Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Dwudzielny

autor	Tomasz Sojka
prowadzący	dr inż. Krzysztof Simiński
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium / ćwiczeń	poniedziałek, 08:30 – 10:00
grupa	2
sekcja	6
termin oddania sprawozdania	2018-01-25
data oddania sprawozdania	2018-01-25

1 Treść zadania

Napisac program, do sprawdzania, czy graf nieskierowany jest dwudzielny. Plik z grafem ma nastepujaca postac:
• Kazda krawedz jest podana w osobnej linii; podane sa dwa wierzchołki, które łaczy krawedz.
• W pliku moga wystapic puste linie.
• W linii moga wystapic dodatkowe (nadmiarowe) znaki białe.

Program wypisuje do pliku wyjsciowego zadany graf i komunikat, czy jest to graf dwudzielny, czy nie. Jezeli zadany graf jest dwudzielny, program wypisuje wierzchołki z obu grup. Program uruchamiany jest z linii polecen z wykorzystaniem nastepujacych przełaczników (kolejnosc przełaczników jest dowolna):

`-i` plik wejsciowy z krawedziami grafu
`-o` plik wyjsciowy z wynikami

2 Analiza zadania

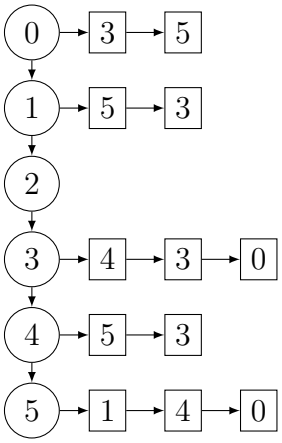
Zagadnienie przedstawia problem sprawdzenia dwudzielności grafu. Wymaga ono opracowania kodu, który odczyta krawędzi grafu z pliku wejściowego, sprawdzi czy graf ten jest dwudzielny i zapisze wynik(podany graf , wynik sprawdzania i dwie grupy wierzcholkow, jeli graf okazał się być dwudzielnym) do pliku wyjściowego.

2.1 Struktury danych

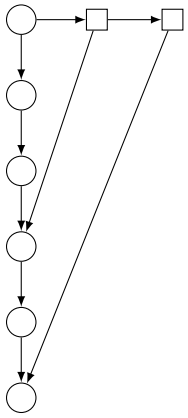
W programie wykorzystano listę jednokierunkową `element_listy`, w którą wpisano wierzcholki krawedzi sczytanych z pliku wejsciowego oraz dwie połączone struktury, `wierzcholek` i `krawedz`, dzięki którym udało się utworzyc listę sąsiedztwa (Rys 1), a także kolejkę potrzebną do jej przechodzenia. Lista jednokierunkowa pozwoliła na nieskomplikowane wczytanie danych z pliku wejciowego. Dzięki połączeniu dwóch struktur, do sprawdzania grafu nie było konieczne użycie osobnej tablicy dynamicznej z kolorami wierzchołków, czy też numerowanie tych wierzchołków(Patrz rys 2).

2.2 Algorytmy

Program sprawdza dwudzielność grafu poprzez kolorowanie jego wierzchołków na kontrastujące kolory (ustawianie wartości `kolor` ze struktury



Rysunek 1: Przykład listy sąsiedztwa przechowującej krawędzie grafu. Lista została utworzona z krawędzi sczytanych z pliku w kolejności:[(0,5), (0,3), (3,1), (4,3), (5,4), (1,5)]. Jest to jednak rysunek poglądowy i nie odzwierciedla prawdziwej budowy stworzonej w programie listy sąsiedztwa.



Rysunek 2: Przykład pokazuje prawdziwą budowę listy na podstawie wierzchołka nr 0 powyższego grafu i jego sąsiadów. Jak widać element listy głównej (pionowej, z wierzchołkami) zawiera: wskaźnik na następny element, na listę krawędzi(pozioma), , zaś element listy krawędzi zwieraj wskaźnik na następny element i na wierzchołek. (Wierzchołki zawierają również informacje do sprawdzania dwudzielność: `kolor` i `odwiedzony`, lecz nie są one istotne w budowie struktury.

`wierzcholek` na 1 lub -1). Po wybraniu początkowego nieodwiedzonego elementu , koloruje go na biało (wartość 1) i wszystkie wierzchołki sąsiadujące (końce krawędzi) na kolor odwrotny(tutaj wartość -1), teraz wierzchołki sąsiadujące zostają wierzchołkami głównymi i algorytm koloruje ich nieodwiedzonych sąsiadów na odwrotny kolor(tutaj wartość 1). Operacja się powtarza do momentu , aż wszystkie wierzchołki grafu zostaną sprawdzone lub do momentu gdy któryś odwiedzony sąsiad ma ten sam kolor co wierzchołek główny(aktualnie rozpartywany). Pierwsza opcja oznacza, że rozpatrywany graf posiada cechę dwudzielności, druga, że nie posiada tej cechy.

3 Specyfikacja zewnętrzna

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

```
program -i graf.txt -o graf_sprawdzony
program -o graf_sprawdzony -i graf.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 błędnymi parametrami

```
program
program -k graf.txt -m graf_sprawdzony
```

powoduje wyświetlenie komunikatu bledu krótkiej pomocy.

Podanie nieprawidłowej nazwy pliku, jego rozszerzenia, lub podanie nazwy pustego pliku powoduje wyświetlenie odpowiedniego komunikatu:

```
Brak pliku, niepoprawne rozszerzenie lub plik pusty !
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. Program rozdzielono na cztery pliki źródłowe (`.cpp`):
`main.cpp`, `pliki_i_przelaczniki.cpp`,
`lista_i_kolejka.cpp`, `sprawdzanie_dwudzielnosci.cpp`.
Nagłówki umieszczono w plikach nagłówkowych (`.h`):
`struktury_i_listy.h`, `pliki_i_przelacznikiy.h`.

Specyfikacja wewnętrzna znajduje się na końcu pliku (Patrz str. 5))

5 Testowanie

Program został przetestowany na krawedziach o różnych wartosciach . Wierzcholki mogą być numerowane dowolnymi liczbami całkowitymi i nie muszą być numerowane w kolejność (od zera, rosnąco co 1). Znaki białe w pliku z grafem są pomijane, a puste linie lub zawierające tylko spacje, czy tabulacje nie powodują żadnych błędów w działaniu programu. Nie podanie parametru lub podanie błędnego powoduje zgłoszenie błędu, wypisanie krótkiej instrukcji i zatrzymanie wykonywania programu. Odwołanie się do nieistniejącego pliku wejściowego, lub pliku pustego powoduje wyświelenie stosownego komunikatu i zakończenie programu.

6 Wnioski

Program Dwudzielny choć z początku wydawał się być programem nie do przejścia, po dogłębnym przeanalizowaniu okazuje się być w zasięgu nawet początkującego programisty. Samodzielne tworzenie kodu pod presją czasu pozwoliło na rozszerzenie wiedzy na temat języka c++ i przede wszystkim na przećwiczenie dynamicznych struktur danych. Największym problemem podczas wykonywania zadania było utworzenie struktury listy sąsiedztwa, funkcji sprawdzającej dwudzielność grafu, oraz ograniczenie czasowe na wykonanie programu.

Projekt2

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:

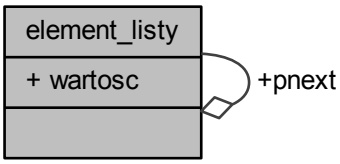
Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury element_listy

`#include <struktury_i_listy.h>`

Diagram współpracy dla element_listy:



Atrybuty publiczne

- int wartosc
- element_listy * pnext

3.1.1 Opis szczegółowy

struktura uzyta do przechowania krawedzi z pliku i jako kolejka do przeszukiwania listy sasiedztwa

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 pnext

```
element_listy* element_listy::pnext
```

wskaznik na nastepny wierzcholek

3.1.2.2 wartosc

```
int element_listy::wartosc
```

wartosc wierzcholka

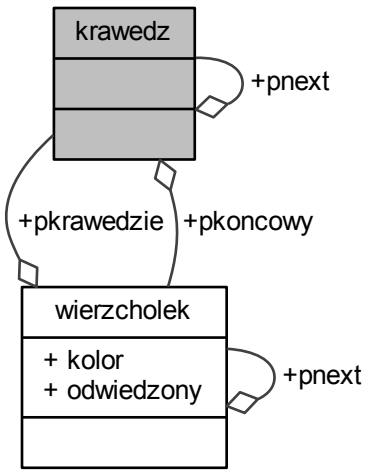
Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury_i_listy.h

3.2 Dokumentacja struktury krawedz

```
#include <struktury_i_listy.h>
```

Diagram współpracy dla krawedz:



Atrybuty publiczne

- wierzcholek * pkoncowy
- krawedz * pnext

3.2.1 Opis szczegółowy

struktura reprezentujaca w liscie sasiedztwa polaczenia wierzcholkow grafu z wierzcholkami koncowymi krawedzi

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 pkoncowy

```
wierzcholek* krawedz::pkoncowy
```

wierzcholek koncowy krawedzi

3.2.2.2 pnext

```
krawedz* krawedz::pnext
```

nastepna krawedz w liscie krawedzi

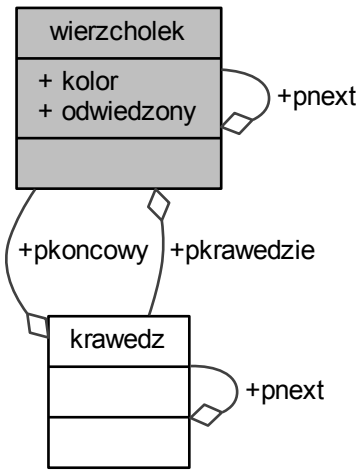
Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury_i_listy.h

3.3 Dokumentacja struktury wierzcholek

```
#include <struktury_i_listy.h>
```

Diagram współpracy dla wierzcholek:



Atrybuty publiczne

- int kolor = 0
- bool odwiedzony = false
- wierzcholek * pnext
- krawedz * pkrawedzie

3.3.1 Opis szczegółowy

struktura reprezentuje wierzcholki grafu w liscie sasiedztwa

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 kolor

```
int wierzcholek::kolor = 0
```

wartosc do sprawdzania dwudzielnosci, na poczatku wierzcholki sa niezakolorowane (wartosc jest rowna 0)

3.3.2.2 odwiedzony

```
bool wierzcholek::odwiedzony = false
```

wartosc do sprawdzania dwudzielnosci, na poczatku wszystkie nieodwiedzone

3.3.2.3 pkrawedzie

```
krawedz* wierzcholek::pkrawedzie
```

wskaznik na pierwszy element listy krawedzi , ktorych poczatkiem jest ten wierzcholek

3.3.2.4 pnext

```
wierzcholek* wierzcholek::pnext
```

kolejny wierzcholek w liscie wierzcholkow

Dokumentacja dla tej struktury została wygenerowana z pliku:

- struktury_i_listy.h

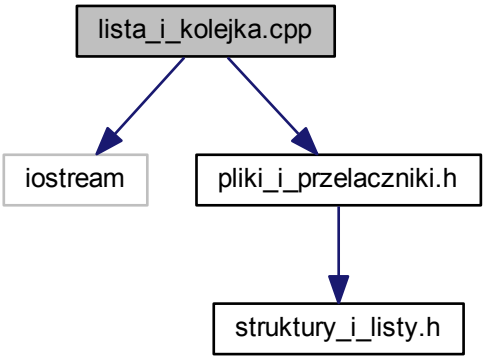
Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku lista_i_kolejka.cpp

```
#include <iostream>
#include "pliki_i_przelaczniki.h"
```

Wykres zależności załączania dla lista_i_kolejka.cpp:



Funkcje

- void tworzenie_listy_sasiedztwa (int liczba_wierz, int liczba_kraw, wierzcholek ∗&phead_listy_wierz, element_listy ∗phead_listy_kraw_z_pliku)
- void usuwanie_listy_krawedzi (krawedz ∗&phead_listy_kraw)
- void usuwanie_listy_sasiedztwa (wierzcholek ∗&phead_listy_wierz)
- void dodawanie_na_koniec_listy (element_listy ∗&phead_listy_kraw_z_pliku, int wartosc)
- void usuwanie_listy (element_listy ∗&phead_listy)
- void dodawanie_na_koniec_kolejki (wierzcholek ∗&phead_kolejki, wierzcholek ∗phead_listy_wierz)
- void usuwanie_z_poczatku_kolejki (wierzcholek ∗&phead_kolejki)
- wierzcholek ∗ zwracanie_poczatku_kolejki (wierzcholek ∗phead_kolejki)

4.1.1 Dokumentacja funkcji

4.1.1.1 dodawanie_na_koniec_kolejki()

```
void dodawanie_na_koniec_kolejki (
            wierzcholek *& phead_kolejki,
            wierzcholek * phead_listy_wierz )
```

Funkcja dodaje na koniec kolejki jeden element i przypisuje mu wszystkie dane(poza pnext) struktury wierzcholek, jakie posiada phead_listy_wierz

Parametry

phead_kolejki	glowa listy jednokierunkowej (kolejki), na ktora sa zapisywane wierzcholki
phead_listy_wierz	glowa listy wierzcholkow

Zwraca

nic

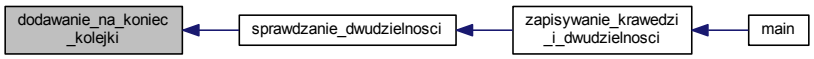
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.1.1.2 dodawanie_na_koniec_listy()

```
void dodawanie_na_koniec_listy (
            element_listy *& phead_listy_kraw_z_pliku,
            int wartosc )
```

Funkcja dodaje liczby calkowite(numery wierzcholkow) na koniec listy

Parametry

phead_listy_kraw_z_pliku	glowa listy jednokierunkowej, na ktora sa zapisywane wierzcholki
wartosc	numer dodawanego wierzcholka

Zwraca

nic

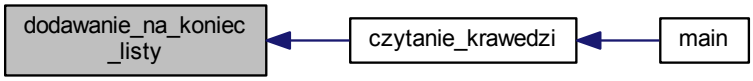
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.1.1.3 tworzenie_listy_sasiedztwa()

```
void tworzenie_listy_sasiedztwa (
             int liczba_wierz,
             int liczba_kraw,
             wierzcholek *& phead_listy_wierz,
             element_listy * phead_listy_kraw_z_pliku )
```

Funkcja tworzy liste sasiedztwa

Parametry

liczba_wierz	najwiekszy numer wieszcholka + 1 (moze istniec wierzcholek z numerem 0)
liczba_kraw	ilosc krawedzi grafu
phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa
phead_listy_kraw_z_pliku	glowa listy jednokierunkowej, z ktorej funkcja bierze krawedzi

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:

4.1.1.4 usuwanie_listy()

```
void usuwanie_listy (
            element_listy *& phead_listy )
```

Funkcja usuwa cala liste

Parametry

phead_listy	glowa listy jednokierunkowej, na ktorej sa zapisane wierzcholki

Zwraca

nic

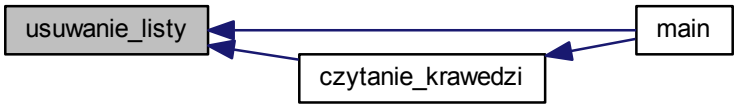
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.1.1.5 usuwanie_listy_krawedzi()

```
void usuwanie_listy_krawedzi (
            krawedz *& phead_listy_kraw )
```

Funkcja usuwa liste krawedzi podwieszona pod wierzcholek z listy sasiedztwa

Parametry

phead_listy_kraw	glowa listy krawedzi

Zwraca

nic

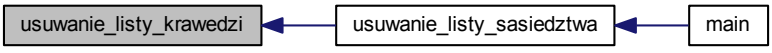
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.1.1.6 usuwanie_listy_sasiedztwa()

```
void usuwanie_listy_sasiedztwa (
            wierzcholek *& phead_listy_wierz )
```

Funkcja usuwa liste sasiedztwa

Parametry

phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.1.1.7 usuwanie_z_poczatku_kolejki()

```
void usuwanie_z_poczatku_kolejki (
            wierzcholek *& phead_kolejki )
```

Funkcja usuwa jeden element kolejki

Parametry

phead_kolejki	glowa listy jednokierunkowej(kolejki)

Zwraca

nic

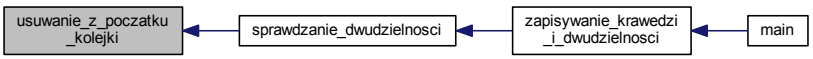
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.1.1.8 zwracanie_poczatku_kolejki()

```
wierzcholek* zwracanie_poczatku_kolejki (
            wierzcholek * phead_kolejki )
```

Funkcja zwraca adres jednego elementu z poczatku kolejki, z adresem zwrocone zostaja dane struktury wierzcholek tego elementu

Parametry

phead_kolejki	glowa listy jednokierunkowej(kolejki)

Zwraca

nic

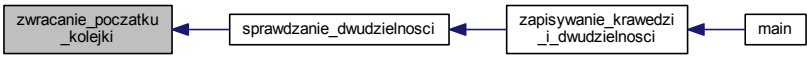
Data

25.01.2018

Autor

Tomasz Sojka

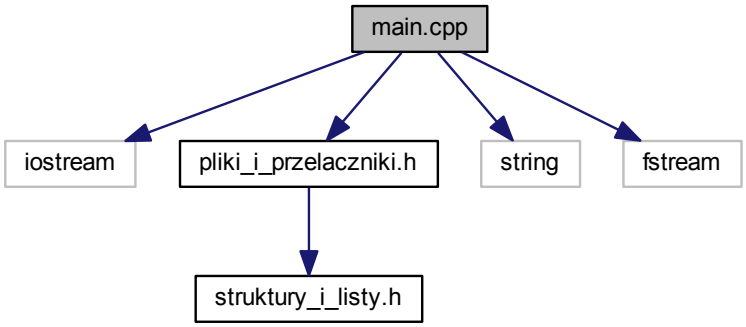
Oto graf wywoływań tej funkcji:



4.2 Dokumentacja pliku main.cpp

```
#include <iostream>
#include "pliki_i_przelaczniki.h"
#include <string>
#include <fstream>
```

Wykres zależności załączania dla main.cpp:



Funkcje

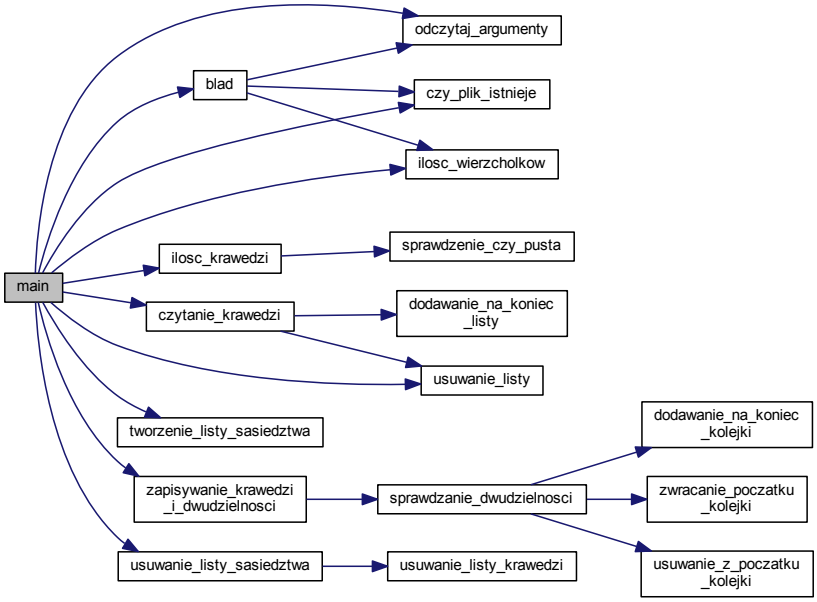
- int main (int argc, char ∗argv[ ])

4.2.1 Dokumentacja funkcji

4.2.1.1 main()

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

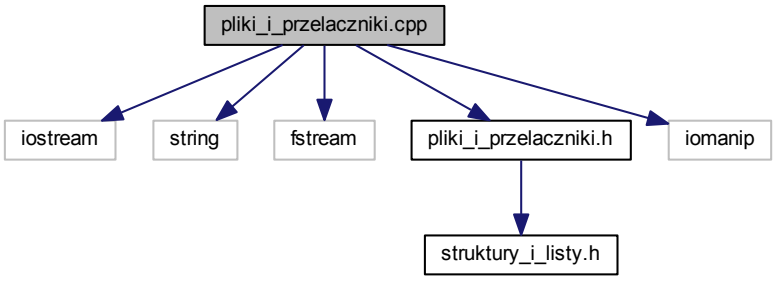
Oto graf wywołań dla tej funkcji:



4.3 Dokumentacja pliku pliki_i_przelaczniki.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include "pliki_i_przelaczniki.h"
#include <iomanip>
```

Wykres zależności załączania dla pliki_i_przelaczniki.cpp:



Funkcje

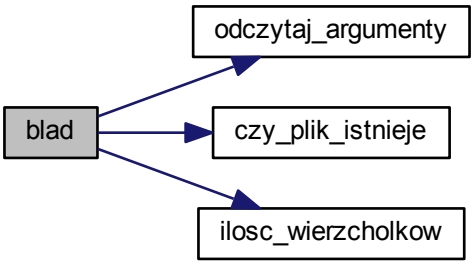
- bool odczytaj_argumenty (int ile, char ∗argumenty[ ], string &nazwa_pliku_wejsciowego, string &nazwa_↩ pliku_wyjsciowego)
- bool sprawdzenie_czy_pusta (const string &linia)
- int ilosc_wierzcholkow (const string &nazwa)
- int ilosc_krawedzi (const string &nazwa)
- void czytanie_krawedzi (const string &nazwa, element_listy ∗&phead_lista_kraw_z_pliku)
- void zapisywanie_krawedzi_i_dwudzielnosci (const string &nazwa, element_listy ∗phead_listy_kraw_z_pliku, wierzcholek ∗phead_lista_wierz, wierzcholek ∗phead_kolejki)
- bool czy_plik_istnieje (const string &nazwa)
- void blad (int ile, char ∗argumenty[ ], string &nazwa_pliku_wejsciowego, string &nazwa_pliku_wyjsciowego)

4.3.1 Dokumentacja funkcji

4.3.1.1 blad()

```
void blad (
            int ile,
            char * argumenty[],
            string & nazwa_pliku_wejsciowego,
            string & nazwa_pliku_wyjsciowego )
```

Oto graf wywołań dla tej funkcji:

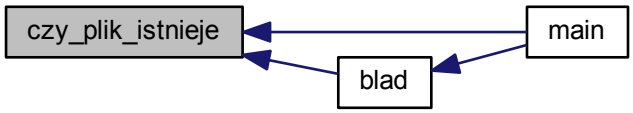


Oto graf wywoływań tej funkcji:

4.3.1.2 czy_plik_istnieje()

```
bool czy_plik_istnieje (
            const string & nazwa )
```

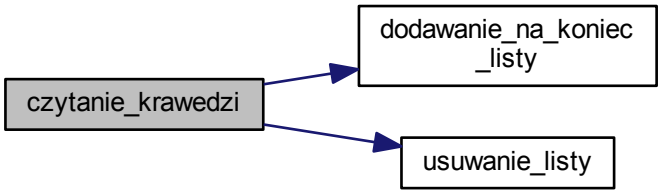
Oto graf wywoływań tej funkcji:



4.3.1.3 czytanie_krawedzi()

```
void czytanie_krawedzi (
            const string & nazwa,
            element_listy *& phead_lista_kraw_z_pliku )
```

Oto graf wywołań dla tej funkcji:

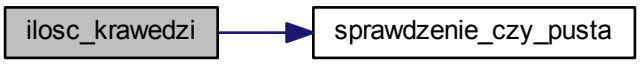


Oto graf wywoływań tej funkcji:

4.3.1.4 ilosc_krawedzi()

```
int ilosc_krawedzi (
            const string & nazwa )
```

Oto graf wywołań dla tej funkcji:

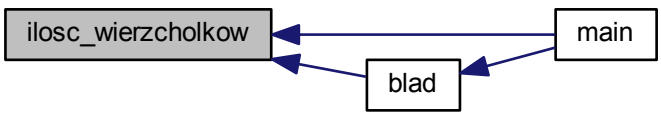


Oto graf wywoływań tej funkcji:

4.3.1.5 ilosc_wierzcholkow()

```
int ilosc_wierzcholkow (
            const string & nazwa )
```

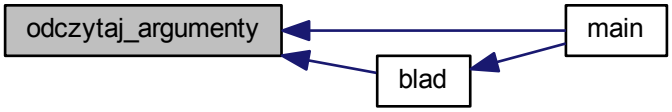
Oto graf wywoływań tej funkcji:



4.3.1.6 odczytaj_argumenty()

```
bool odczytaj_argumenty (
            int ile,
            char * argumenty[],
            string & nazwa_pliku_wejsciowego,
            string & nazwa_pliku_wyjsciowego )
```

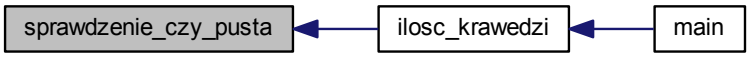
Oto graf wywoływań tej funkcji:



4.3.1.7 sprawdzenie_czy_pusta()

```
bool sprawdzenie_czy_pusta (
            const string & linia )
```

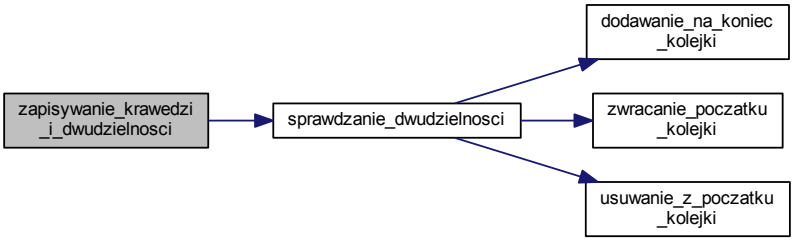
Oto graf wywoływań tej funkcji:



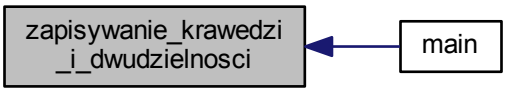
4.3.1.8 zapisywanie_krawedzi_i_dwudzielnosci()

```
void zapisywanie_krawedzi_i_dwudzielnosci (
            const string & nazwa,
            element_listy * phead_listy_kraw_z_pliku,
            wierzcholek * phead_lista_wierz,
            wierzcholek * phead_kolejki )
```

Oto graf wywołań dla tej funkcji:



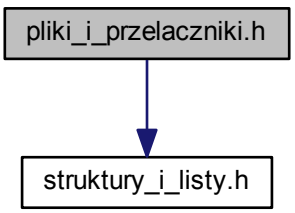
Oto graf wywoływań tej funkcji:



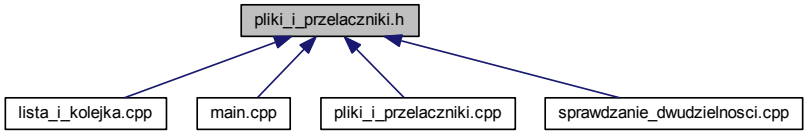
4.4 Dokumentacja pliku pliki_i_przelaczniki.h

```
#include "struktury_i_listy.h"
```

Wykres zależności załączania dla pliki_i_przelaczniki.h:



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



Funkcje

- bool odczytaj_argumenty (int ile, char ∗argumenty[ ], std::string &nazwa_pliku_wejsciowego, std::string &nazwa_pliku_wyjsciowego)
- int ilosc_wierzcholkow (const std::string &nazwa)
- int ilosc_krawedzi (const std::string &nazwa)
- void czytanie_krawedzi (const std::string &nazwa, element_listy ∗&phead_listy_kraw_z_pliku)
- void zapisywanie_krawedzi_i_dwudzielnosci (const std::string &nazwa, element_listy ∗phead_listy_kraw_z_pliku, wierzcholek ∗phead_listy_z_wierz, wierzcholek ∗phead_kolejki)
- bool sprawdzanie_dwudzielnosci (wierzcholek ∗phead_listy_wierz, wierzcholek ∗phead_kolejka)
- bool czy_plik_istnieje (const std::string &nazwa)
- void blad (int ile, char ∗argumenty[ ], std::string &nazwa_pliku_wejsciowego, std::string &nazwa_pliku_wyjsciowego)

4.4.1 Dokumentacja funkcji

4.4.1.1 blad()

```
void blad (
            int ile,
            char * argumenty[],
            std::string & nazwa_pliku_wejsciowego,
            std::string & nazwa_pliku_wyjsciowego )
```

Funkcja bledu, wyswietlajaca odpowiedni komunikat, w zaleznosci od bledu

Parametry

ile	ilosc potrzebnych parametrow
argumenty	tablica przechowujaca parametry
nazwa_pliku_wejsciowego	plik z grafem
nazwa_pliku_wyjsciwego	plik do zapisu

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.2 czy_plik_istnieje()

```
bool czy_plik_istnieje (
            const std::string & nazwa )
```

Funkcja sprawdza, czy podany plik istnieje

Parametry

nazwa	nazwa pliku z grafem

Zwraca

true jesli plik istnieje, false, gdy plik nie istnieje

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.3 czytanie_krawedzi()

```
void czytanie_krawedzi (
            const std::string & nazwa,
            element_listy *& phead_listy_kraw_z_pliku )
```

Funkcja sczytuje wierzcholki krawedzi zapisanych w pliku i zapisuje je na koniec listy jednokierunkowej, zeby pozniej wyciagac je od poczatku w dobrej kolejnosci

Parametry

nazwa	nazwa pliku z grafem
phead_kolejka	glowa listy jednokierunkowej

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.4 ilosc_krawedzi()

```
int ilosc_krawedzi (
            const std::string & nazwa )
```

Funkcja do zliczania ilosci krawedzi zapisanych w pliku, potrzebna przy tworzeniu listy sasiedztwa. Funkcja ignoruje puste linie, lub linie zawierajace tylko biale znaki

Parametry

nazwa	nazwa pliku z grafem

Zwraca

licznik - ilosc linii zawierajacych krawedzi

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.5 ilosc_wierzcholkow()

```
int ilosc_wierzcholkow (
            const std::string & nazwa )
```

Funkcja do wyszukiwania w pliku wierzcholka o najwyzszym numerze , potrzebna, aby lista wierzcholkow przy tworzeniu listy sasiedztwa byla odpowiedniej dlugosci

Parametry

nazwa	nazwa pliku z grafem

Zwraca

max+1, czyli maksymalny numer wierzcholka + 1 (poniewaz moze istniec wierzcholek 0)

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.6 odczytaj_argumenty()

```
bool odczytaj_argumenty (
            int ile,
            char * argumenty[],
            std::string & nazwa_pliku_wejsciowego,
            std::string & nazwa_pliku_wyjsciowego )
```

Funkcja do czytania parametrow.

Parametry

	ile	ilosc potrzebnych parametrow
	argumenty	tablica przechowujaca parametry
out	*nazwa_pliku_wejsciowego*	plik z ktorego czytam krawedzie
out	*nazwa_pliku_wyjsciowego*	plik do ktorego zapisuje sprawdzany graf, wynik przeszukiwania i jesli graf okaze sie byc dwudzielnym, dwie grupy wierzcholkow

Zwraca

true kiedy parametry zostaly wprowadzone poprawnie, false kiedy parametry nie zostaly wprowadzone lub zostaly wprowadzone niepoprawnie

Data

25.01.2018

Autor

Tomasz Sojka

4.4.1.7 sprawdzanie_dwudzielnosci()

```
bool sprawdzanie_dwudzielnosci (
            wierzcholek * phead_listy_wierz,
            wierzcholek * phead_kolejka )
```

Funkcja sprawdza dwudzielnosc grafu poprzez przechodzenie go i kolorowanie wierzcholkow (ustawianie wartosci kolor ze struktury wierzcholek na 1 lub -1)

Parametry

phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa
phead_kolejki	glowa kolejki do przechodzenia listy sasiedztwa

Zwraca

true kiedy udalo sie sprawdzic wszystkie wierzcholki i zadna krawedz nie ma wierzcholkow o tym samym kolorze, false, gdy sie nie udalo

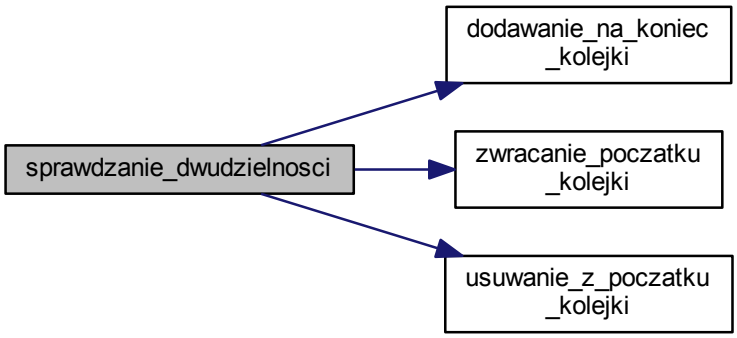
Data

25.01.2018

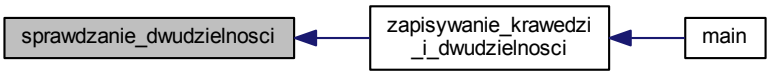
Autor

Tomasz Sojka

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.4.1.8 zapisywanie_krawedzi_i_dwudzielnosci()

```
void zapisywanie_krawedzi_i_dwudzielnosci (
            const std::string & nazwa,
            element_listy * phead_listy_kraw_z_pliku,
            wierzcholek * phead_listy_z_wierz,
            wierzcholek * phead_kolejki )
```

Funkcja zapisuje wynik do pliku (krawedzi grafu, wynik sprawdzania dwudzielnosci i jesli graf okazal sie byc dwudzielnym, dwie grupy wierzcholkow)

Parametry

nazwa	nazwa pliku do zapisu (jesli plik nie istnieje, zostanie utworzony)
phead_listy_kraw_z_pliku	glowa listy jednokierunkowej, z ktorej funkcja czyta krawedzi
phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa
phead_kolejki	glowa kolejki do przechodzenia listy sasiedztwa

Zwraca

nic

Data

25.01.2018

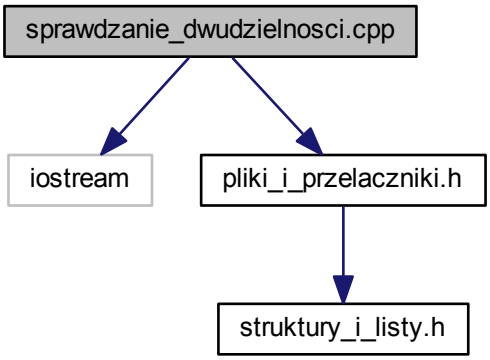
Autor

Tomasz Sojka

4.5 Dokumentacja pliku sprawdzanie_dwudzielnosci.cpp

```
#include <iostream>
#include "pliki_i_przelaczniki.h"
```

Wykres zależności załączania dla sprawdzanie_dwudzielnosci.cpp:



Funkcje

- bool sprawdzanie_dwudzielnosci (wierzcholek ∗phead_listy_wierz, wierzcholek ∗phead_kolejki)

4.5.1 Dokumentacja funkcji

4.5.1.1 sprawdzanie_dwudzielnosci()

```
bool sprawdzanie_dwudzielnosci (
            wierzcholek * phead_listy_wierz,
            wierzcholek * phead_kolejka )
```

Funkcja sprawdza dwudzielnosc grafu poprzez przechodzenie go i kolorowanie wierzcholkow (ustawianie wartosci kolor ze struktury wierzcholek na 1 lub -1)

Parametry

phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa
phead_kolejki	glowa kolejki do przechodzenia listy sasiedztwa

Zwraca

true kiedy udalo sie sprawdzic wszystkie wierzcholki i zadna krawedz nie ma wierzcholkow o tym samym kolorze, false, gdy sie nie udalo

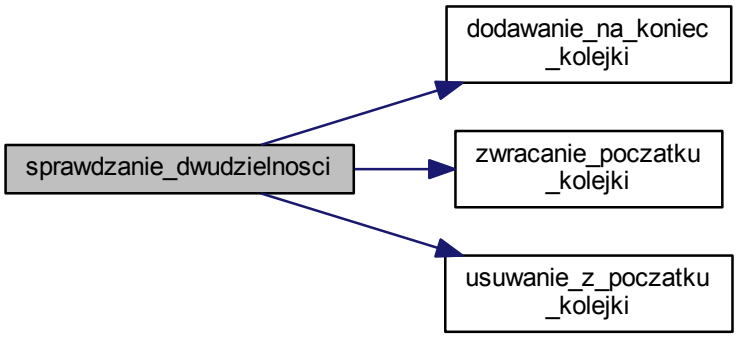
Data

25.01.2018

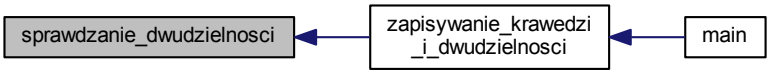
Autor

Tomasz Sojka

Oto graf wywołań dla tej funkcji:

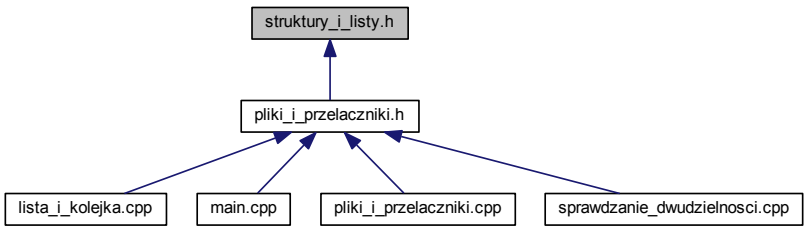


Oto graf wywoływań tej funkcji:



4.6 Dokumentacja pliku struktury_i_listy.h

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



Komponenty

- struct krawedz
- struct wierzcholek
- struct element_listy

Funkcje

- void tworzenie_listy_sasiedztwa (int liczba_wierz, int liczba_kraw, wierzcholek *&phead_listy_wierz, element_listy *phead_listy_kraw_z_pliku)
- void usuwanie_listy_krawedzi (krawedz *&phead_listy_kraw)
- void usuwanie_listy_sasiedztwa (wierzcholek *&phead_listy_wierz)
- void dodawanie_na_koniec_listy (element_listy *&phead_listy_kraw_z_pliku, int wartosc)
- void usuwanie_listy (element_listy *&phead_listy)
- void dodawanie_na_koniec_kolejki (wierzcholek *&phead_kolejki, wierzcholek *phead_listy_wierz)
- void usuwanie_z_poczatku_kolejki (wierzcholek *&phead_kolejki)
- wierzcholek * zwracanie_poczatku_kolejki (wierzcholek *phead_kolejki)

4.6.1 Dokumentacja funkcji

4.6.1.1 dodawanie_na_koniec_kolejki()

```
void dodawanie_na_koniec_kolejki (
            wierzcholek *& phead_kolejki,
            wierzcholek * phead_listy_wierz )
```

Funkcja dodaje na koniec kolejki jeden element i przypisuje mu wszystkie dane(poza pnext) struktury wierzcholek, jakie posiada phead_listy_wierz

Parametry

phead_kolejki	glowa listy jednokierunkowej (kolejki), na ktora sa zapisywane wierzcholki
phead_listy_wierz	glowa listy wierzcholkow

Zwraca

nic

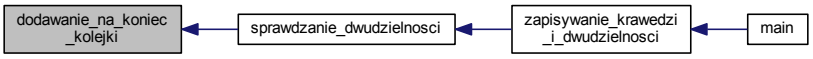
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.6.1.2 dodawanie_na_koniec_listy()

```
void dodawanie_na_koniec_listy (
            element_listy *& phead_listy_kraw_z_pliku,
            int wartosc )
```

Funkcja dodaje liczby calkowite(numery wierzcholkow) na koniec listy

Parametry

phead_listy_kraw_z_pliku	glowa listy jednokierunkowej, na ktora sa zapisywane wierzcholki
wartosc	numer dodawanego wierzcholka

Zwraca

nic

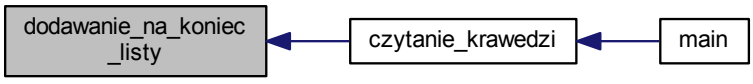
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.6.1.3 tworzenie_listy_sasiedztwa()

```
void tworzenie_listy_sasiedztwa (
            int liczba_wierz,
            int liczba_kraw,
            wierzcholek *& phead_listy_wierz,
            element_listy * phead_listy_kraw_z_pliku )
```

Funkcja tworzy liste sasiedztwa

Parametry

liczba_wierz	najwiekszy numer wieszcholka + 1 (moze istniec wierzcholek z numerem 0)
liczba_kraw	ilosc krawedzi grafu
phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa
phead_listy_kraw_z_pliku	glowa listy jednokierunkowej, z ktorej funkcja bierze krawedzi

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:

4.6.1.4 usuwanie_listy()

```
void usuwanie_listy (
            element_listy *& phead_listy )
```

Funkcja usuwa cala liste

Parametry

phead_listy	glowa listy jednokierunkowej, na ktorej sa zapisane wierzcholki

Zwraca

nic

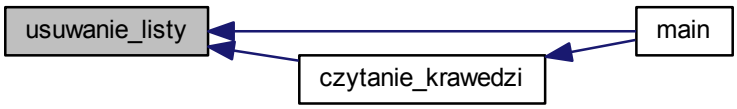
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.6.1.5 usuwanie_listy_krawedzi()

```
void usuwanie_listy_krawedzi (
             krawedz *& phead_listy_kraw )
```

Funkcja usuwa liste krawedzi podwieszona pod wierzcholek z listy sasiedztwa

Parametry

phead_listy_kraw	glowa listy krawedzi

Zwraca

nic

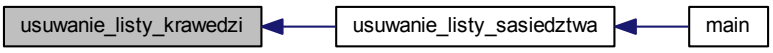
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.6.1.6 usuwanie_listy_sasiedztwa()

```
void usuwanie_listy_sasiedztwa (
            wierzcholek *& phead_listy_wierz )
```

Funkcja usuwa liste sasiedztwa

Parametry

phead_listy_wierz	glowa listy wierzcholkow tworzacej liste sasiedztwa

Zwraca

nic

Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.6.1.7 usuwanie_z_poczatku_kolejki()

```
void usuwanie_z_poczatku_kolejki (
            wierzcholek *& phead_kolejki )
```

Funkcja usuwa jeden element kolejki

Parametry

phead_kolejki	glowa listy jednokierunkowej(kolejki)

Zwraca

nic

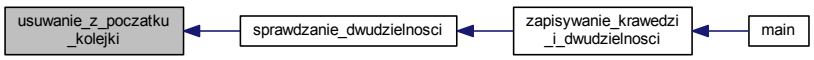
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



4.6.1.8 zwracanie_poczatku_kolejki()

```
wierzcholek* zwracanie_poczatku_kolejki (
            wierzcholek * phead_kolejki )
```

Funkcja zwraca adres jednego elementu z poczatku kolejki, z adresem zwrocone zostaja dane struktury wierzcholek tego elementu

Parametry

phead_kolejki	glowa listy jednokierunkowej(kolejki)

Zwraca

nic

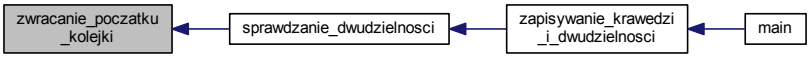
Data

25.01.2018

Autor

Tomasz Sojka

Oto graf wywoływań tej funkcji:



Skorowidz