Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Mapa

autor	Patryk Lipka
prowadzący	mgr. inż. Agnieszka Michalczuk
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	czwartek, 13:00-14:30
sekcja	13
termin oddania sprawozdania	2019-01-19

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: W każdej linii podana jest jedna droga:

⟨miasto początkowe⟩ ⟨miasto końcowe⟩ ⟨odległość⟩

Przykładowy plik dróg (liczba dróg nie jest ograniczona):

```
Katowice Krakow 70
Krakow Tarnow 70
Tarnow Jaslo 50
Katowice Gliwice 22
Lodz Poznan 205
Gliwice Katowice 22
Katowice Czestochowa 70
Czestochowa Lodz 120
Lodz Torun 165
Krakow Katowice 70
Gliwice Wroclaw 180
```

Drugim plikiem wejściowym jest plik z trasami do wyznaczenia. Każda linia pliku zawiera jedną trasę w postaci:

⟨miasto początkowe⟩ ⟨miasto końcowe⟩

Przykładowy plik tras do wyznaczenia (liczba tras nie jest ograniczona):

```
Katowice Torun
Krakow Poznan
Tarnow Wroclaw
```

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 --> Torun (355 km):
   Katowice --> Czestochowa    70
   Czestochowa --> Lodz       120
   Lodz --> Torun             165

trasa: Krakow --> Poznan (465 km):
   Krakow --> Katowice         70
   Katowice --> Czestochowa    70
   Czestochowa --> Lodz       120
   Lodz --> Poznan            205

trasa: Tarnow --> Wroclaw
   TRASA NIEMOZLIWA DO WYZNACZENIA
```

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):

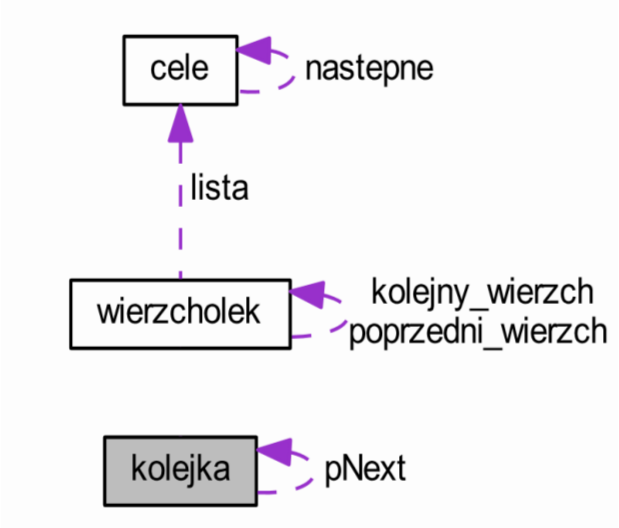
- `-d` plik wejściowy z drogami
- `-t` plik wejściowy z trasami do wyznaczenia
- `-o` plik wynikowy z wyznaczony trasami

2 Analiza zadania

Zadanie przedstawia problem wyznaczania najkrótszej drogi między dwoma punktami.

2.1 Struktury danych

W programie wykorzystano listę podwieszaną złożoną z dwóch struktur, służy ona do przechowywania poszczególnych dróg wraz z ich długościami. W programie możemy znaleźć jeszcze jedną strukturę służącą do stworzenia kolejki priorytetowej



2.2 Algorytm

Program wykorzystuje algorytm Dijkstry do wyznaczenia najktrótszej trasy między dwoma miastami. Wczytywane są dwa pliki: jeden z drogami, drugi z trasami do wyznaczenia po czym dane te są wykorzystane do stworzenia listy podwieszanej i przekazywane do funkcji wykorzystującej algorytm Dijkstry, która tworzy kolejkę priorytetową i po każdym obiegu

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: wejściowego i wyjściowego po odpowiednich przełącznikach (odpowiednio: -d dla pliku wejściowego z drogami, -t dla pliku wyjściowego z trasami do wyznaczenia i -o dla pliku wyjściowego z wyznaczonymi trasami) , np.

Project2 -d plik.txt -t plikW.txt -o wyniki.txt
Project2 -o wyniki.txt -d plik.txt -t plikW.txt

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (sortowania liczb).

4.1 Ogólna struktura programu

W funkcji głównej wywołana jest funkcja **wczytanie**, która sprawdza poprawność zapisania danych w pliku wejściowym z drogami i wywołuje funkcję **dodajWierzcholek** która dodaje miasto do listy ewentualnie jeśli takiej listy nie ma, tworzy listę miast, funkcja **dodajWierzcholek** wywołuje funkcję **szukaj**, która sprawdza czy dany wierzchołek już istnieje, jeśli tak, zwraca wskaźnik na ten wierzchołek, jeśli nie, zwraca nullptr i funkcja **dodajWierzcholek** tworzy nowy wierzchołek listy. Funkcja **szukaj** umożliwia połączenie celu z wierzchołkiem do którego ma być przypisany. Po stworzeniu listy podwieszanej wierzchołków i celów funkcja **wczytanie** wywołuje funkcję **wczytanieWynikow**, która wczytuje plik z trasami do wyznaczenia, a następnie wywołuje funkcję **dijkstra** wyznaczającą najkrótszą trasę między tymi dwoma miastami (jeśli taka istnieje). Funkcja **dijkstra** wykorzystuje do swojego działania dwie inne funkcje: **stworz_kolejke** i **sortuj**- pierwsza tworzy kolejkę priorytetową, druga natomiast sortuje ją, przed każdorazowym sortowaniem funkcja **dijkstra** usuwa sprawdzoną trasę.
Po wyznaczeniu tej trasy funkcja **wczytanieWynikow** wypisuje wyznaczoną trasę wraz z łączną odległością między tymi miastami, a następnie wywołuje funkcję **wypisywanie**, która wypisuje rekurencyjnie poszczególne odcinki wyznaczonej trasy, natomiast jeżeli trasa nie mogła być wyznaczona funkcja **wczytanieWynikow** wypisze odpowiedni komunikat do pliku z wynikami. Po wszystkim funkcja **wczytanie** wywołuje funkcję **usuwanieWierzcholka**, która wywołuje **usuwanieCelu**. Funkcje te usuwają listę podwieszaną.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany za pomocą różnych plików, dla plików z błędnymi danymi program wypisywał odpowiedni komunikat w pliku z wynikami. Dla miast, które są w pliku dróg i między którymi można wyznaczyć trasę program zwracał tą trasę wraz z poszczególnymi jej odcinkami, dla miast które są w pliku dróg, ale nie można wyznaczyć między nimi drogi program w pliku wynikowym wypisywał informację o braku połączenia między nimi, w przypadku podania miast lub miasta spoza pliku z drogami w pliku wynikowym widniała informacja o braku miasta w pliku z drogami. Program został także przetestowany pod kątem wycieków pamięci i jest od nich wolny.

6 Wnioski

Program okazał się dość sporym wyzwaniem w implementacji. Najważniejsze było zrozumienie działania algorytmu Dijkstry, który okazało się, że działa inaczej niż początkowo myślałem, dlatego musiałem nieco przebudować projekt. Nauczyło mnie to upewniania się w słuszności rozwiązania zanim napiszę odpowienie funkcje. Największą trudność sprawiło mi obsłużenie błędnych danych, ponieważ początkowo program działał dobrze, jednakże przy dodaniu odpowiednich zabezpieczeń pomijał wypisanie wyznaczanej trasy- naprawianie jednego problemu generowało następne.

Literatura

Stephen Prata *Język C++ Szkoła programowania, wydanie 6.*
Wydawnictwa Helion, 2013

Dodatek
Szczegółowy opis typów i funkcji

Mapa

Wygenerowano przez Doxygen 1.8.15

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 plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury cele

`#include <header.h>`

Diagram współpracy dla cele:

Atrybuty publiczne

- void * **wierzcholek**
- double **dystans**
- **cele** * **nastepne**

3.1.1 Opis szczegółowy

Struktura zawiera cele poszczegolnych drog

Definicja w linii 8 pliku header.h.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 dystans

```
double cele::dystans
```

Definicja w linii 11 pliku header.h.

3.1.2.2 nastepne

```
cele* cele::nastepne
```

Definicja w linii 12 pliku header.h.

3.1.2.3 wierzcholek

```
void* cele::wierzcholek
```

Definicja w linii 10 pliku header.h.

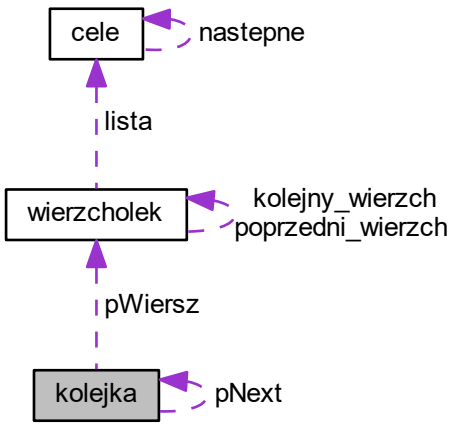
Dokumentacja dla tej struktury została wygenerowana z pliku:

- **header.h**

3.2 Dokumentacja struktury kolejka

```
#include <header.h>
```

Diagram współpracy dla kolejka:



Atrybuty publiczne

- **wierzcholek** ∗ **pWiersz**
- **kolejka** ∗ **pNext**

3.2.1 Opis szczegółowy

Struktura wykorzystana do stworzenia kolejki priorytetowej

Definicja w linii 28 pliku header.h.

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 pNext

```
kolejka* kolejka::pNext
```

Definicja w linii 31 pliku header.h.

3.2.2.2 pWiersz

```
wierzcholek* kolejka::pWiersz
```

Definicja w linii 30 pliku header.h.

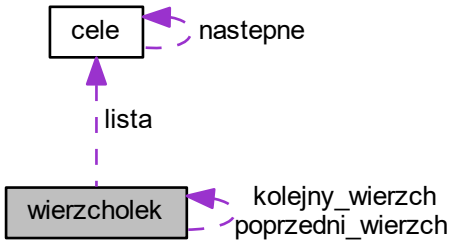
Dokumentacja dla tej struktury została wygenerowana z pliku:

- **header.h**

3.3 Dokumentacja struktury wierzcholek

```
#include <header.h>
```

Diagram współpracy dla wierzcholek:



Atrybuty publiczne

- std::string **nazwa**
- **wierzcholek** ∗ **kolejny_wierzch**
- **cele** ∗ **lista**
- double **odleglosc**
- **wierzcholek** ∗ **poprzedni_wierzch**

3.3.1 Opis szczegółowy

Struktura zawiera wierzcholki wraz z wskaznikami na cele i odleglosciami do nich

Definicja w linii 17 pliku header.h.

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 kolejny_wierzch

wierzcholek∗ wierzcholek::kolejny_wierzch

Definicja w linii 20 pliku header.h.

3.3.2.2 lista

cele∗ wierzcholek::lista

Definicja w linii 21 pliku header.h.

3.3.2.3 nazwa

std::string wierzcholek::nazwa

Definicja w linii 19 pliku header.h.

3.3.2.4 odleglosc

double wierzcholek::odleglosc

Definicja w linii 22 pliku header.h.

3.3.2.5 poprzedni_wierzch

wierzcholek∗ wierzcholek::poprzedni_wierzch

Definicja w linii 23 pliku header.h.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- **header.h**

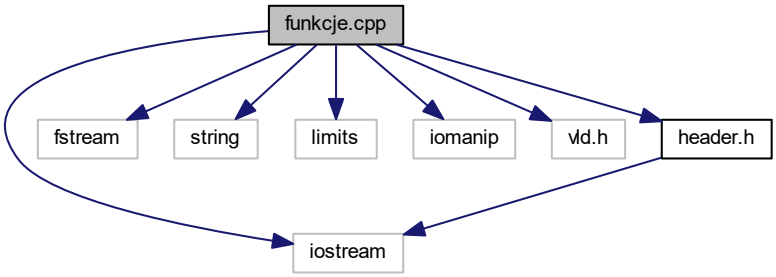
Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku funkcje.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <limits>
#include <iomanip>
#include "vld.h"
#include "header.h"
```

Wykres zależności załączania dla funkcje.cpp:



Funkcje

- void **wczytanie** (std::string nazwa_pliku, std::string plik, std::string wynik)
- void **wczytanieWynikow** (std::string plik, **wierzcholek** ∗&pHead, std::string wynik)
- void **dodajWierzcholek** (**wierzcholek** ∗&pHead, std::string nazwa_miasta, std::string nazwa_celu, double dystans)
- void **dodajCel** (**wierzcholek** ∗&w1, **wierzcholek** ∗&w2, double odleglosc)
- void **usuwanieCelu** (**cele** ∗pHead)
- void **usuwanieWierzcholka** (**wierzcholek** ∗&pHead)
- **wierzcholek** ∗ **szukaj** (**wierzcholek** ∗&pHead, std::string szukane_miasto)
- void **dijkstra** (**wierzcholek** ∗pHead, std::string m1, std::string m2)
- void **stworz_kolejke** (std::string W, **wierzcholek** ∗pHead, **kolejka** ∗&pWiersz)
- void **sortuj** (**kolejka** ∗&pHead)
- void **wypisywanie** (**wierzcholek** ∗wyp, std::string W, std::string W2, **wierzcholek** ∗pHead, std::ofstream &wypisz, int szerokosc)

4.1.1 Dokumentacja funkcji

4.1.1.1 dijkstra()

```
void dijkstra (
              wierzcholek * pHead,
             std::string m1,
             std::string m2 )
```

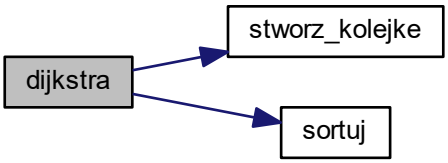
Implementacja algorytmu dijkstry

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
m1	nazwa miasta z ktorego wyjezdzamy
m2	nazwa miasta do ktorego szukamy drogi

Definicja w linii 161 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.2 dodajCel()

```
void dodajCel (
              wierzcholek *& w1,
              wierzcholek *& w2,
             double odleglosc )
```

Funkcja dodaje cel dla wierzcholka

Parametry

w1	wskaznik na wierzcholek do ktorego dodajemy cel
w2	wskaznik na cel ktory dodajemy do wierzcholka
odleglosc	dystans z wierzcholka do celu

Definicja w linii 114 pliku funkcje.cpp.

4.1.1.3 dodajWierzcholek()

```
void dodajWierzcholek (
              wierzcholek *& pHead,
             std::string nazwa_miasta,
             std::string nazwa_celu,
             double dystans )
```

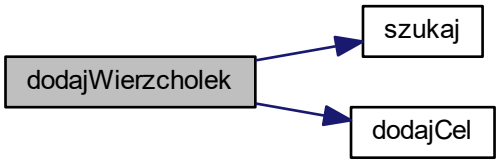
Funkcja dodaje wierzcholek do listy

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
nazwa_miasta	nazwa miasta z ktorego wyjezdzamy
nazwa_celu	nazwa miasta do ktorego mozna jechac

Definicja w linii 87 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.4 sortuj()

```
void sortuj (
              kolejka *& pHead )
```

Funkcja sortuje kolejke priorytetowa po wykonaniu obiegu algorytmu dijkstry i usunieciu jednego elementu tej kolejki

Parametry

pHead	wskaznik na pierwszy element kolejki priorytetowej

Definicja w linii 220 pliku funkcje.cpp.

4.1.1.5 stworz_kolejke()

```
void stworz_kolejke (
             std::string W,
              wierzcholek * pHead,
              kolejka *& pWiersz )
```

Funkcja tworzy kolejke priorytetowa

Parametry

W	miasto z ktorego szukamy drogi
pHead	wskaznik na pierwszy element listy wierzcholkow
pWiersz	wskaznik, za pomoca ktorego tworzymy kolejke priorytetowa

Definicja w linii 190 pliku funkcje.cpp.

4.1.1.6 szukaj()

```
wierzcholek* szukaj (
              wierzcholek *& pHead,
             std::string szukane_miasto )
```

Funkcja sprawdza czy dany wierzcholek jest juz w liscie

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
szukane_miasto	miasto, ktorego szukamy w liscie

Zwraca

zwraca wskaznik na znalezione miasto, jezeli nie znajdzie tego miasta- zwraca nullptr

Definicja w linii 141 pliku funkcje.cpp.

4.1.1.7 usuwanieCelu()

```
void usuwanieCelu (
              cele * pHead )
```

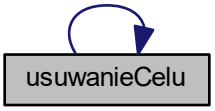
Funkcja usuwa cele

Parametry

pHead	wskaznik na pierwszy element listy celow

Definicja w linii 119 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.8 usuwanieWierzcholka()

```
void usuwanieWierzcholka (
              wierzcholek *& pHead )
```

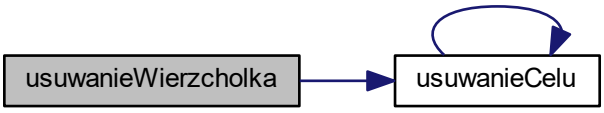
Funkcja usuwa wierzcholki

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow

Definicja w linii 130 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.9 wczytanie()

```
void wczytanie (
            std::string nazwa_pliku,
            std::string plik,
            std::string wynik )
```

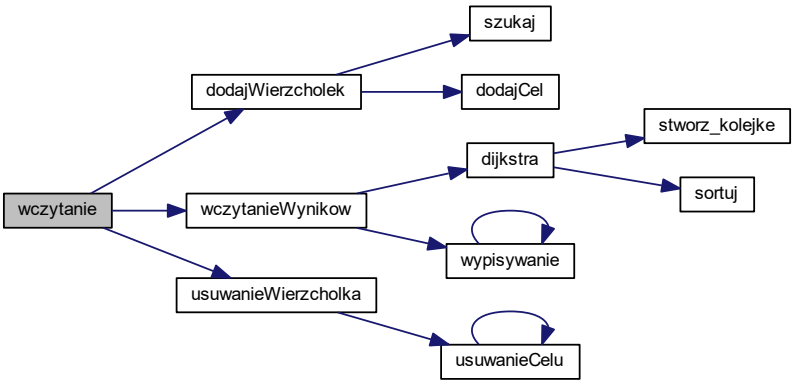
Funkcja wczytuje plik z drogami

Parametry

nazwa_pliku	plik z drogami
plik	plik z trasami do wyznaczenia
wynik	plik z wynikami

Definicja w linii 11 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.10 wczytanieWynikow()

```
void wczytanieWynikow (
            std::string plik,
             wierzcholek *& pHead,
            std::string wynik )
```

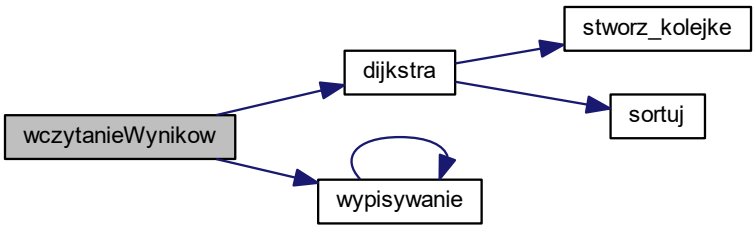
Funkcja wczytuje plik szukanych tras oraz przekazuje wyniki do odpowiedniego pliku

Parametry

W	plik szukanych tras
pHead	wskaznik na pierwszy element listy wierzcholkow
wynik	nazwa pliku do ktorego wypisujemy wyniki

Definicja w linii 26 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.1.1.11 wypisywanie()

```
void wypisywanie (
              wierzcholek * wyp,
             std::string W,
             std::string W2,
              wierzcholek * pHead,
             std::ofstream & wypisz,
             int szerokosc )
```

Funkcja wypisuje poszczegolne odcinki tras

Parametry

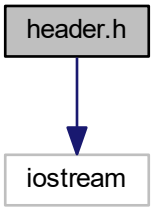
wyp	wskaznik pomocniczy sluzacy do wypisywania poszczegolnych odcinkow tras
W	miasto poczatkowe danego odcinka
W2	miasto docelowe danego odcinka
pHead	wskaznik na pierwszy element listy wierzcholkow
wypisz	strumien sluzacy do wypisania wynikow
szerokosc	argument wykorzystany do sformatowania szerokosci wypisania poszczegolnych odcinkow

Definicja w linii 279 pliku funkcje.cpp.

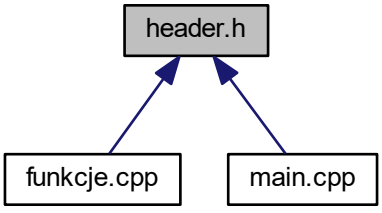
Oto graf wywołań dla tej funkcji:

4.2 Dokumentacja pliku header.h

`#include <iostream>`
Wykres zależności załączania dla header.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- struct **cele**
- struct **wierzcholek**
- struct **kolejka**

Funkcje

- void **wczytanie** (std::string nazwa_pliku, std::string plik, std::string wynik)
- void **dodajWierzcholek** (**wierzcholek** ∗&pHead, std::string nazwa_miasta, std::string nazwa_celu, double dystans)
- void **dodajCel** (**wierzcholek** ∗&w1, **wierzcholek** ∗&w2, double odleglosc)
- **wierzcholek** ∗ **szukaj** (**wierzcholek** ∗&pHead, std::string szukane_miasto)
- void **dijkstra** (**wierzcholek** ∗pHead, std::string m1, std::string m2)
- void **usuwanieCelu** (**cele** ∗pHead)
- void **usuwanieWierzcholka** (**wierzcholek** ∗&pHead)
- void **stworz_kolejke** (std::string W, **wierzcholek** ∗pHead, **kolejka** ∗&pWiersz)
- void **wczytanieWynikow** (std::string plik, **wierzcholek** ∗&pHead, std::string wynik)
- void **wypisywanie** (**wierzcholek** ∗wyp, std::string W, std::string W2, **wierzcholek** ∗pHead, std::ofstream &wypisz, int szerokosc)
- void **sortuj** (**kolejka** ∗&pHead)

4.2.1 Dokumentacja funkcji

4.2.1.1 dijkstra()

```
void dijkstra (
              wierzcholek * pHead,
             std::string m1,
             std::string m2 )
```

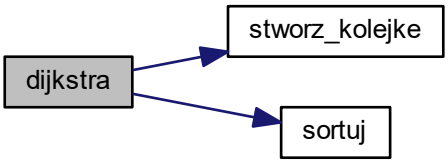
Implementacja algorytmu dijkstry

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
m1	nazwa miasta z ktorego wyjezdzamy
m2	nazwa miasta do ktorego szukamy drogi

Definicja w linii 161 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.2 dodajCel()

```
void dodajCel (
              wierzcholek *& w1,
              wierzcholek *& w2,
             double odleglosc )
```

Funkcja dodaje cel dla wierzcholka

Parametry

w1	wskaznik na wierzcholek do ktorego dodajemy cel
w2	wskaznik na cel ktory dodajemy do wierzcholka
odleglosc	dystans z wierzcholka do celu

Definicja w linii 114 pliku funkcje.cpp.

4.2.1.3 dodajWierzcholek()

```
void dodajWierzcholek (
              wierzcholek *& pHead,
             std::string nazwa_miasta,
             std::string nazwa_celu,
             double dystans )
```

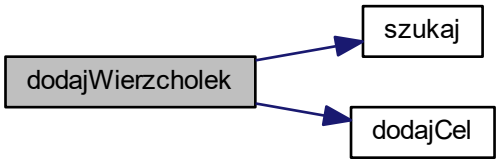
Funkcja dodaje wierzcholek do listy

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
nazwa_miasta	nazwa miasta z ktorego wyjezdzamy
nazwa_celu	nazwa miasta do ktorego mozna jechac

Definicja w linii 87 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.4 sortuj()

```
void sortuj (
              kolejka *& pHead )
```

Funkcja sortuje kolejke priorytetowa po wykonaniu obiegu algorytmu dijkstry i usunieciu jednego elementu tej kolejki

Parametry

pHead	wskaznik na pierwszy element kolejki priorytetowej

Definicja w linii 220 pliku funkcje.cpp.

4.2.1.5 stworz_kolejke()

```
void stworz_kolejke (
             std::string W,
              wierzcholek * pHead,
              kolejka *& pWiersz )
```

Funkcja tworzy kolejke priorytetowa

Parametry

W	miasto z ktorego szukamy drogi
pHead	wskaznik na pierwszy element listy wierzcholkow
pWiersz	wskaznik, za pomoca ktorego tworzymy kolejke priorytetowa

Definicja w linii 190 pliku funkcje.cpp.

4.2.1.6 szukaj()

```
wierzcholek* szukaj (
             wierzcholek *& pHead,
            std::string szukane_miasto )
```

Funkcja sprawdza czy dany wierzcholek jest juz w liscie

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow
szukane_miasto	miasto, ktorego szukamy w liscie

Zwraca

zwraca wskaznik na znalezione miasto, jezeli nie znajdzie tego miasta- zwraca nullptr

Definicja w linii 141 pliku funkcje.cpp.

4.2.1.7 usuwanieCelu()

```
void usuwanieCelu (
             cele * pHead )
```

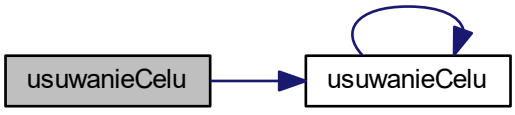
Funkcja usuwa cele

Parametry

pHead	wskaznik na pierwszy element listy celow

Definicja w linii 119 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.8 usuwanieWierzcholka()

```
void usuwanieWierzcholka (
              wierzcholek *& pHead )
```

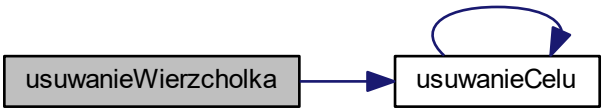
Funkcja usuwa wierzcholki

Parametry

pHead	wskaznik na pierwszy element listy wierzcholkow

Definicja w linii 130 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.9 wczytanie()

```
void wczytanie (
             std::string nazwa_pliku,
             std::string plik,
             std::string wynik )
```

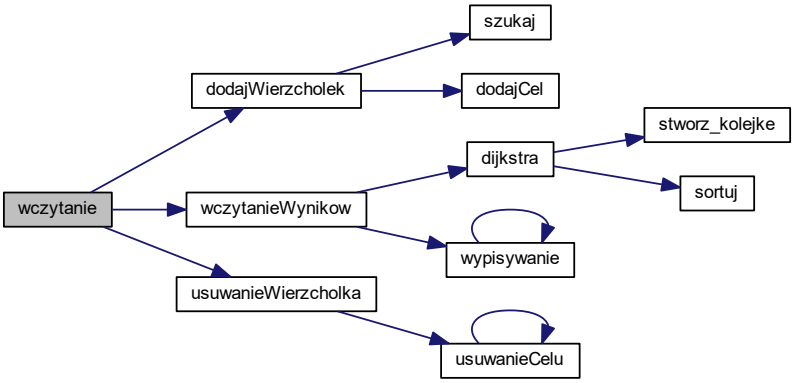
Funkcja wczytuje plik z drogami

Parametry

nazwa_pliku	plik z drogami
plik	plik z trasami do wyznaczenia
wynik	plik z wynikami

Definicja w linii 11 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.10 wczytanieWynikow()

```
void wczytanieWynikow (
            std::string plik,
             wierzcholek *& pHead,
            std::string wynik )
```

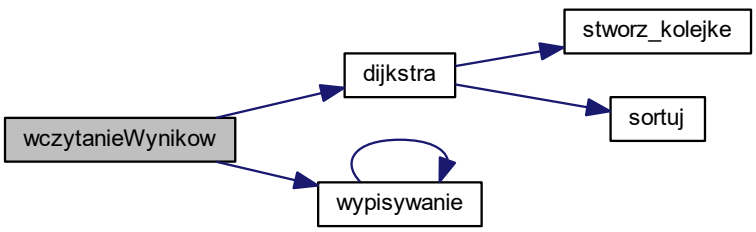
Funkcja wczytuje plik szukanych tras oraz przekazuje wyniki do odpowiedniego pliku

Parametry

W	plik szukanych tras
pHead	wskaznik na pierwszy element listy wierzcholkow
wynik	nazwa pliku do ktorego wypisujemy wyniki

Definicja w linii 26 pliku funkcje.cpp.

Oto graf wywołań dla tej funkcji:



4.2.1.11 wypisywanie()

```
void wypisywanie (
              wierzcholek * wyp,
             std::string W,
             std::string W2,
              wierzcholek * pHead,
             std::ofstream & wypisz,
             int szerokosc )
```

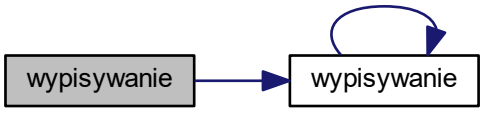
Funkcja wypisuje poszczegolne odcinki tras

Parametry

wyp	wskaznik pomocniczy sluzacy do wypisywania poszczegolnych odcinkow tras
W	miasto poczatkowe danego odcinka
W2	miasto docelowe danego odcinka
pHead	wskaznik na pierwszy element listy wierzcholkow
wypisz	strumien sluzacy do wypisania wynikow
szerokosc	argument wykorzystany do sformatowania szerokosci wypisania poszczegolnych odcinkow

Definicja w linii 279 pliku funkcje.cpp.

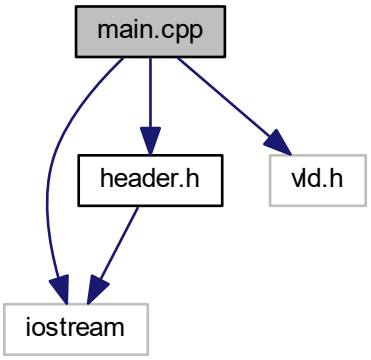
Oto graf wywołań dla tej funkcji:



4.3 Dokumentacja pliku main.cpp

```
#include <iostream>
#include "header.h"
#include "vld.h"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int **main** (int argc, char ∗argv[ ])

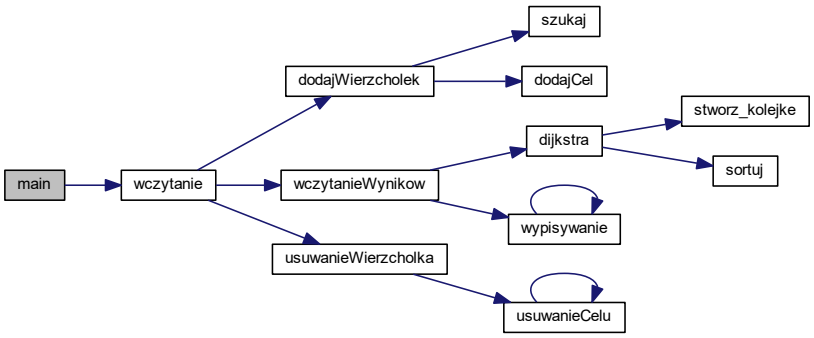
4.3.1 Dokumentacja funkcji

4.3.1.1 main()

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

Definicja w linii 5 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



Indeks