Zwraca

element - wskaznik na szukany element mapy

Wyjątki

noroute	- gdy nie odnaleziono elementu

4.8.2.6 findinpath()

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

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

Ostrzeżenie

Nie wolno szukac elementow ktore nie istnieja, szczegolnie jesli wywolujemy to z funkcji findshortAB().

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa szukanego miasta

Zwraca

element - wskaznik 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 poszczegolne dystanse z tablicy pathtab. Poleca sie uzycie weightless na tablicy pathtab by otrzymac poprawne dane, z |X-Y|, a nie |A-X|+|X-Y|.

Zobacz również

wieghtless
findshortAB

Parametry

pVisited	- wskaznik na strukture trasy z A do B
city	- nazwa szukanego miasta

Zwraca

pHeadV - wskaznik 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 wskaznik na minimalna wartosc wiercholka, co jest jednym z kloczowych krokow algorytmu Dijksty.

Ostrzeżenie

Minimimum moze okazac sie nieskonczonoscia, czyli 0. Funkcja pomija miasto A oraz miasta odwiedzone, wiec jesli wszystkie byly odwiedzone to zwroci wartosc poczatku tablicy pathtab.
Findmin porownoje wagi (value, czyli sdist), a nie dystanse ze struktury mapy!

Parametry

path	- wskaznik na pathtab
cityA	- nazwa miasta do ignorowania (wynika to z algorytmu Dijkstry, gdzie do miasta pczatkowego nie wracamy)

Zwraca

min - wskaznik na minimalny pod wzgledem sdist i zgodny z algorytmem element tablicy pathtab

Wyjątki

noroute	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 wierzcholkami nalezy ja podac funkcji weightless()

Parametry

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

Zwraca

path - uzupelniona wagami tablica pathtab

Wyjątki

earlyhit	- jesli droga zostala znaleziona wczesniej, niz odwiedzenie wszystkich miast przez Dijkstre
noroute	- 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 ostatecznocia 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

Szczegolnie dobym uzyciem tej funkcji jest zebranie odleglosci z A do B przez uzyciem weightless na tablicy pathtab, daje to gotowa sume dystansow.
Droga z A do B musi istniec.

Parametry

path	- wskaznik na tablice pathtab
city	- nazwa miasta o szukanej wadze

Zwraca

value - waga odleglosci
0 - gdy waga jest nie mozliwa do odnalezienia

4.8.2.11 initialize()

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

Funkcja inicjalizuje liste (tablice) pathtab na podstawie wierzcholkow struktuy mapy.

Ostrzeżenie

Wartosci domyslne elelentow listy to: new pathtab{ pHeadN,0,nullptr,false,nullptr }

Parametry

pName	- wskaznik na strukture mapy
path	- wskaznik na pathtab do utworzenia

Zwraca

path - wskaznik na zainicjalizowana tablice
nullptr -jesli pName bylo puste

4.8.2.12 interpret()

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

Funkcja wypelnia strukture ldirect trasami a punktu A do B na podstawie wartosci vertex i pvertex z tablicy pathtab.

Parametry

path	- wskaznik na tablice pathtab
cityA	- nazwa miasta A
cityB	- nazwa miasta B
pName	- wskaznik na strukture mapy

Zwraca

pVisited - struktura zawierajaca intersujaca nas droge z A do B
nullptr - w wypadku niepowodzenia/przerwania petli

Wyjątki

noroute	- gdy funkcja otrzyma sygnal, ze droga nie istnieje

4.8.2.13 weightless()

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

Funkcja przywroci dystanse z |X-Y| usuwajac odleglosc |A-X|, powstale w wyniku algorytmu Dijkstry.

Parametry

route	- wskaznik na strukture trasy z A do B
endcity	- nazwa miasta koncowego
startcity	- wskaznik na miasto poczatkowe

Zwraca

void

4.8.2.14 writeroute()

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

Funkcja wypisuje do zadanego strumienia droge z A do B.

Parametry

route	- wskaznik na strukture trasy
stream	- strumien wyjsciowy do zapisu

Zwraca

void

4.9 Dokumentacja pliku Service.cpp

Zawiera funkcje dla main(), do obslugi wprowadzanych danych, czy to w postaci argumentow linii polecen, czy danych bezposrednio 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 &rfpath)

  Funkcja sprawdza poprawnosc parametrow.
- void doparameters (int paramixmax, char ∗params[ ], string &mfpath, string &ofpath, string &rfpath, 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.

4.9.1 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.h

4.9.2 Dokumentacja funkcji

4.9.2.1 allexcep()

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

Funkcja obsluguje wszystkie wyjatki ktore moga wystapic w programie.

Zobacz również

errorid

Parametry

excep	- wyjatek (na podstawie wyliczenia errorid)

Zwraca

- "nieznany blad \n"
- "blad pliku wejsciowego 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 wyjsciowego \n"
- "blad pliku wyjsciowego \n"
- "blad pliku z drogami do wyznaczenia \n"
- "nie podano pliku z trasami, ani wyjsciowego \n"
- ""

4.9.2.2 core()

```
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.

Parametry

pName	- wskaznik na strukture mapy

Parametry

path	- wskaznik na strukture pathtab
route	- wskaznik na strukture ldirect, droge z A do B
outfile	- strumien wyjsciowy
rfile	- strumien wejsciowy z ktorego czytane sa poszukiwane trasy

Zwraca

void

4.9.2.3 correctparams()

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

Funkcja sprawdza poprawnosc parametrow.

Parametry

ifpath	- sciezka do pliku wejsciowego
ofpath	- sciezka do pliku wyjsciowego
rfpath	- sciezka do pliku z trasami

Zwraca

true - zawsze true, o ile wczesniej nie bylo bledu krytycznego (brak pliku mapy, niepoprawnie otwarty/nie otwarty plik)

Wyjątki

nominusd	gdy nie ma pliku z mapa
noTnoO	gdy nie ma ani pliku z trasami do wyznaczenia, ani pliku wyjsciowego
nominuso	gdy nie ma pliku wyjsciowego
nominust	gdy nie ma pliku z trasami do wyznaczenia

4.9.2.4 doparameters()

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

```
            string & mfpath,
            string & ofpath,
            string & rfpath,
            bool & wasrf,
            bool & wasof )
```

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

Parametry

	paramixmax	-zmienna z iloscia parametrow
	params	- tablica parametrow
out	*mfpath*	- sciezka do pliku z mapa
out	*ofpath*	- sciezka do pliku wyjsciowego
	rfpath	- sciezka do pliku z trasami do wyznaczenia
	wasrf	- (nie)byl plik z trasami
	wasof	- (nie)byl plik wyjsciowy

Zwraca

void

4.9.2.5 mainexception()

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

Funkcja obsluguje tylko czesc wyjatkow 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

excep	- wyjatek (na podstawie wyliczenia errorid)
wasrf	- czy byl plik z trasami
wasof	- 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 zaisuje do podanego strumienia wyjsciowego informacje o nieistniejacej drodze z A do B.

Parametry

outfile	- strumien wyjsciowy
cityA	- miasto poczatkowe a miasta A
cityB	- 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 obslugi wprowadzanych danych, czy to w postaci argumentow linii polecen, czy danych bezposrednio 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 &rfpath)

  Funkcja sprawdza poprawnosc parametrow.
- void doparameters (int paramixmax, char ∗params[ ], string &mfpath, string &ofpath, string &rfpath, 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

excep	- wyjatek (na podstawie wyliczenia errorid)

Zwraca

- "nieznany blad \n"
- "blad pliku wejsciowego 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 wyjsciowego \n"
- "blad pliku wyjsciowego \n"
- "blad pliku z drogami do wyznaczenia \n"
- "nie podano pliku z trasami, ani wyjsciowego \n"
- ""

4.10.2.2 core()

```
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.

Parametry

pName	- wskaznik na strukture mapy
path	- wskaznik na strukture pathtab
route	- wskaznik na strukture ldirect, droge z A do B
outfile	- strumien wyjsciowy
rfile	- strumien wejsciowy z ktorego czytane sa poszukiwane trasy

Zwraca

void

4.10.2.3 correctparams()

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

```
            const string & ofpath,
            const string & rfpath )
```

Funkcja sprawdza poprawnosc parametrow.

Parametry

ifpath	- sciezka do pliku wejsciowego
ofpath	- sciezka do pliku wyjsciowego
rfpath	- sciezka do pliku z trasami

Zwraca

true - zawsze true, o ile wczesniej nie bylo bledu krytycznego (brak pliku mapy, niepoprawnie otwarty/nie otwarty plik)

Wyjątki

nominusd	gdy nie ma pliku z mapa
noTnoO	gdy nie ma ani pliku z trasami do wyznaczenia, ani pliku wyjsciowego
nominuso	gdy nie ma pliku wyjsciowego
nominust	gdy nie ma pliku z trasami do wyznaczenia

4.10.2.4 doparameters()

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

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

Parametry

	paramixmax	-zmienna z iloscia parametrow
	params	- tablica parametrow
out	*mfpath*	- sciezka do pliku z mapa
out	*ofpath*	- sciezka do pliku wyjsciowego
	rfpath	- sciezka do pliku z trasami do wyznaczenia
	wasrf	- (nie)byl plik z trasami
	wasof	- (nie)byl plik wyjsciowy

Zwraca

void

4.10.2.5 mainexception()

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

Funkcja obsluguje tylko czesc wyjatkow 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

excep	- wyjatek (na podstawie wyliczenia errorid)
wasrf	- czy byl plik z trasami
wasof	- 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

outfile	- strumien wyjsciowy
cityA	- miasto poczatkowe a miasta A
cityB	- 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

input	- 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

input	- lancuch do przetworzeina

Zwraca

tempinputinternal - nowo utworzony string

Skorowidz