Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Spedycja

autor	Hanna Podeszwa
prowadzący	dr hab. inż. Krzysztof Simiński
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	środa, 13:45 – 15:15
sekcja	18
termin oddania sprawozdania	2019-01-13

1 Treść zadania

Napisać program wyszukujący najkrótsze możliwe trasy spedycyjne, wykorzystując algorytm Dijkstry. Miasta sąsiednie wraz z odległościami między nimi są zawarte w pliku tekstowym. Każda para miast jest w osobnej linii. Najkrótsze trasy spedycyjne zostaną zapisane do pliku, począwszy od miasta startowego, poprzez miasta sąsiednie (każda trasa w osobnej linii).

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

- `-i` plik wejściowy z sąsiednimi miastami i odległościami między nimi
- `-o` plik wyjściowy z trasami spedycyjnymi
- `-s` miasto startowe

2 Analiza zadania

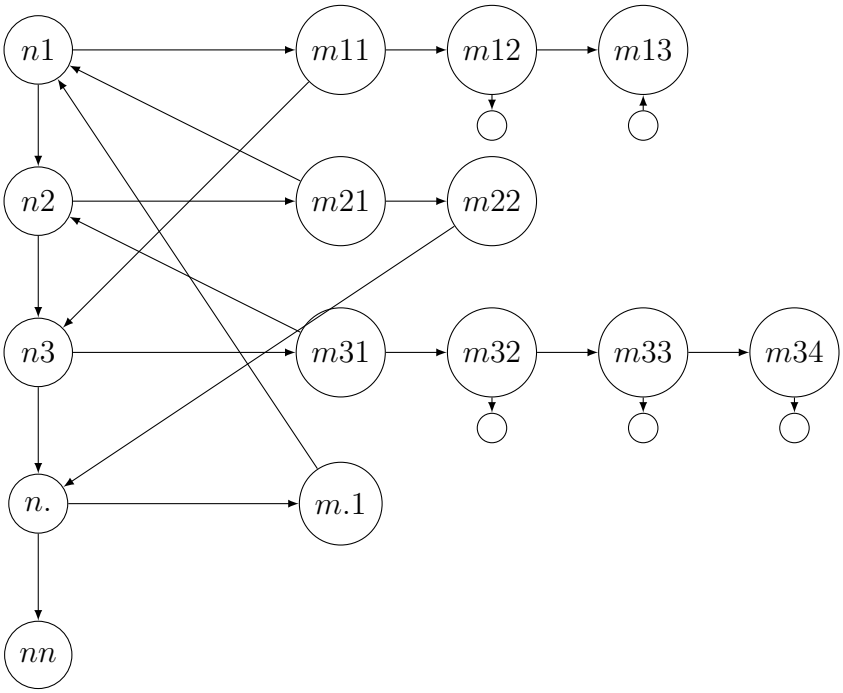
Zagadnienie przedstawia problem znalezienia najkrótszej możliwej trasy spedycyjnej pomiędzy miastem startowym, a wszystkimi innymi miastami zapisanymi w pliku.

2.1 Struktury danych

W programie wykorzystano listy jednokierunkowe do przechowywania miasta i dróg. Każdy element listy miast zawiera trzy wskaźniki: na nastepny element listy miast, na listę dróg oraz na miasto poprzednie, nazwę miasta, odległość od centrali oraz informację, czy dane miasto zostało już odwiedzone. Elementy listy dróg zawierają wskaźnik na odpowiednie miasto z listy miast oraz odległość między tym miastem, a miastem poprzednim. Rys. 1 przedstawia przykład list miast i dróg. Taka struktura danych umożliwia łatwe zastosowanie algorytmu Dijkstry.

2.2 Algorytmy

Program wyszukuje najkrótsze możliwe trasy spedycyjne, wykorzystując algorytm Dijkstry [1] [2]. Zaczynając zawsze od miasta położonego najbliżej miasta startowego, porównuje wartość sumy wartości zmiennej odleglosc_od_centrali poprzedniego miasta i odległości między tymi miastami z aktualną wartością zmiennej odleglosc_od_centrali miasta następnego. Kolejnym krokiem jest relaksacja odległości od centrali wraz z ewentualnym zapamiętaniem adresu miasta poprzedniego. Utworzenie list i ich przejście jest wykonywane w średnim czasie $O(n^2 + m)$ (gdzie n oznacza liczbę elementów listy miast, a m liczbę elementów list dróg).



Rysunek 1: Przykład listy miast i listy dróg.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń.

Należy przekazać do programu nazwy plików: wejściowego i wyjściowego oraz nazwę miasta startowego po odpowiednich przełącznikach (odpowiednio: `-i` dla pliku wejściowego, `-o` dla pliku wyjściowego i `-s` dla miasta startowego), np.

```
program -i miasta.txt -o trasy -s startowe
program -o trasy -s startowe -i miasta.txt
```

Pliki są plikami tekstowymi, ale mogą mieć dowolne rozszerzenie (lub go nie mieć.) Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez żadnego parametru lub z parametrem `-h`

```
program
program -h
```

powoduje wyświetlenie krótkiej pomocy. Uruchomienie programu z nieprawidłowymi parametrami powoduje wyświetlenie komunikatu

```
Bledne argumenty
```

i wyświetlenie pomocy.

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie odpowiedniego komunikatu:

```
Nie udalo sie otworzyc pliku.
```

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 sprawdzArgumenty. Funkcja ta sprawdza, czy program został wywołany w prawidłowy sposób. Gdy program nie został wywołany prawidłowo, zostaje wypisany stosowny komunikat i program się kończy. Następnie wywoływana jest funkcja wczytajzPliku. Funkcja ta otwiera plik wejściowy, sczytuje miasta oraz odległości między nimi i umieszcza je w odpowiednich listach jednokierunkowych. Po sczytaniu wszystkich miast funkcja zamyka plik. W razie wystąpienia błędu funkcja zwraca **false**, w przeciwnym wypadku **true**. Następnie wywoływana jest funkcja Dijkstra. Funkcja przechodzi przez listy i zapisuje odpowiednie wartosci w zmiennych odleglosc_od_centrali i pMiastoPoprzednie. Następnie wywoływana jest funkcja wypisz_wynik.Funkcja zapisuje trasy spedycyjne i odległości od centrali do pliku wyjściowego. Po zapisaniu tras funkcja zamyka plik. Ostatnią funkcją programu jest funkcja usun zwalniająca pamięć.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki niepoprawne (zawierające zbyt małą liczbę miast lub niezawierające odległości) powodują zgłoszenie błędu. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego (nie zostały podane żadne miasta ani odległości).

Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program, wyszukujący najkrótsze możliwe trasy spedycyjne, jest programem prostym, chociaż wymaga samodzielnego zarządzania pamięcią. Najbardziej wymagające okazało się zastosowanie algorytmy Dijkstry. Szczególnie trudne było zapewnienie wpisywania prawidłowych wartości do zmiennej odleglosc_od_centrali , tak by wypisane trasy spedycyjne były możliwie najkrótsze. Przygotowanie tego projektu pozwoliło mi lepiej zrozumieć sposób działania list jednokierunkowych oraz wskaźników.

Literatura

[1] Adam Drozdek. *C++. Algorytmy i struktury danych.* Helion, Gliwice, 2001.

[2] Kumarss Naimipour Richard Neapolitan. *Podstawy algorytmów z przykładami w C++.* Helion, Gliwice, 2004.

Dodatek
Szczegółowy opis typów i funkcji

Spedycja

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 plików z ich krótkimi opisami:

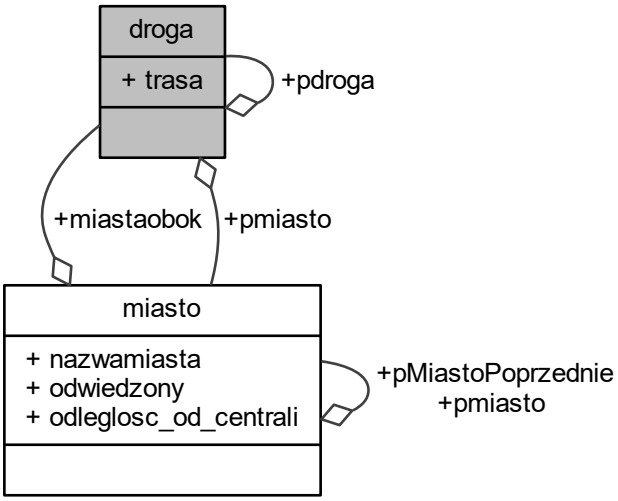
Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury droga

`#include <struktury.h>`

Diagram współpracy dla droga:



Atrybuty publiczne

- int trasa
- droga * pdroga
- miasto * pmiasto

3.1.1 Opis szczegółowy

Struktura droga

Parametry

trasa	odleglosc miedzy miastami
pdroga	wskaznik na nastepna droge
pmiasto	wskaznik na odpowiednie miasto

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 pdroga

```
droga* droga::pdroga
```

3.1.2.2 pmiasto

```
miasto* droga::pmiasto
```

3.1.2.3 trasa

```
int droga::trasa
```

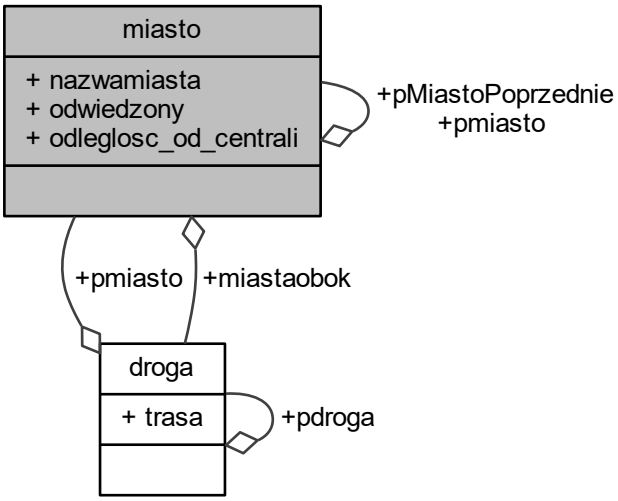
Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury.h

3.2 Dokumentacja struktury miasto

```
#include <struktury.h>
```

Diagram współpracy dla miasto:



Atrybuty publiczne

- std::string nazwamiasta
- miasto ∗ pmiasto
- droga ∗ miastaobok
- bool odwiedzony
- int odleglosc_od_centrali
- miasto ∗ pMiastoPoprzednie

3.2.1 Opis szczegółowy

Struktura miasto

Parametry

nazwamiasta	nazwa miasta
pmiasto	wskaznik na nastepne miasto
miastaobok	wskaznik na pierwszy element list drog
odwiedzony	true, gdy miasto zostalo odwiedzone, false – nie zostalo odwiedzone
odleglosc_od_centrali	odleglosc od centrali
pMiastoPoprzednie	wskaznik na miasto poprzednie

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 miastaobok

```
droga* miasto::miastaobok
```

3.2.2.2 nazwamiasta

```
std::string miasto::nazwamiasta
```

3.2.2.3 odleglosc_od_centrali

```
int miasto::odleglosc_od_centrali
```

3.2.2.4 odwiedzony

```
bool miasto::odwiedzony
```

3.2.2.5 pmiasto

```
miasto* miasto::pmiasto
```

3.2.2.6 pMiastoPoprzednie

```
miasto* miasto::pMiastoPoprzednie
```

Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury.h

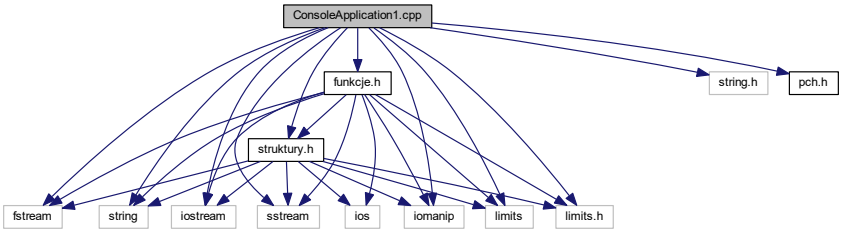
Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku ConsoleApplication1.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <iomanip>
#include <limits>
#include <limits.h>
#include "string.h"
#include "pch.h"
#include "struktury.h"
#include "funkcje.h"
```

Wykres zależności załączania dla ConsoleApplication1.cpp:



Definicje

- #define debug(x) std::cerr << "(" << __LINE__ << ") " << #x << " == " << (x) << std::endl;

Funkcje

- int main (int ile, char **params)

4.1.1 Dokumentacja definicji

4.1.1.1 debug

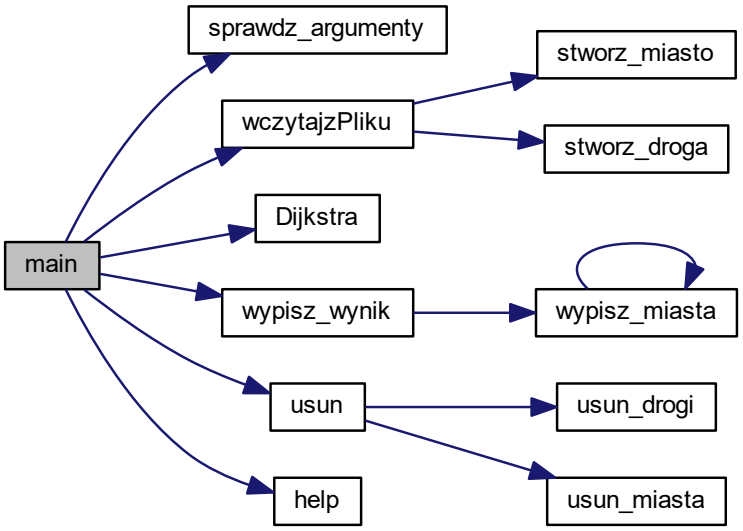
```
#define debug(
             x ) std::cerr << "(" << __LINE__ << ") " << #x << " == " << (x) << std↩
::endl;
```

4.1.2 Dokumentacja funkcji

4.1.2.1 main()

```
int main (
             int ile,
             char ** params )
```

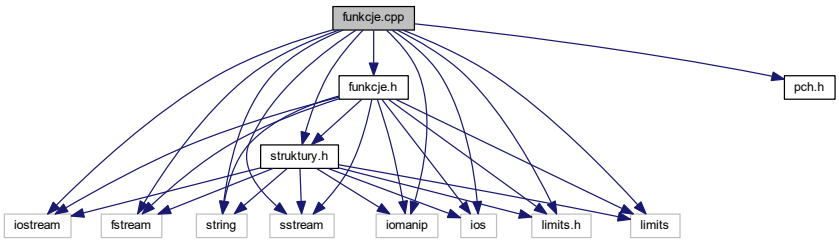
Oto graf wywołań dla tej funkcji:



4.2 Dokumentacja pliku funkcje.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <limits>
#include <iomanip>
#include <ios>
#include <limits.h>
#include "pch.h"
#include "struktury.h"
#include "funkcje.h"
```

Wykres zależności załączania dla funkcje.cpp:



Funkcje

- miasto ∗ stworz_miasto (miasto ∗&pHead_miasto, const std::string &nowanazwa)
- void stworz_droga (int kilometry, miasto ∗&nowe_miasto1, miasto ∗&nowe_miasto2)
- void wypisz_droga (droga ∗pHead_droga)
- void wypisz_miasto (miasto ∗pHead)
- bool Dijkstra (const std::string &startowy, miasto ∗pHead)
- void wypisz_miasta (miasto ∗pHead, std::ostream &wyjscie)
- void wypisz_wynik (miasto ∗pHead, const std::string &wyjscie)
- bool sprawdz_argumenty (int ile, char ∗∗params, std::string &wejscie, std::string &wyjscie, std::string &start)
- void wczytajzPliku (const std::string &wejscie, miasto ∗&pGlowa)
- void usun_drogi (miasto ∗pmiasto)
- void usun_miasta (miasto ∗&pHead)
- void usun (miasto ∗glowa_miasta)
- void help (int ile, char ∗∗params)

4.2.1 Dokumentacja funkcji

4.2.1.1 Dijkstra()

```
bool Dijkstra (
            const std::string & startowy,
            miasto * pHead )
```

Funkcja, wykorzystujca algorytm Dijkstry do znalezienia najkrotszych drog z miasta startowego do pozostalych miast

Parametry

startowy	miasto, od ktorego beda rozpoczynac sie wszystkie trasy
pHead	wskaznik na pierwszy element listy

Zwraca

Funkcja zwraca true, gdy miasto startowe bylo w liscie i false, gdy nie bylo w liscie

Oto graf wywoływań tej funkcji:

4.2.1.2 help()

```
void help (
            int ile,
            char ** params )
```

Funkcja wyswietla pomoc

Parametry

ile	ilosc argumentow
params	tablica argumentow

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:

4.2.1.3 sprawdz_argumenty()

```
bool sprawdz_argumenty (
            int ile,
            char ** params,
            std::string & wejscie,
            std::string & wyjscie,
            std::string & start )
```

Funkcja sprawdza argumenty wywolania programu

Parametry

	ile	ilosc argumentow
	params	tablica argumentow
out	*wejscie*	plik wejsciowy
out	*wyjscie*	plik wyjsciowy
out	*start*	miasto startowe

Zwraca

Funkcja zwraca true, gdy podane argumenty byly poprawne i false, gdy byly bledne

Oto graf wywoływań tej funkcji:

4.2.1.4 stworz_droga()

```
void stworz_droga (
            int kilometry,
            miasto *& nowe_miasto1,
            miasto *& nowe_miasto2 )
```

Funkcja dodaje nowe elementy do listy drParametry *kilometry* odlegosc pomiedzy miastami

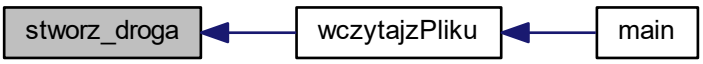
nowe_miasto1 wskaznik na miasto poczatkowe

nowe_miasto2 wskaznik na miasto docelowe

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.2.1.5 stworz_miasto()

```
miasto* stworz_miasto (
            miasto *& pHead,
            const std::string & nowanazwa )
```

Funkcja dodaje nowe miasto do listy

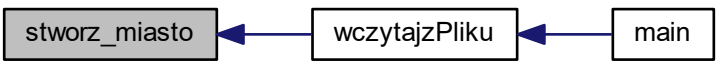
Parametry

pHead	wskaznik na pierwszy element listy
nowanazwa	nazwa miasta dodawanego do listy

Zwraca

Funkcja zwraca wskaznik na nowoutworzone miasto.

Oto graf wywoływań tej funkcji:



4.2.1.6 usun()

```
void usun (
            miasto * glowa_miasta )
```

Funkcja usuwa liste miast i drog

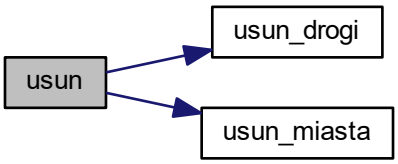
Parametry

glowa_miasta	wskaznik na pierwszy element listy miast

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

4.2.1.7 usun_drogi()

```
void usun_drogi (
            miasto * pmiasto )
```

Funkcja usuwa listy drog wszystkich miast

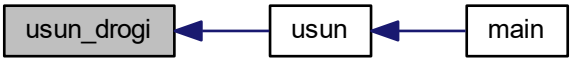
Parametry

pmiasto	wskaznik na kolejne miasto

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.2.1.8 usun_miasta()

```
void usun_miasta (
            miasto *& pHead )
```

Funkcja usuwa liste miast

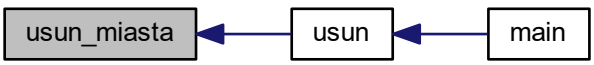
Parametry

in,out	*pHead*	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.2.1.9 wczytajzPliku()

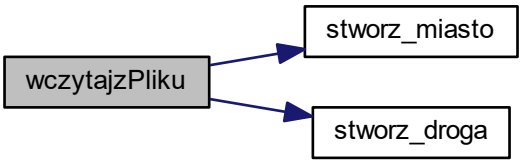
```
void wczytajzPliku (
            const std::string & wejscie,
            miasto *& pGlowa )
```

Funkcja wczytuje dane (miasta oraz odleglosci) z pliku

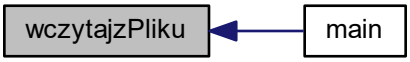
Parametry

	wejscie	nazwa pliku wejsciowego
in,out	*pGlowa*	wskaznik na pierwszy element listy

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.2.1.10 wypisz_droga()

```
void wypisz_droga (
            droga * pHead_droga )
```

Funkcja wypisuje liste drog

Parametry

pHead_droga	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:

4.2.1.11 wypisz_miasta()

```
void wypisz_miasta (
            miasto * pHead,
            std::ostream & wyjscie )
```

Funkcja wypisuje trase do wskazanego miasta

Parametry

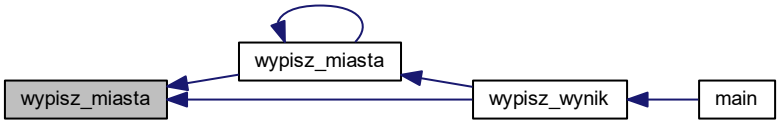
pHead	wskaznik na pierwszy element listy
wyjscie	nazwa pliku wyjsciowego

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:



4.2.1.12 wypisz_miasto()

```
void wypisz_miasto (
            miasto * pHead )
```

Funkcja wypisuje liste miast

Parametry

pHead	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:

4.2.1.13 wypisz_wynik()

```
void wypisz_wynik (
            miasto * pHead,
            const std::string & wyjscie )
```

Funkcja zapisuje wynik (wszystkie trasy do miast wraz z odleglosciami) do pliku wyjsciowego

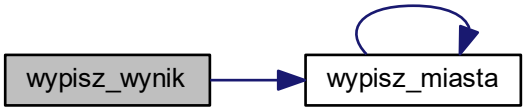
Parametry

pHead	wskaznik na pierwszy element listy
wyjscie	nazwa pliku wyjsciowego

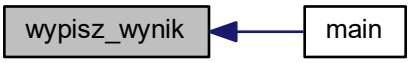
Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

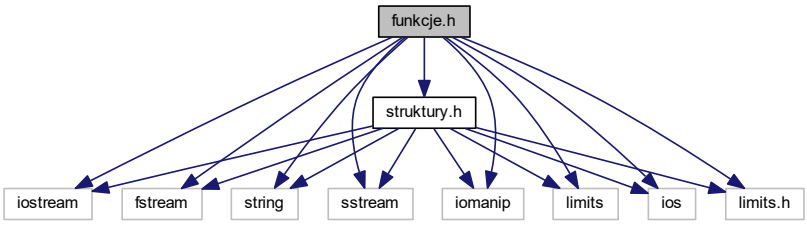


4.3 Dokumentacja pliku funkcje.h

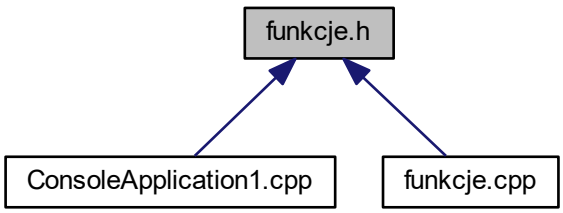
```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <iomanip>
#include <limits>
#include <ios>
#include <limits.h>
```

```
#include "struktury.h"
```

Wykres zależności załączania dla funkcje.h:



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



Funkcje

- miasto ∗ stworz_miasto (miasto ∗&pHead, const std::string &nowanazwa)
- void stworz_droga (int kilometry, miasto ∗&nowe_miasto1, miasto ∗&nowe_miasto2)
- void wypisz_miasto (miasto ∗pHead)
- void wypisz_droga (droga ∗pHead_droga)
- bool Dijkstra (const std::string &startowy, miasto ∗pHead)
- void wypisz_wynik (miasto ∗pHead, const std::string &wyjscie)
- void wypisz_miasta (miasto ∗pHead, std::ostream &wyjscie)
- bool sprawdz_argumenty (int ile, char ∗∗params, std::string &wejscie, std::string &wyjscie, std::string &start)
- void wczytajzPliku (const std::string &wejscie, miasto ∗&pGlowa)
- void usun_drogi (miasto ∗pmiasto)
- void usun_miasta (miasto ∗&pHead)
- void usun (miasto ∗glowa_miasta)
- void help (int ile, char ∗∗params)

4.3.1 Dokumentacja funkcji

4.3.1.1 Dijkstra()

```
bool Dijkstra (
            const std::string & startowy,
            miasto * pHead )
```

Funkcja, wykorzystujca algorytm Dijkstry do znalezienia najkrotszych drog z miasta startowego do pozostalych miast

Parametry

startowy	miasto, od ktorego beda rozpoczynac sie wszystkie trasy
pHead	wskaznik na pierwszy element listy

Zwraca

Funkcja zwraca true, gdy miasto startowe bylo w liscie i false, gdy nie bylo w liscie

Oto graf wywoływań tej funkcji:

4.3.1.2 help()

```
void help (
            int ile,
            char ** params )
```

Funkcja wyswietla pomoc

Parametry

ile	ilosc argumentow
params	tablica argumentow

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:

4.3.1.3 sprawdz_argumenty()

```
bool sprawdz_argumenty (
             int ile,
             char ** params,
             std::string & wejscie,
             std::string & wyjscie,
             std::string & start )
```

Funkcja sprawdza argumenty wywolania programu

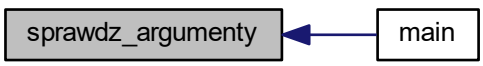
Parametry

	ile	ilosc argumentow
	params	tablica argumentow
out	*wejscie*	plik wejsciowy
out	*wyjscie*	plik wyjsciowy
out	*start*	miasto startowe

Zwraca

Funkcja zwraca true, gdy podane argumenty byly poprawne i false, gdy byly bledne

Oto graf wywoływań tej funkcji:



4.3.1.4 stworz_droga()

```
void stworz_droga (
            int kilometry,
            miasto *& nowe_miasto1,
            miasto *& nowe_miasto2 )
```

Funkcja dodaje nowe elementy do listy drParametry *kilometry* odlegosc pomiedzy miastami

---

nowe_miasto1 wskaznik na miasto poczatkowe

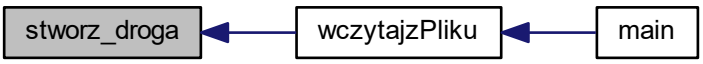
---

nowe_miasto2 wskaznik na miasto docelowe

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.3.1.5 stworz_miasto()

```
miasto* stworz_miasto (
            miasto *& pHead,
            const std::string & nowanazwa )
```

Funkcja dodaje nowe miasto do listy

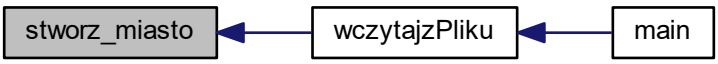
Parametry

pHead	wskaznik na pierwszy element listy
nowanazwa	nazwa miasta dodawanego do listy

Zwraca

Funkcja zwraca wskaznik na nowoutworzone miasto.

Oto graf wywoływań tej funkcji:



4.3.1.6 usun()

```
void usun (
            miasto * glowa_miasta )
```

Funkcja usuwa liste miast i drog

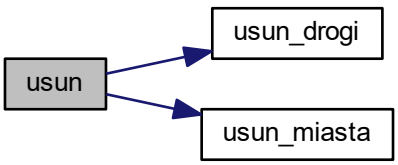
Parametry

glowa_miasta	wskaznik na pierwszy element listy miast

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

4.3.1.7 usun_drogi()

```
void usun_drogi (
            miasto * pmiasto )
```

Funkcja usuwa listy drog wszystkich miast

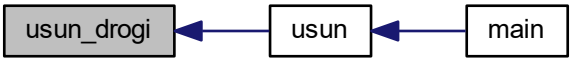
Parametry

pmiasto	wskaznik na kolejne miasto

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.3.1.8 usun_miasta()

```
void usun_miasta (
            miasto *& pHead )
```

Funkcja usuwa liste miast

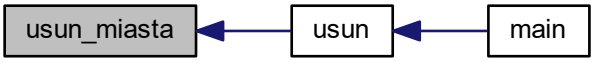
Parametry

in,out	*pHead*	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:



4.3.1.9 wczytajzPliku()

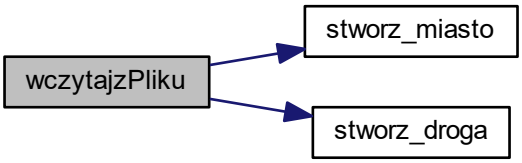
```
void wczytajzPliku (
            const std::string & wejscie,
            miasto *& pGlowa )
```

Funkcja wczytuje dane (miasta oraz odleglosci) z pliku

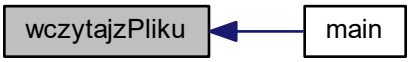
Parametry

	wejscie	nazwa pliku wejsciowego
in,out	*pGlowa*	wskaznik na pierwszy element listy

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.3.1.10 wypisz_droga()

```
void wypisz_droga (
            droga * pHead_droga )
```

Funkcja wypisuje liste drog

Parametry

pHead_droga	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywoływań tej funkcji:

4.3.1.11 wypisz_miasta()

```
void wypisz_miasta (
            miasto * pHead,
            std::ostream & wyjscie )
```

Funkcja wypisuje trase do wskazanego miasta

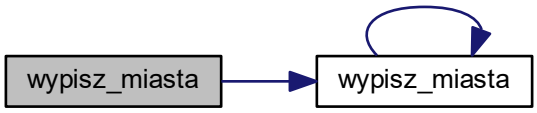
Parametry

pHead	wskaznik na pierwszy element listy
wyjscie	nazwa pliku wyjsciowego

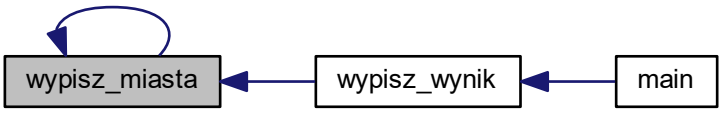
Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.3.1.12 wypisz_miasto()

```
void wypisz_miasto (
            miasto * pHead )
```

Funkcja wypisuje liste miast

Parametry

pHead	wskaznik na pierwszy element listy

Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:

4.3.1.13 wypisz_wynik()

```
void wypisz_wynik (
            miasto * pHead,
            const std::string & wyjscie )
```

Funkcja zapisuje wynik (wszystkie trasy do miast wraz z odleglosciami) do pliku wyjsciowego

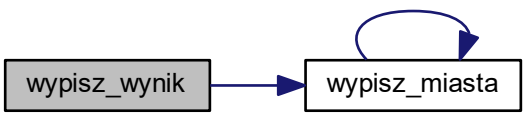
Parametry

pHead	wskaznik na pierwszy element listy
wyjscie	nazwa pliku wyjsciowego

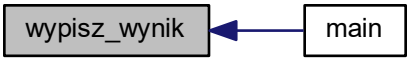
Zwraca

Funkcja nie zwraca niczego.

Oto graf wywołań dla tej funkcji:



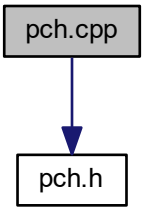
Oto graf wywoływań tej funkcji:



4.4 Dokumentacja pliku pch.cpp

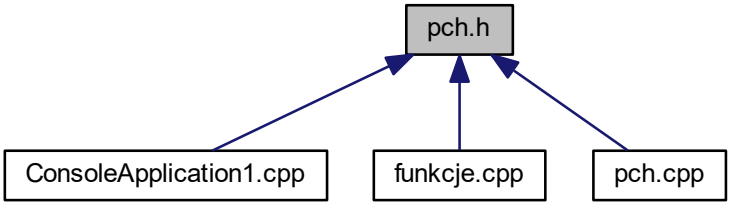
```
#include "pch.h"
```

Wykres zależności załączania dla pch.cpp:



4.5 Dokumentacja pliku pch.h

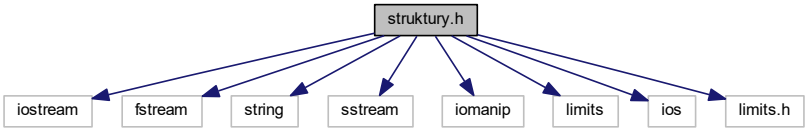
Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



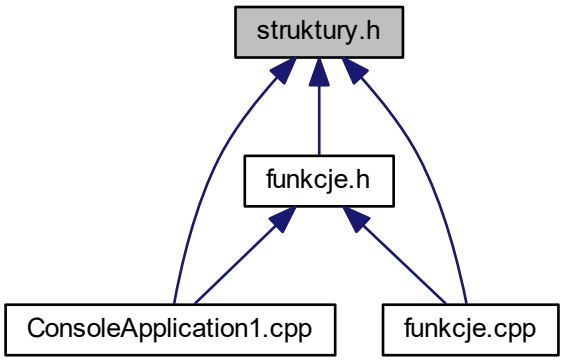
4.6 Dokumentacja pliku struktury.h

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <iomanip>
#include <limits>
#include <ios>
#include <limits.h>
```

Wykres zależności załączania dla struktury.h:



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



Komponenty

- struct miasto
- struct droga