PAMSI - pwilkosz

1.0

Wygenerowano przez Doxygen 1.8.6

Wt, 6 maj 2014 23:38:56

Spis treści

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy algorytm

Definicja klasy algorytm Jest to klasa bazowa, ktora ma za zadanie wczytac, przetworzyc i porownac dane z plikiem wzorcowym.

```
#include <algorytm.hh>
```

Diagram dziedziczenia dla algorytm

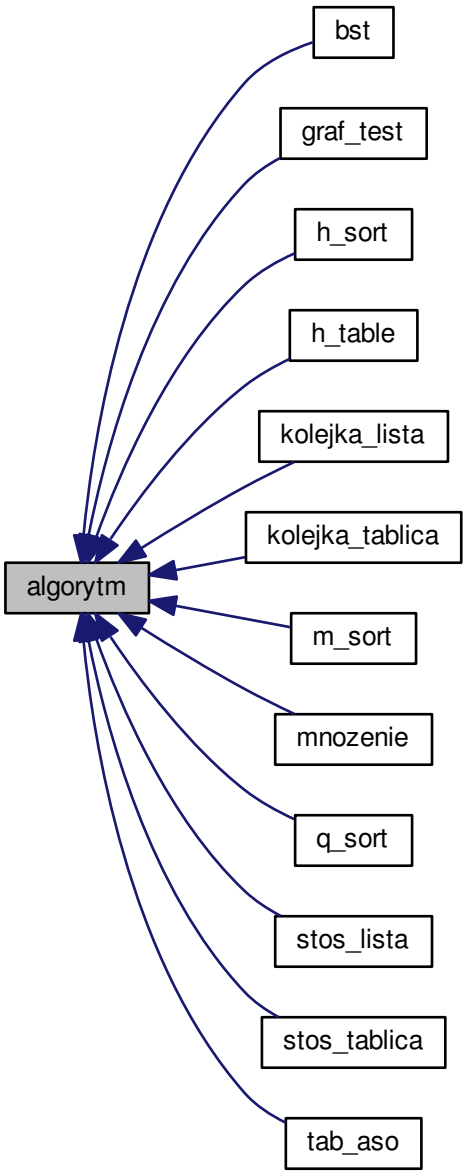
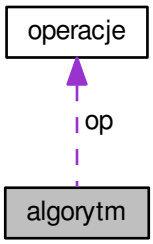


Diagram współpracy dla algorytm:



Metody publiczne

- algorytm (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor kopiujacy - przekazuje informacje o nazwach plikow, ktore zapisywane sa do pol klasy
- void wykonaj (ofstream &out)

  funkcja dokonuje operacji na pliku wejsciowym, wywoluje metody odpowiedzialne za pomiar czasu oraz za porownanie wyniku operacji z plikiem wzorcowym
- bool wczytaj (ifstream &plik)

  Metoda wczytuje plik wejsciowy do tablicy `dane` *oraz do obiektu* `op` *klasy* `operacje`.
- void set_N (int wart)

  metoda ustawia wartosc n
- bool wczytaj_wzor (ifstream &plik)

  Metoda wczytuje plik wzorcowy do tablicy `dane_wz`.
- virtual float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.
- bool porownaj ()

  porownuje przetworzony dane z danymi wzorcowymi
- int ile_danych ()
- float * jaki_czas ()
- void wlacz_zegar ()

  Metoda wlacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` *i przechowanie czasu w zmiennej* `start`.
- void wylacz_zegar ()

  Metoda wyacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` *i przechowanie czasu w zmiennej* `end`.
- void zapisz_do_csv (ofstream &out)

  Metoda zapisuje tablice `czas` *do pliku* `wyjscie.csv`.
- void zapisz_do_gnuplot (ofstream &out, float sr, float od)

  metoda zapisuje do pliku .csv parametry takie jak: srednia, ilosc liczb, odchylenie standardowe

Atrybuty publiczne

- float * czas

  zawiera wyniki dzialania algorytmu

Atrybuty chronione

- float * dane

  Tablica liczb wczytana z pliku.
- float * dane_wz

  tablica liczb zawartych w pliku wzorcowym
- int n

  ilosc danych w pliku
- int m

  ilosc powtorzen
- operacje op

  klasa zawierajaca tablice i metody do operacji na niej
- double czas1
- double czas2

4.1.1 Opis szczegółowy

Definicja klasy algorytm Jest to klasa bazowa, ktora ma za zadanie wczytac, przetworzyc i porownac dane z plikiem wzorcowym.

Definicja w linii 37 pliku algorytm.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 algorytm::algorytm (ifstream & *plik1*, ifstream & *plik2*, int *N*, int *M*) [inline]

konstruktor kopiujacy - przekazuje informacje o nazwach plikow, ktore zapisywane sa do pol klasy

Parametry

in	*plik1*	- plik wejsciowy
in	*plik2*	- plik wzorcowy
in	*N*	- ilosc danych wejsciowych
in	*M*	- ilosc powtorzen

Definicja w linii 80 pliku algorytm.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 int algorytm::ile_danych ( )

Zwraca

ilosc liczb wejsciowych

Definicja w linii 31 pliku algorytm.cpp.

4.1.3.2 float * algorytm::jaki_czas ( )

Zwraca

tablica `czas` z danymi pomiarowymi czasu wykonywania algorytmu

Definicja w linii 34 pliku algorytm.cpp.

4.1.3.3 bool algorytm::porownaj ( )

porownuje przetworzony dane z danymi wzorcowymi

Zwraca

true - gdy pliki zgodne false - w przeciwnym przypadku

Definicja w linii 99 pliku algorytm.cpp.

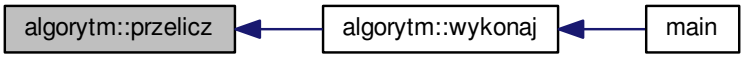
4.1.3.4 float algorytm::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana w graf_test, tab_aso, h_table, bst, m_sort, h_sort, q_sort, kolejka_lista, kolejka_tablica, stos_lista, stos_tablica i mnozenie.

Definicja w linii 9 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

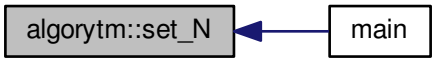


4.1.3.5 void algorytm::set_N (int *wart*) [inline]

metoda ustawia wartosc n

Definicja w linii 94 pliku algorytm.hh.

Oto graf wywoływań tej funkcji:



4.1.3.6 bool algorytm::wczytaj (ifstream & *plik*)

Metoda wczytuje plik wejsciowy do tablicy `dane` oraz do obiektu `op` klasy `operacje`.

Parametry

in	*plik*	- strumien pliku wejsciowego

Definicja w linii 10 pliku algorytm.cpp.

4.1.3.7 bool algorytm::wczytaj_wzor (ifstream & *plik*)

Metoda wczytuje plik wzorcowy do tablicy `dane_wz`.

Parametry

in	*plik*	- strumien pliku wejsciowego

Definicja w linii 21 pliku algorytm.cpp.

4.1.3.8 void algorytm::wlacz_zegar ( )

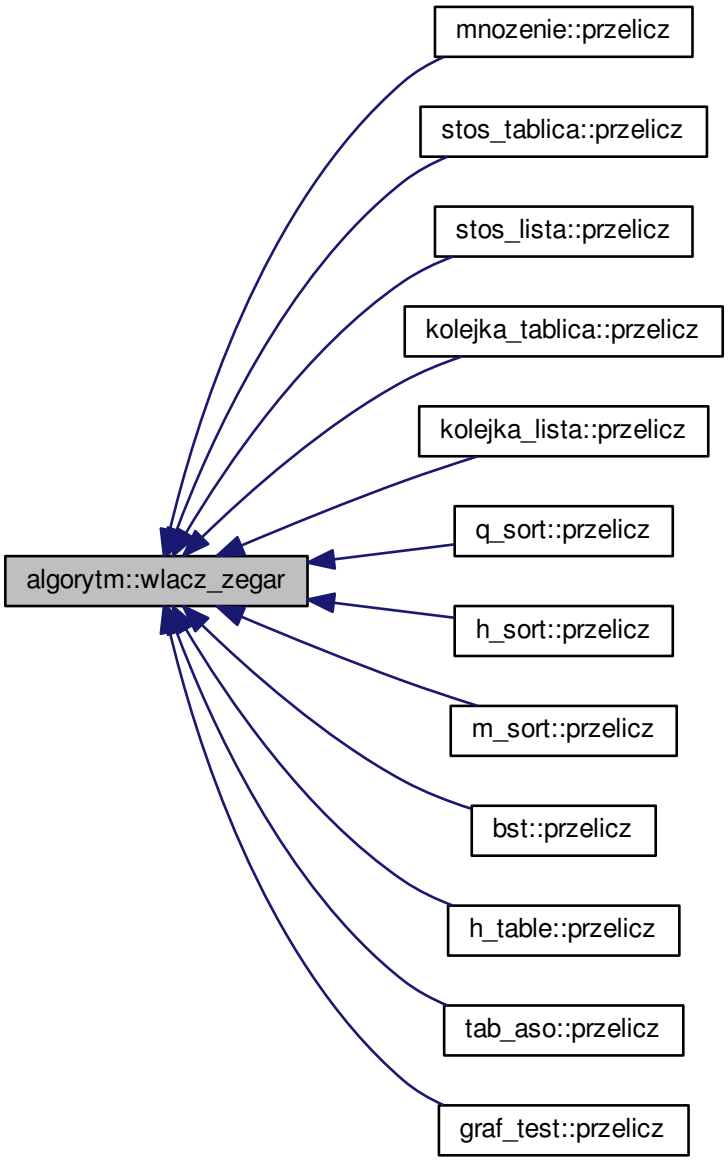
Metoda wlacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `start`.

Zwraca

start - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

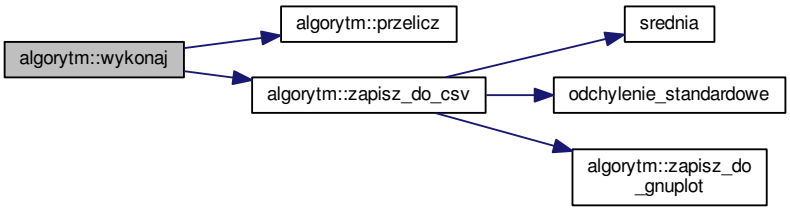


4.1.3.9 void algorytm::wykonaj (ofstream & *out*)

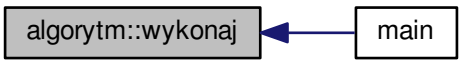
funkcja dokonuje operacji na pliku wejsciowym, wywoluje metody odpowiedzialne za pomiar czasu oraz za porownanie wyniku operacji z plikiem wzorcowym

Definicja w linii 78 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.1.3.10 void algorytm::wylacz_zegar ( )

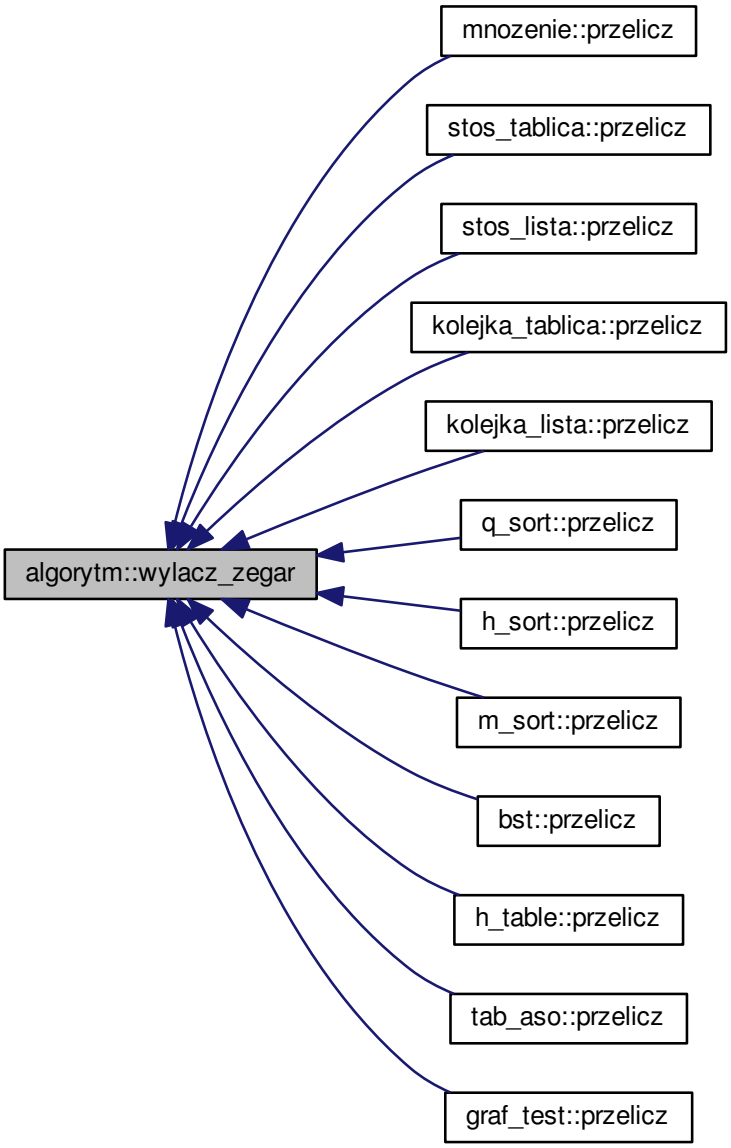
Metoda wyacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `end`.

Zwraca

end - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 49 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

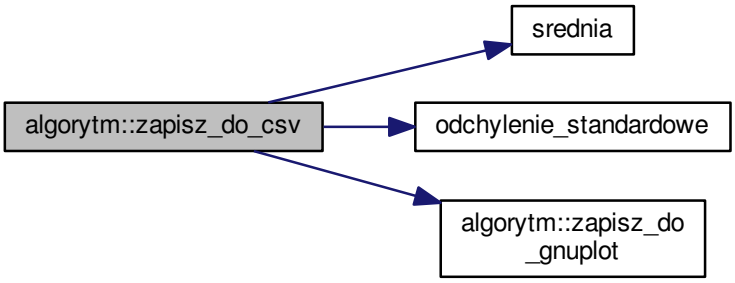


4.1.3.11 void algorytm::zapisz_do_csv (ofstream & *out*)

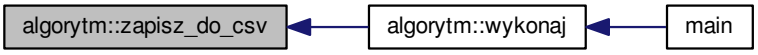
Metoda zapisuje tablice `czas` do pliku `wyjscie.csv`.

Definicja w linii 62 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

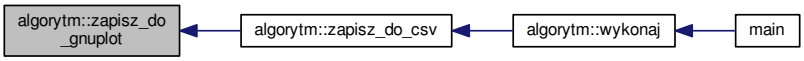


4.1.3.12 void algorytm::zapisz_do_gnuplot (ofstream & *out,* float *sr,* float *od*)

metoda zapisuje do pliku .csv parametry takie jak: srednia, ilosc liczb, odchylenie standardowe

Definicja w linii 106 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:



4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 float* algorytm::czas

zawiera wyniki dzialania algorytmu

Definicja w linii 70 pliku algorytm.hh.

4.1.4.2 double algorytm::czas1 [protected]

Definicja w linii 65 pliku algorytm.hh.

4.1.4.3 double algorytm::czas2 [protected]

Definicja w linii 65 pliku algorytm.hh.

4.1.4.4 float* algorytm::dane [protected]

Tablica liczb wczytana z pliku.

Definicja w linii 45 pliku algorytm.hh.

4.1.4.5 float* algorytm::dane_wz [protected]

tablica liczb zawartych w pliku wzorcowym

Definicja w linii 50 pliku algorytm.hh.

4.1.4.6 int algorytm::m [protected]

ilosc powtorzen

Definicja w linii 60 pliku algorytm.hh.

4.1.4.7 int algorytm::n [protected]

ilosc danych w pliku

Definicja w linii 56 pliku algorytm.hh.

4.1.4.8 operacje algorytm::op [protected]

klasa zawierajaca tablice i metody do operacji na niej

Definicja w linii 64 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.2 Dokumentacja klasy bst

Modeluje drzewo binarne przeznaczone do testowania szybkosci wyszukiwnaia.

`#include <algorytm.hh>`

Diagram dziedziczenia dla bst

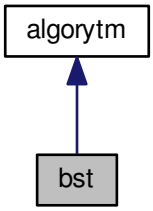
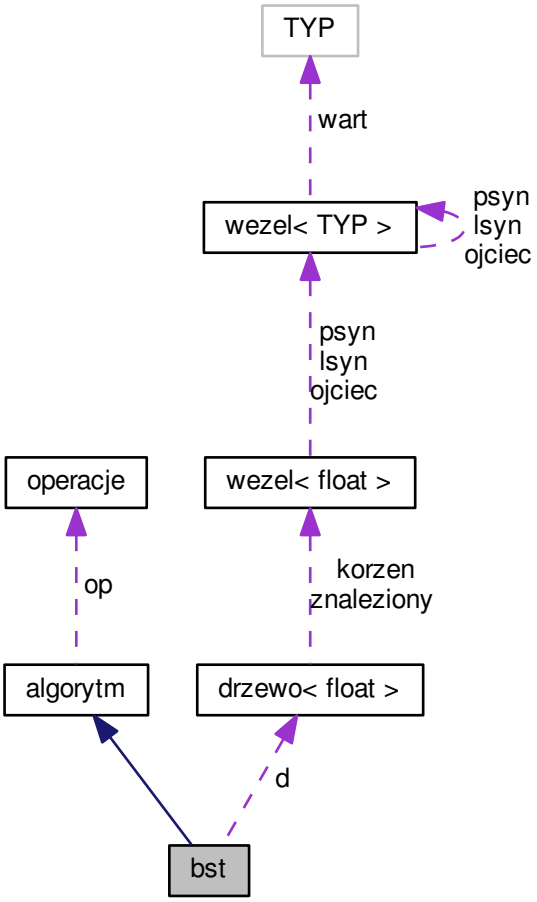


Diagram współpracy dla bst:



Metody publiczne

- void wczytaj_klucze (ifstream &plik)
- bst (ifstream &plik1, ifstream &plik2, ifstream &plik3, int N, int M)
- ~bst ()
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- drzewo< float > d
- string * klucze

Dodatkowe Dziedziczone Składowe

4.2.1 Opis szczegółowy

Modeluje drzewo binarne przeznaczone do testowania szybkosci wyszukiwnaia.

Definicja w linii 227 pliku algorytm.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 bst::bst (ifstream & *plik1,* ifstream & *plik2,* ifstream & *plik3,* int *N,* int *M*) [inline]

Definicja w linii 232 pliku algorytm.hh.

4.2.2.2 bst::~bst ( ) [inline]

Definicja w linii 240 pliku algorytm.hh.

4.2.3 Dokumentacja funkcji składowych

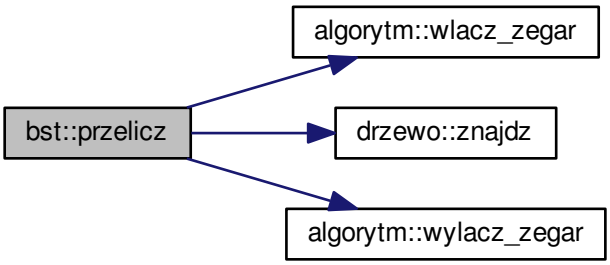
4.2.3.1 float bst::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 204 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.2.3.2 void bst::wczytaj_klucze (ifstream & *plik*)

Definicja w linii 199 pliku algorytm.cpp.

4.2.4 Dokumentacja atrybutów składowych

4.2.4.1 drzewo<float> bst::d [private]

Definicja w linii 228 pliku algorytm.hh.

4.2.4.2 string* bst::klucze [private]

Definicja w linii 229 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.3 Dokumentacja szablonu klasy drzewo< TYP >

modeluje binarne drzewo przeszukiwan

`#include <drzewo.hh>`

Metody publiczne

- drzewo ()

  konstruktor bezparametryczny
- drzewo (string k, TYP v)

  konstruktor parametryczny - przypisuje korzeniowi klucz i wartosc
- void dodaj_wezel (wezel< TYP > *W)

  dodaje wezel do drzewa

- void dodaj (string k, TYP v)

  dodaje wezel do drzewa
- bool znajdz (string k)

  szuka wezla o zadanym kluczu
- bool szukaj (string k, wezel< TYP > ∗w)

  sprawdza, czy w danym wezle znajduje sie szukany klucz
- void usun (string k)

  usuwa element o kluczu k, jezeli zostanie on znaleizony
- void czysc (wezel< TYP > ∗w)

  rekursywne czyszczenie wezla
- void wyczysc ()

  czysci cale drzewo

Atrybuty publiczne

- wezel< TYP > ∗ korzen

  korzen drzewa
- wezel< TYP > ∗ znaleziony

  znaleziony wezel w drzewie

4.3.1 Opis szczegółowy

template<typename TYP>class drzewo< TYP >

modeluje binarne drzewo przeszukiwan

Definicja w linii 70 pliku drzewo.hh.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 template<typename TYP> drzewo< TYP >::drzewo ( ) [inline]

konstruktor bezparametryczny

Definicja w linii 77 pliku drzewo.hh.

4.3.2.2 template<typename TYP> drzewo< TYP >::drzewo (string *k*, TYP *v*) [inline]

konstruktor parametryczny - przypisuje korzeniowi klucz i wartosc

Definicja w linii 79 pliku drzewo.hh.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 template<typename TYP> void drzewo< TYP >::czysc (wezel< TYP > ∗ *w*) [inline]

rekursywne czyszczenie wezla

Parametry

in	*w*	- czyszczony wezel

Definicja w linii 180 pliku drzewo.hh.

4.3.3.2 template<typename TYP> void drzewo< TYP >::dodaj (string *k*, TYP *v*) [inline]

dodaje wezel do drzewa

Parametry

in	*k*	- klucz wezla
in	*v*	= wartosc wezla

Definicja w linii 91 pliku drzewo.hh.

4.3.3.3 template<typename TYP> void drzewo< TYP >::dodaj_wezel (wezel< TYP > * *W*) [inline]

dodaje wezel do drzewa

Parametry

in	*W*	- utworzony uprzednio wezel

Definicja w linii 83 pliku drzewo.hh.

4.3.3.4 template<typename TYP> bool drzewo< TYP >::szukaj (string *k*, wezel< TYP > * *w*) [inline]

sprawdza, czy w danym wezle znajduje sie szukany klucz

Parametry

in	*k*	- klucz
in	*w*	- wezel, w ktorym sprawdzany jest klucz

Zwraca

true, gdy znaleziono, false w przeciwnym przypadku

Definicja w linii 109 pliku drzewo.hh.

4.3.3.5 template<typename TYP> void drzewo< TYP >::usun (string *k*) [inline]

usuwa element o kluczu k, jezeli zostanie on znaleizony

Parametry

in	*k*	- klucz wezla, ktory nalezy usunac

Definicja w linii 124 pliku drzewo.hh.

4.3.3.6 template<typename TYP> void drzewo< TYP >::wyczysc ( ) [inline]

czysci cale drzewo

Definicja w linii 188 pliku drzewo.hh.

4.3.3.7 template<typename TYP> bool drzewo< TYP >::znajdz (string *k*) [inline]

szuka wezla o zadanym kluczu

Parametry

in	*k*	- klucz

Zwraca

true, gdy znaleziono, w przeciwnym wypadku zwraca false

Definicja w linii 100 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:

4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 template<typename TYP> wezel<TYP>* drzewo< TYP >::korzen

korzen drzewa

Definicja w linii 73 pliku drzewo.hh.

4.3.4.2 template<typename TYP> wezel<TYP>* drzewo< TYP >::znaleziony

znaleziony wezel w drzewie

Definicja w linii 75 pliku drzewo.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

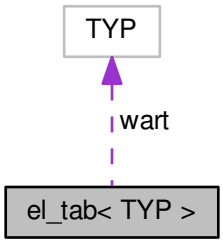
- drzewo.hh

4.4 Dokumentacja szablonu klasy el_tab< TYP >

pojedynczy element tablicy haszujacej

`#include <hashtab.hh>`

Diagram współpracy dla el_tab< TYP >:



Metody publiczne

- el_tab ()
- ~el_tab ()

Atrybuty publiczne

- string klucz

  identyfikator
- TYP wart

  wartosc pola
- bool zajety

  flaga informujaca, czy pole jest zajete

4.4.1 Opis szczegółowy

template<typename TYP>class el_tab< TYP >

pojedynczy element tablicy haszujacej

Definicja w linii 11 pliku hashtab.hh.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 template<typename TYP > el_tab< TYP >::el_tab ( ) [inline]

Definicja w linii 26 pliku hashtab.hh.

4.4.2.2 template<typename TYP > el_tab< TYP >::~el_tab ( ) [inline]

Definicja w linii 27 pliku hashtab.hh.

4.4.3 Dokumentacja atrybutów składowych

4.4.3.1 template<typename TYP > string el_tab< TYP >::klucz

identyfikator

Definicja w linii 16 pliku hashtab.hh.

4.4.3.2 template<typename TYP > TYP el_tab< TYP >::wart

wartosc pola

Definicja w linii 21 pliku hashtab.hh.

4.4.3.3 template<typename TYP > bool el_tab< TYP >::zajety

flaga informujaca, czy pole jest zajete

Definicja w linii 25 pliku hashtab.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

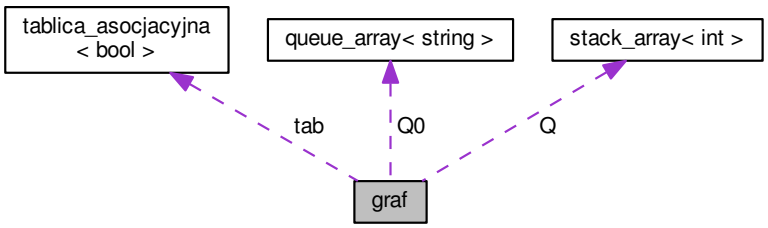
- hashtab.hh

4.5 Dokumentacja klasy graf

Klasa modeluje pojecie grafu w oparciu o liste incydencji, Operacje na grafie mozliwe sa na dwa sposoby \n.

`#include <graf.hh>`

Diagram współpracy dla graf:



Metody publiczne

- graf ()

  Konstruktor nieparametryczny - ustala sposob zarzadzania pamiecia na stosie.
- void dodaj_wierzcholek ()

  Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.
- bool czy_sasiad (unsigned int id1, unsigned int id2)

  Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez identyfikatory.
- bool czy_sasiad (wierzcholek w1, wierzcholek w2)

Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez obiekt klasy wierzcholek.

- void sasiedztwo (wierzcholek w)

  wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez obiekt typu wierzcholek
- void dodaj_wierzcholek (wierzcholek w)

  Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.
- void dodaj_krawedz (unsigned int id1, unsigned int id2, unsigned int waga)

  Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez identyfikatory wierzcholkow.
- void dodaj_krawedz (wierzcholek w1, wierzcholek w2, unsigned int waga)

  Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez obiekty typu wierzcholek.
- void sasiedztwo (unsigned int id)

  wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez identyfikator wierzcholka
- void usun_krawedz (unsigned int id1, unsigned int id2)

  usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez identyfikatory wierzcholkow
- void usun_krawedz (wierzcholek w1, wierzcholek w2)

  usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez obiekt typu wierzcholek
- void usun_wierzcholek (unsigned int id)

  usuwa podany wierzcholek, a scislej, ustawia flage w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika
- void usun_wierzcholek (wierzcholek w)

  usuwa podany wierzcholek, a scislej, ustawia flage w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika
- void wypisz_liste ()

  wypisuje pelna liste incydencji grafu
- void wyczysc ()

  usuwa wszystkie obiekty z listy inceydencji grafu
- bool przeszukaj_wezel (int id, int wzor)

  Jeżeli wezel nei byl odwiedzony, odklada na stos wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.
- void dfs (int id)

  Metoda przeszukuje wglab caly graf.
- bool przeszukaj_wezel_1 (int id, int wzor)

  Jeżeli wezel nei byl odwiedzony, odklada do kolejki wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.
- void bfs (int id)

  Metoda przeszukuje wszerz caly graf.
- bool przeszukaj_wezel_2 (int id, int wzor)

  Jeżeli wezel nei byl odwiedzony, odklada do tablicy asocjacyjnej wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje, poczynajac od tego, do ktorego mamy najkrotsza sciezke.
- void best_first (int id)

  Metoda przeszukuje graf, poczynajac od najkrotszej sciezki.

Atrybuty prywatne

- queue_array< string > Q0
- stack_array< int > Q
- tablica_asocjacyjna< bool > tab

  stos sluzacy do przeszukiwania grafu
- vector< tablica_asocjacyjna < int > > lista_incydencji

  lista incydencji grafu

4.5.1 Opis szczegółowy

Klasa modeluje pojecie grafu w oparciu o liste incydencji, Operacje na grafie mozliwe sa na dwa sposoby \n.

1. Podajac wierzcholek grafu jako parametr metody \n
2. Podajac id wierzcholka jako parametr metody

Definicja w linii 37 pliku graf.hh.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 graf::graf ( ) [inline]

Konstruktor nieparametryczny - ustala sposob zarzadzania pamiecia na stosie.

Definicja w linii 49 pliku graf.hh.

4.5.3 Dokumentacja funkcji składowych

4.5.3.1 void graf::best_first (int *id*)

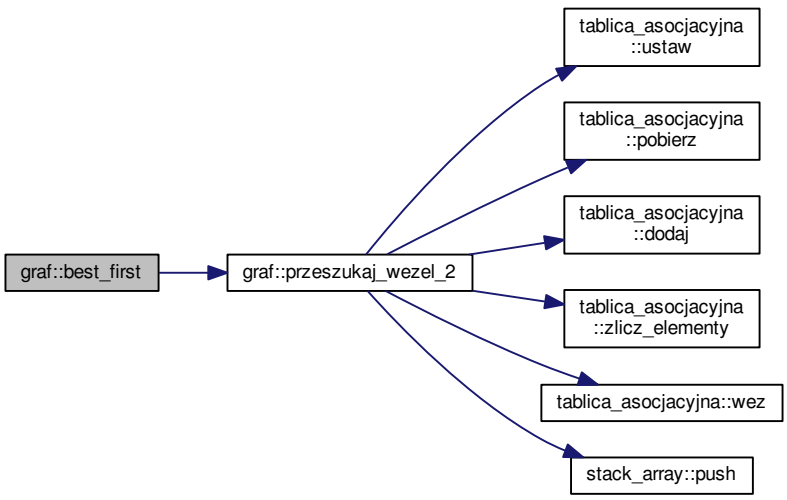
Metoda przeszukuje graf, poczynajac od najkrotszej sciezki.

Parametry

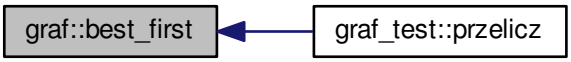
in	*id*	- wezel, ktorego szukamy

Definicja w linii 222 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.2 void graf::bfs (int *id*)

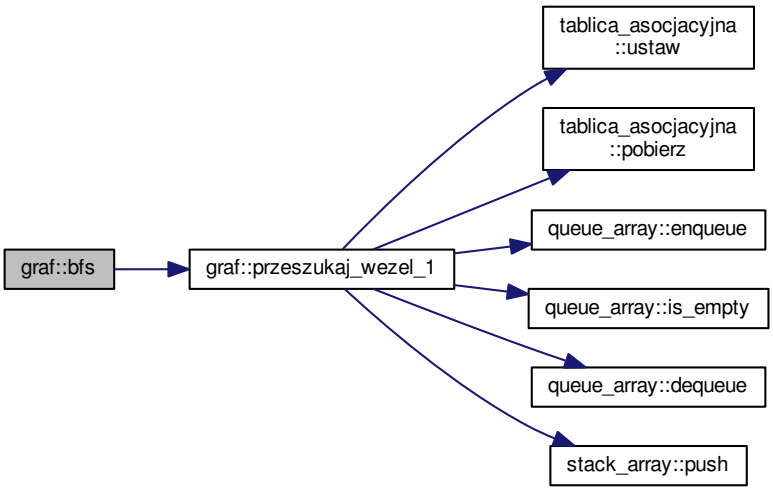
Metoda przeszukuje wszerz caly graf.

Parametry

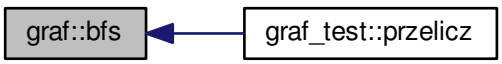
in	*id*	- wezel, ktorego szukamy

Definicja w linii 172 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.3 bool graf::czy_sasiad (unsigned int *id1,* unsigned int *id2*)

Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez identyfikatory.

Parametry

in	*id1*	- id 1. wierzcholka
in	*id2*	- id 2. wierzcholka

Zwraca

true - gdy sa sasiadami, false - gdy nie sa

Definicja w linii 61 pliku graf.cpp.

Oto graf wywoływań tej funkcji:

4.5.3.4 bool graf::czy_sasiad (wierzcholek *w1,* wierzcholek *w2*)

Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez obiekt klasy wierzcholek.

Parametry

in	*w1*	- pierwszy wierzcholek
in	*w2*	- drugi wierzcholek

Zwraca

true - gdy sa sasiadami, false - gdy nie sa

Definicja w linii 68 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

4.5.3.5 void graf::dfs (int *id*)

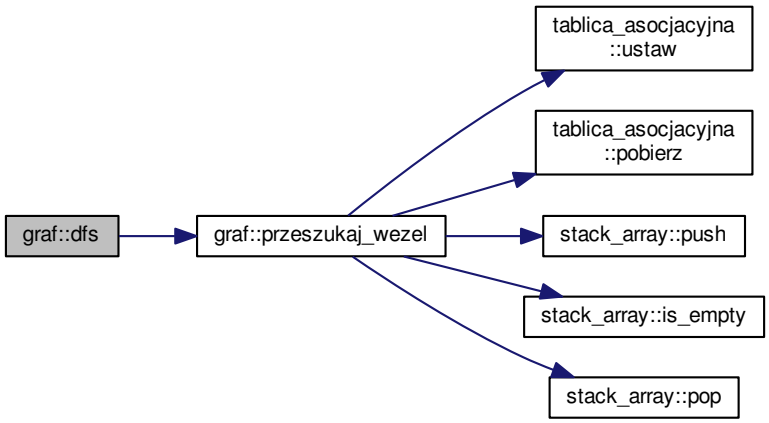
Metoda przeszukuje wglab caly graf.

Parametry

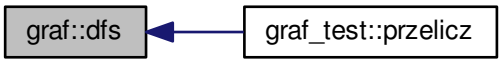
in	*id*	- wezel, ktorego szukamy

Definicja w linii 127 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.6 void graf::dodaj_krawedz (unsigned int *id1*, unsigned int *id2*, unsigned int *waga*)

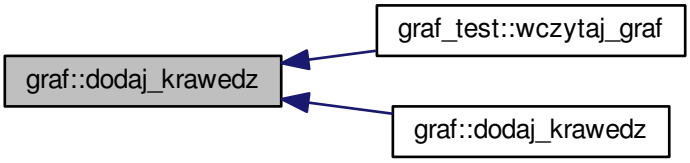
Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez identyfikatory wierzcholkow.

Parametry

in	*id1*	- id 1. wierzcholka
in	*id2*	- id 2. wierzcholka
in	*waga*	- waga krawedzi

Definicja w linii 28 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



4.5.3.7 void graf::dodaj_krawedz (wierzcholek *w1*, wierzcholek *w2*, unsigned int *waga*)

Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez obiekty typu wierzcholek.

Parametry

in	*w1*	- pierwszy wierzcholek
in	*w2*	- drugi wierzcholek
in	*waga*	- waga krawedzi

Definicja w linii 24 pliku graf.cpp.

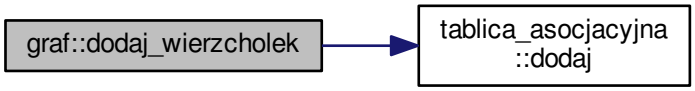
Oto graf wywołań dla tej funkcji:

4.5.3.8 void graf::dodaj_wierzcholek ( )

Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.

Definicja w linii 6 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



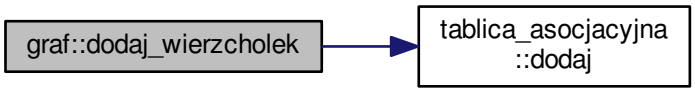
Oto graf wywoływań tej funkcji:

4.5.3.9 void graf::dodaj_wierzcholek (wierzcholek *w*)

Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.

Definicja w linii 14 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



4.5.3.10 bool graf::przeszukaj_wezel (int *id,* int *wzor*)

Jeżeli wezel nei byl odwiedzony, odklada na stos wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.

Parametry

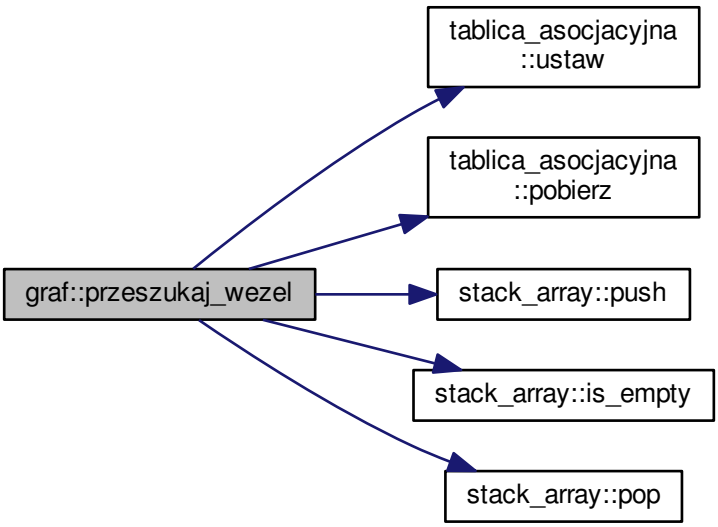
in	*id*	- id wezla, ktory ma byc przeszukany
in	*wzor*	- id wezla, ktorego szukamy

Zwraca

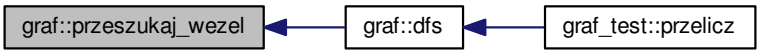
informacja, czy wezel zostal znaleziony

Definicja w linii 95 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.11 bool graf::przeszukaj_wezel_1 (int *id,* int *wzor*)

Jeżeli wezel nei byl odwiedzony, odklada do kolejki wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.

Parametry

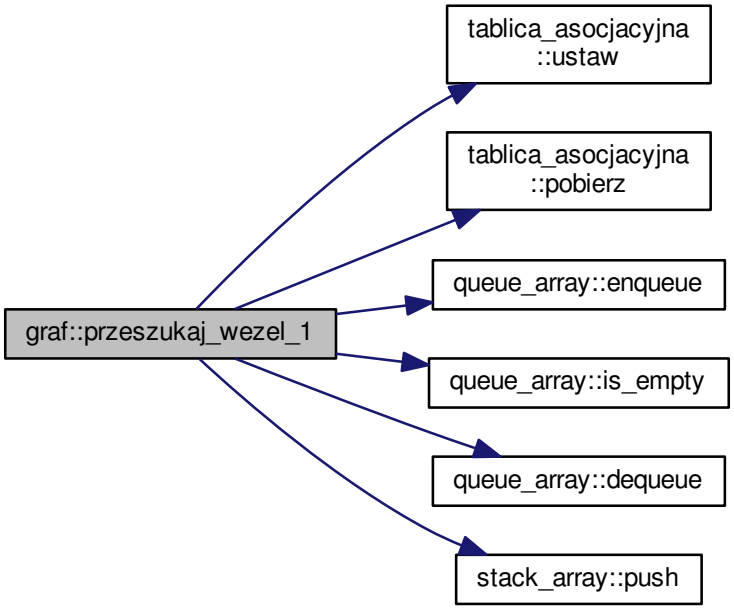
in	*id*	- id wezla, ktory ma byc przeszukany
in	*wzor*	- id wezla, ktorego szukamy

Zwraca

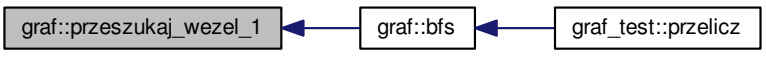
informacja, czy wezel zostal znaleziony

Definicja w linii 138 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.12 bool graf::przeszukaj_wezel_2 (int *id,* int *wzor*)

Jeżeli wezel nei byl odwiedzony, odklada do tablicy asocjacyjnej wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje, poczynajac od tego, do ktorego mamy najkrotsza sciezke.

Parametry

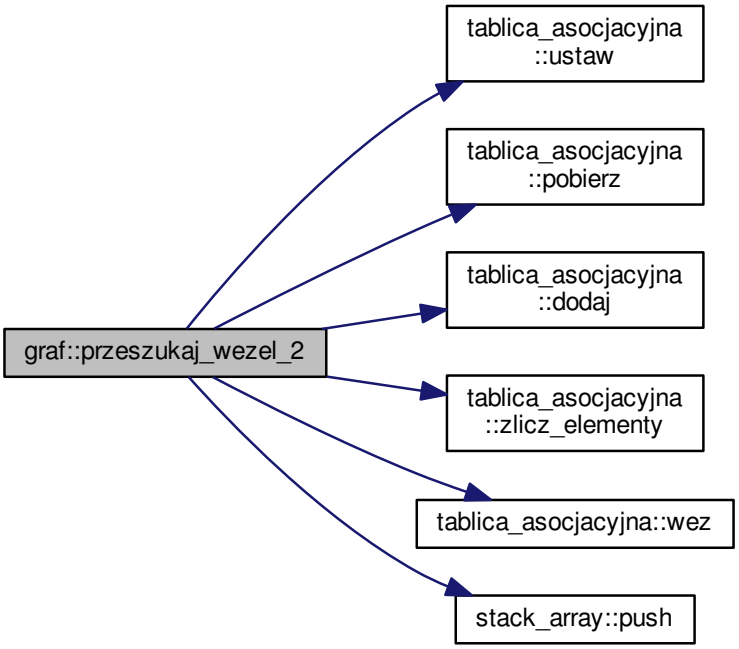
in	*id*	- id wezla, ktory ma byc przeszukany
in	*wzor*	- id wezla, ktorego szukamy

Zwraca

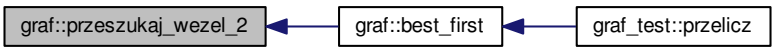
informacja, czy wezel zostal znaleziony

Definicja w linii 184 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.5.3.13 void graf::sasiedztwo (wierzcholek *w*)

wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez obiekt typu wierzcholek

Parametry

in	*w*	- zadany wierzcholek

Definicja w linii 45 pliku graf.cpp.

4.5.3.14 void graf::sasiedztwo (unsigned int *id*)

wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez identyfikator wierzcholka

Parametry

in	*id*	- id wierzcholka

Definicja w linii 36 pliku graf.cpp.

4.5.3.15 void graf::usun_krawedz (unsigned int *id1*, unsigned int *id2*)

usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez identyfikatory wierzcholkow

Parametry

in	*id1*	- id 1. wierzcholka
in	*id2*	- id 2. wierzcholka

Definicja w linii 73 pliku graf.cpp.

Oto graf wywoływań tej funkcji:

4.5.3.16 void graf::usun_krawedz (wierzcholek *w1*, wierzcholek *w2*)

usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez obiekt typu wierzcholek

Parametry

in	*w1*	- pierwszy wierzcholek
in	*w2*	- drugi wierzcholek

Definicja w linii 82 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

4.5.3.17 void graf::usun_wierzcholek (unsigned int *id*)

usuwa podany wierzcholek, a scislej, ustawia flage w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika

Parametry

in	*id*	- id wierzcholka

Definicja w linii 85 pliku graf.cpp.

Oto graf wywoływań tej funkcji:

4.5.3.18 void graf::usun_wierzcholek (wierzcholek *w*)

usuwa podany wierzcholek, a scislej, ustawia flage w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika

Parametry

in	*w*	- wierzcholek ktory trzeba usunac

Definicja w linii 91 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

4.5.3.19 void graf::wyczysc ( ) [inline]

usuwa wszystkie obiekty z listy inceydencji grafu

Definicja w linii 113 pliku graf.hh.

4.5.3.20 void graf::wypisz_liste ( )

wypisuje pelna liste incydencji grafu

Definicja w linii 49 pliku graf.cpp.

4.5.4 Dokumentacja atrybutów składowych

4.5.4.1 vector<tablica_asocjacyjna<int> > graf::lista_incydencji [private]

lista incydencji grafu

Definicja w linii 46 pliku graf.hh.

4.5.4.2 stack_array<int> graf::Q [private]

Definicja w linii 40 pliku graf.hh.

4.5.4.3 queue_array<string> graf::Q0 [private]

Definicja w linii 39 pliku graf.hh.

4.5.4.4 tablica_asocjacyjna<bool> graf::tab [private]

stos sluzacy do przeszukiwania grafu

struktura sluzaca do przechowywania grafu, zawiera informacje, czy wierzcholek byl odwiedzony

Definicja w linii 44 pliku graf.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- graf.hh
- graf.cpp

4.6 Dokumentacja klasy graf_test

modeluje strukture grafów uzytych do badan

`#include <algorytm.hh>`

Diagram dziedziczenia dla graf_test

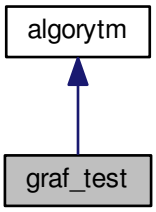
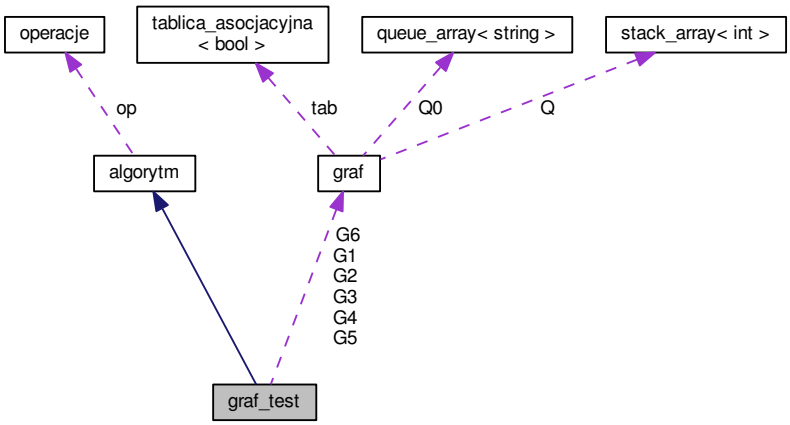


Diagram współpracy dla graf_test:



Metody publiczne

- void wczytaj_graf ()

  na podstawie danych z pliku, tworzone sa grafy
- graf_test (ifstream &plik1, ifstream &plik2, int N, int M, int t)

  konstruktor
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty publiczne

- int typ

  informuje o tym, jaki algorytm zastosowac

Atrybuty prywatne

- graf G1
- graf G2
- graf G3
- graf G4
- graf G5
- graf G6

Dodatkowe Dziedziczone Składowe

4.6.1 Opis szczegółowy

modeluje strukture grafów uzytych do badan

Definicja w linii 291 pliku algorytm.hh.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 graf_test::graf_test (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M,* int *t*) [inline]

konstruktor

Definicja w linii 300 pliku algorytm.hh.

4.6.3 Dokumentacja funkcji składowych

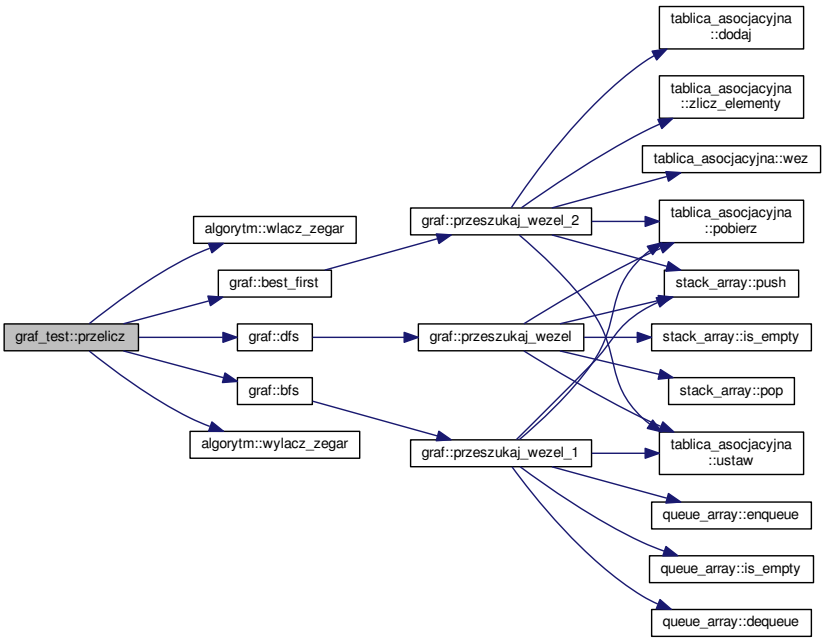
4.6.3.1 float graf_test::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 329 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:

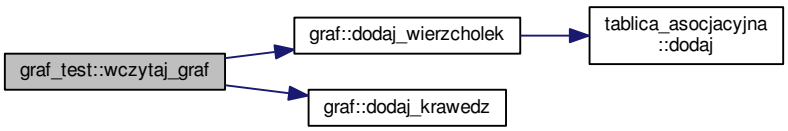


4.6.3.2 void graf_test::wczytaj_graf ( )

na podstawie danych z pliku, tworzone sa grafy

Definicja w linii 262 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 graf graf_test::G1 [private]

\ grafy, na ktorych testuje sie algorytmu przeszukiwania

Definicja w linii 293 pliku algorytm.hh.

4.6.4.2 graf graf_test::G2 [private]

Definicja w linii 293 pliku algorytm.hh.

4.6.4.3 graf graf_test::G3 [private]

Definicja w linii 293 pliku algorytm.hh.

4.6.4.4 graf graf_test::G4 [private]

Definicja w linii 293 pliku algorytm.hh.

4.6.4.5 graf graf_test::G5 [private]

Definicja w linii 293 pliku algorytm.hh.

4.6.4.6 graf graf_test::G6 [private]

Definicja w linii 293 pliku algorytm.hh.

4.6.4.7 int graf_test::typ

informuje o tym, jaki algorytm zastosowac

Definicja w linii 296 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.7 Dokumentacja klasy h_sort

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

`#include <algorytm.hh>`

Diagram dziedziczenia dla h_sort

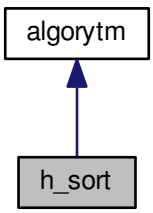
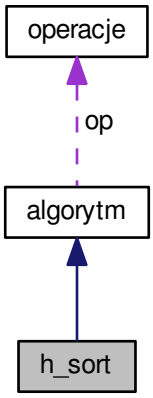


Diagram współpracy dla h_sort:



Metody publiczne

- h_sort (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor klasy
- float przelicz ()

  metoda dokonujaca sortowania danych

Dodatkowe Dziedziczone Składowe

4.7.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

Definicja w linii 208 pliku algorytm.hh.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 h_sort::h_sort (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M*) [inline]

konstruktor klasy

Definicja w linii 211 pliku algorytm.hh.

4.7.3 Dokumentacja funkcji składowych

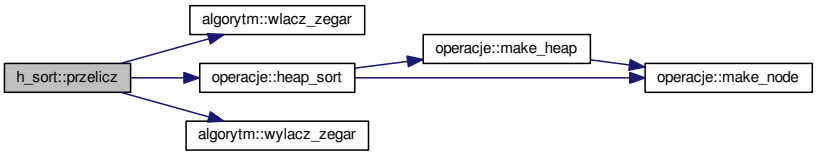
4.7.3.1 float h_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 180 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.8 Dokumentacja klasy h_table

Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwnaia.

`#include <algorytm.hh>`

Diagram dziedziczenia dla h_table

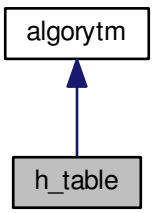
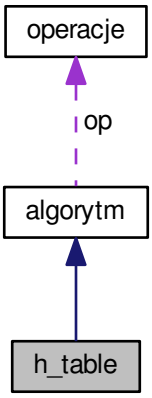


Diagram współpracy dla h_table:



Metody publiczne

- void wczytaj_klucze (ifstream &plik)

  wczytywanie kluczy
- h_table (ifstream &plik1, ifstream &plik2, ifstream &plik3, int N, int M)

  konstruktor
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- string ∗ klucze

  tablica kluczy

Dodatkowe Dziedziczone Składowe

4.8.1 Opis szczegółowy

Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwnaia.

Definicja w linii 247 pliku algorytm.hh.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 h_table::h_table (ifstream & *plik1,* ifstream & *plik2,* ifstream & *plik3,* int *N,* int *M*) [inline]

konstruktor

Definicja w linii 256 pliku algorytm.hh.

4.8.3 Dokumentacja funkcji składowych

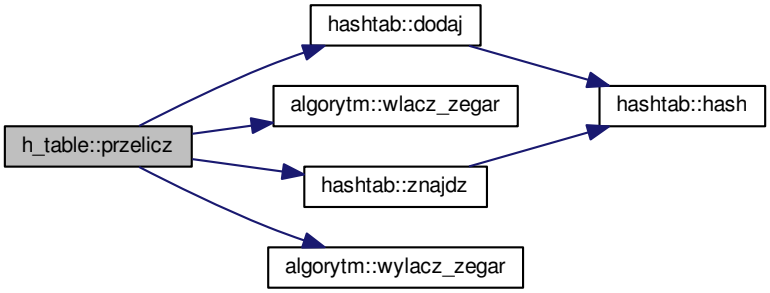
4.8.3.1 float h_table::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 219 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.8.3.2 void h_table::wczytaj_klucze (ifstream & *plik*)

wczytywanie kluczy

Parametry

in	*plik*	- strumien z kluczmai uzytymi podczas testow

Definicja w linii 213 pliku algorytm.cpp.

4.8.4 Dokumentacja atrybutów składowych

4.8.4.1 string* h_table::klucze [private]

tablica kluczy

Definicja w linii 249 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.9 Dokumentacja szablonu klasy hashtab< TYP >

modeluje tablice haszujca w oparciu o kontener klasy el_tab

`#include <hashtab.hh>`

Metody publiczne

- void ustaw_dlugosc (int d)

  ustawia dlugosc tablicy
- hashtab ()

  konstruktor bezparametryczny
- hashtab (int N)

  konsruktor parametryczny
- unsigned long hash (string k)

  funkcja haszujaca
- void dodaj (string k, TYP v)

  metoda dodaje element do tablicy hasuzjacej
- el_tab< TYP > * znajdz (string k)

  metoda szuka zadanego elementu w oparciu o klucz
- void usun (string k)

  usuwa element jesli znajduje sie w tablicy
- void wypisz ()

Atrybuty prywatne

- int dlugosc

  dlugosc tablicy
- vector< el_tab< TYP > > tab

  tablica haszujaca

4.9.1 Opis szczegółowy

template<typename TYP>class hashtab< TYP >

modeluje tablice haszujca w oparciu o kontener klasy el_tab

Definicja w linii 34 pliku hashtab.hh.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 template<typename TYP> hashtab< TYP >::hashtab ( ) [inline]

konstruktor bezparametryczny

Definicja w linii 43 pliku hashtab.hh.

4.9.2.2 template<typename TYP> hashtab< TYP >::hashtab (int *N*) [inline]

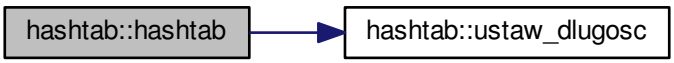
konsruktor parametryczny

Parametry

in	*N*	- rozmiar tablicy

Definicja w linii 47 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



4.9.3 Dokumentacja funkcji składowych

4.9.3.1 template<typename TYP> void hashtab< TYP >::dodaj (string *k*, TYP *v*) [inline]

metoda dodaje element do tablicy hasuzjacej

Definicja w linii 59 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:

4.9.3.2 template<typename TYP> unsigned long hashtab< TYP >::hash (string *k*) [inline]

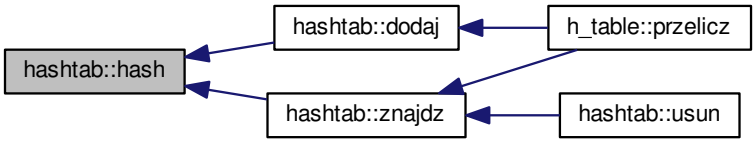
funkcja haszujaca

Zwraca

h - liczba, ktora po kompresji bedzie indeksem danego elementu

Definicja w linii 51 pliku hashtab.hh.

Oto graf wywoływań tej funkcji:

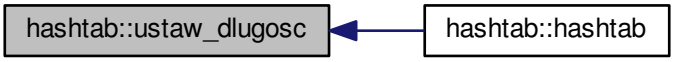


4.9.3.3 template<typename TYP> void hashtab< TYP >::ustaw_dlugosc (int *d*) [inline]

ustawia dlugosc tablicy

Definicja w linii 41 pliku hashtab.hh.

Oto graf wywoływań tej funkcji:



4.9.3.4 template<typename TYP> void hashtab< TYP >::usun (string *k*) [inline]

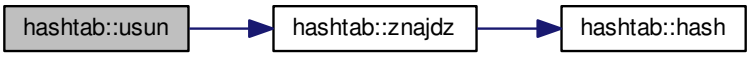
usuwa element jesli znajduje sie w tablicy

Parametry

in	*k*	- klucz

Definicja w linii 89 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



4.9.3.5 template<typename TYP> void hashtab< TYP >::wypisz ( ) [inline]

Definicja w linii 93 pliku hashtab.hh.

4.9.3.6 template<typename TYP> el_tab<TYP>* hashtab< TYP >::znajdz (string *k*) [inline]

metoda szuka zadanego elementu w oparciu o klucz

Parametry

in	*k*	- klucz elementu

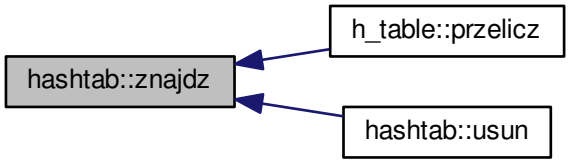
Zwraca

znaleziony element

Definicja w linii 74 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:



4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 template<typename TYP> int hashtab< TYP >::dlugosc [private]

dlugosc tablicy

Definicja w linii 36 pliku hashtab.hh.

4.9.4.2 template<typename TYP> vector<el_tab<TYP> > hashtab< TYP >::tab [private]

tablica haszujaca

Definicja w linii 38 pliku hashtab.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- hashtab.hh

4.10 Dokumentacja klasy kolejka_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

`#include <algorytm.hh>`

Diagram dziedziczenia dla kolejka_lista

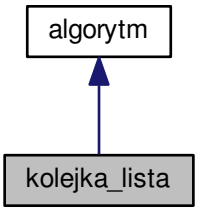
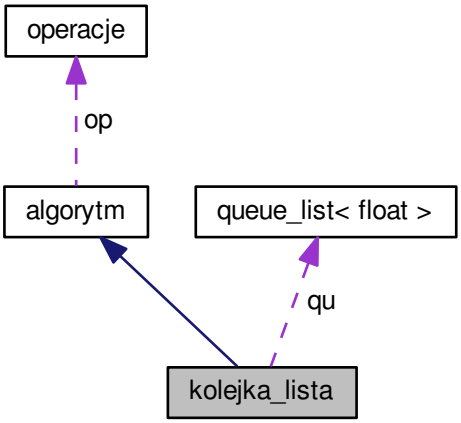


Diagram współpracy dla kolejka_lista:



Metody publiczne

- kolejka_lista (ifstream &plik1, ifstream &plik2, int N, int M)
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- queue_list< float > qu

Dodatkowe Dziedziczone Składowe

4.10.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 191 pliku algorytm.hh.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 kolejka_lista::kolejka_lista (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M*) [inline]

Definicja w linii 194 pliku algorytm.hh.

4.10.3 Dokumentacja funkcji składowych

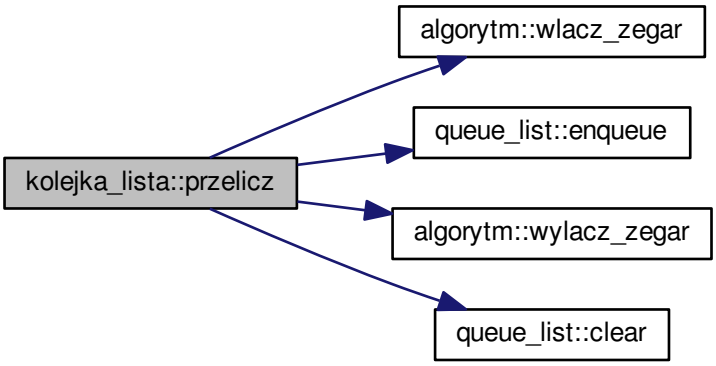
4.10.3.1 float kolejka_lista::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 160 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.10.4 Dokumentacja atrybutów składowych

4.10.4.1 queue_list<float> kolejka_lista::qu [private]

Definicja w linii 192 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.11 Dokumentacja klasy kolejka_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

`#include <algorytm.hh>`

Diagram dziedziczenia dla kolejka_tablica

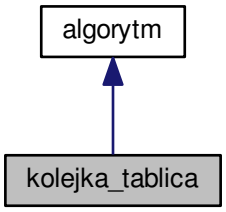
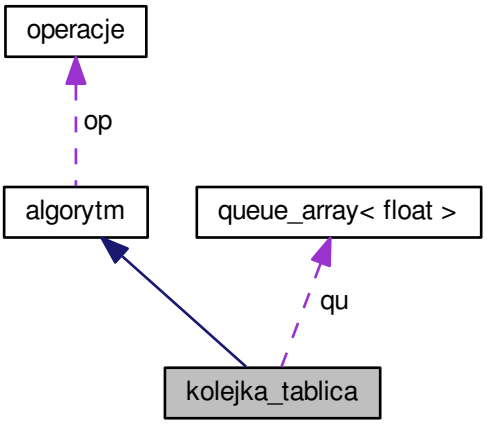


Diagram współpracy dla kolejka_tablica:



Metody publiczne

- kolejka_tablica (ifstream &plik1, ifstream &plik2, int N, int M, flag F)

  konstruktor - ustawia flage w zadany stan
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- queue_array< float > qu

Dodatkowe Dziedziczone Składowe

4.11.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 179 pliku algorytm.hh.

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 kolejka_tablica::kolejka_tablica (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M,* flag *F*) [inline]

konstruktor - ustawia flage w zadany stan

Definicja w linii 185 pliku algorytm.hh.

4.11.3 Dokumentacja funkcji składowych

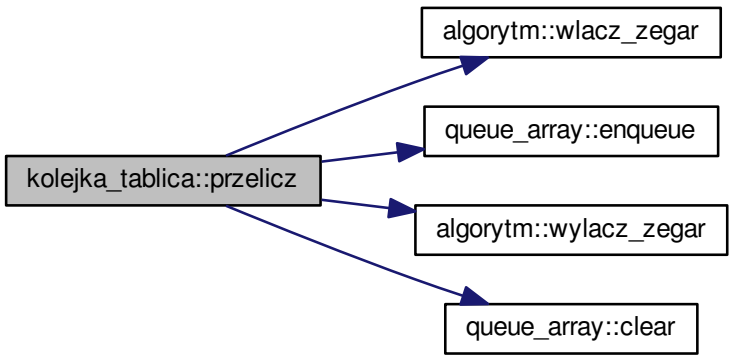
4.11.3.1 float kolejka_tablica::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 148 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.11.4 Dokumentacja atrybutów składowych

4.11.4.1 queue_array<float> kolejka_tablica::qu [private]

Definicja w linii 180 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.12 Dokumentacja klasy m_sort

klasa reprezentuje dane poddane sortowaniu przez scalanie

`#include <algorytm.hh>`

Diagram dziedziczenia dla m_sort

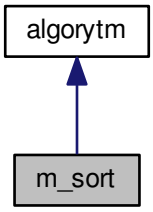
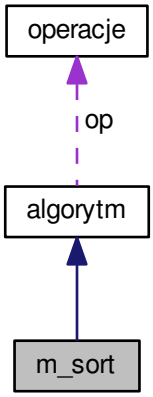


Diagram współpracy dla m_sort:



Metody publiczne

- m_sort (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor
- float przelicz ()

  metoda dokonujaca sortowania danych

Dodatkowe Dziedziczone Składowe

4.12.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez scalanie

Definicja w linii 217 pliku algorytm.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 m_sort::m_sort (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M*) [inline]

konstruktor

Definicja w linii 220 pliku algorytm.hh.

4.12.3 Dokumentacja funkcji składowych

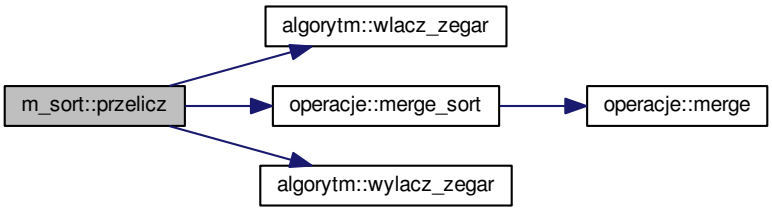
4.12.3.1 float m_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 189 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.13 Dokumentacja klasy mnozenie

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2

`#include <algorytm.hh>`

Diagram dziedziczenia dla mnozenie

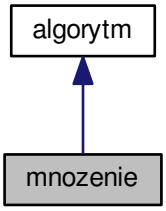
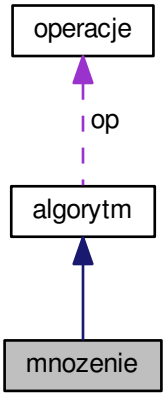


Diagram współpracy dla mnozenie:



Metody publiczne

- mnozenie (ifstream &plik1, ifstream &plik2, int N, int M)
- float przelicz ()

  wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

Dodatkowe Dziedziczone Składowe

4.13.1 Opis szczegółowy

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2

Definicja w linii 139 pliku algorytm.hh.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 mnozenie::mnozenie (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M*) [inline]

/brief konstruktor przekazuje do pol klasy informacje o nazwach pliku wejsciowego i wzorcowego

Parametry

in	*plik1*	- plik wejsciowy
in	*plik2*	- plik wzorcowy
in	*N*	- ilosc danych wejsciowych
in	*M*	- ilosc powtorzen

Definicja w linii 148 pliku algorytm.hh.

4.13.3 Dokumentacja funkcji składowych

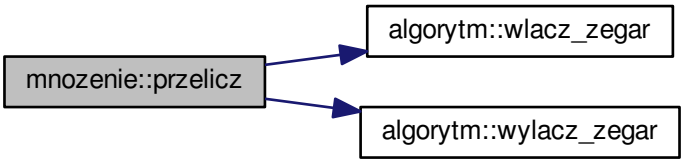
4.13.3.1 float mnozenie::przelicz ( ) [virtual]

wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

Reimplementowana z algorytm.

Definicja w linii 114 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.14 Dokumentacja klasy operacje

Klasa modeluje tablice z danymi i metody sluzace do operacji na niej.

`#include <operacje.hh>`

Metody publiczne

- operacje ()

  konstruktor bezparametryczny
- operacje (int N)

  konstruktor parametryczny - alokuje pamiec w dynamicznej tablicy `tab`
- bool zamien_elementy (int i, int j)

  Metoda zamienia 2 elementy tablicy.
- void quick_sort (int l, int p)

  Metoda Dokonuje sortownaia szybkiego.

- void make_node (int rozmiar, int i)

  Metoda tworzy wezel drzewa, przypisujac mu 2 synow, ustawiajac ich w odpowiedniej kolejnosci (ojciec ma najwieksza wartosc)
- void make_heap ()

  Metoda tworzy kopiec binarny.
- void heap_sort ()

  Metoda dokonuje sortowania po uprzednim utworzeniu kopca.
- void merge (int poczatek, int srodek, int koniec)

  Metoda scala dwie czesci tablicy, jednoczesnie je porzadkujac.
- void merge_sort (int poczatek, int koniec)
- void odwroc_tablice ()

  metoda odwraca wszystkie elementy tablicy
- void dodaj_element (float e)

  metoda dodaje element do tablicy, alokujac dodatkowa pamiec
- void dodaj_elementy (float *tab2, int rozm)

  metoda dodaje elementy do tablicy
- void operator= (float *tab1)

  Przeciazenie operatora przypisania; przypisuje elementy tablicy `tab1` do tablicy bedacej polem klasy.
- bool operator== (float *tab1)

  Przeciazenie operatora porownania; metoda porownuje zawartosci dwoch tablic.
- float & operator[] (int ind)

Atrybuty publiczne

- int n

  ilosc elementow w tablicy
- float * tab

  tablica z liczbami

4.14.1 Opis szczegółowy

Klasa modeluje tablice z danymi i metody sluzace do operacji na niej.

Definicja w linii 11 pliku operacje.hh.

4.14.2 Dokumentacja konstruktora i destruktora

4.14.2.1 operacje::operacje ( )

konstruktor bezparametryczny

4.14.2.2 operacje::operacje (int *N*) [inline]

konstruktor parametryczny - alokuje pamiec w dynamicznej tablicy `tab`

Parametry

`in`	*N*	- ilosc elementow w tablicy; parametr przypisywany do pola `n` w klasie, oraz alokuje pamiec o takim wlasnie rozmiarze

Definicja w linii 28 pliku operacje.hh.

4.14.3 Dokumentacja funkcji składowych

4.14.3.1 void operacje::dodaj_element (float *e*)

metoda dodaje element do tablicy, alokujac dodatkowa pamiec

Parametry

in	*e*	- element, ktory nalezy dolaczyc do tablicy

Definicja w linii 27 pliku operacje.cpp.

4.14.3.2 void operacje::dodaj_elementy (float * *tab2*, int *rozm*)

metoda dodaje elementy do tablicy

Parametry

in	*tab2*	- tablica, ktora nalezy dolaczyc
in	*rozm*	- rozmiar tablicy tab2

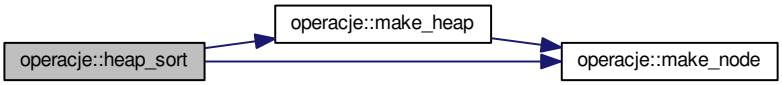
Definicja w linii 46 pliku operacje.cpp.

4.14.3.3 void operacje::heap_sort ( )

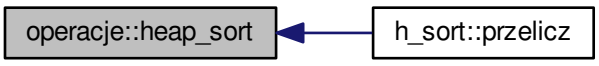
Metoda dokonuje sortowania po uprzednim utworzeniu kopca.

Definicja w linii 116 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



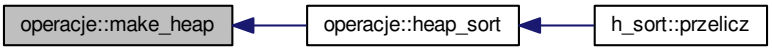
4.14.3.4 void operacje::make_heap ( )

Metoda tworzy kopiec binarny.

Definicja w linii 110 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:

Oto graf wywoływań tej funkcji:



4.14.3.5 void operacje::make_node (int *rozmiar,* int *i*)

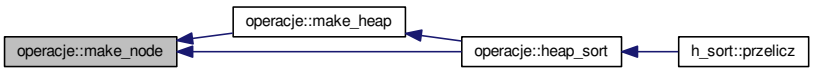
Metoda tworzy wezel drzewa, przypisujac mu 2 synow, ustawiajac ich w odpowiedniej kolejnosci (ojciec ma najwieksza wartosc)

Parametry

in	*rozmiar*	- rozmiar tablicy
in	*i*	- indeks elementu, do ktorego przypisujemy synow

Definicja w linii 95 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



4.14.3.6 void operacje::merge (int *poczatek,* int *srodek,* int *koniec*)

Metoda scala dwie czesci tablicy, jednoczesnie je porzadkujac.

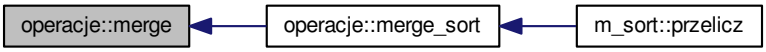
Parametry

in	*poczatek*	- pierwszy indeks tablicy
in	*srodek*	- srodkowy indeks tablicy

in	*koniec*	- ostatni indeks tablicy

Definicja w linii 130 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



4.14.3.7 void operacje::merge_sort (int *poczatek*, int *koniec*)

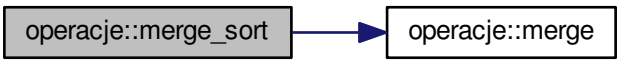
\ brief Metoda dokonuje sortowania poprzez rekurencyjne wywolanie dla obu polow tablic, nastepnie metoda dokonuje scalenia danych

Parametry

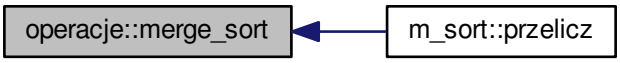
in	*poczatek*	- pierwszy indeks tablicy
in	*koniec*	- ostatni indeks tablicy

Definicja w linii 166 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.14.3.8 void operacje::odwroc_tablice ( )

metoda odwraca wszystkie elementy tablicy

Definicja w linii 12 pliku operacje.cpp.

4.14.3.9 void operacje::operator= (float * *tab1*)

Przeciazenie operatora przypisania; przypisuje elementy tablicy `tab1` do tablicy bedacej polem klasy.

Parametry

in	*tab1*	- tablica, ktorej zawartosc przypisujemy

Definicja w linii 63 pliku operacje.cpp.

4.14.3.10 bool operacje::operator== (float * *tab1*)

Przeciazenie operatora porownania; metoda porownuje zawartosci dwoch tablic.

Parametry

in	*tab1*	- tablica, ktorej wartosci porownujemy

Zwraca

true - gdy zawartsoc tablic jest identyczna false - w przeciwnym przypadku

Definicja w linii 69 pliku operacje.cpp.

4.14.3.11 float& operacje::operator[] (int *ind*) [inline]

Definicja w linii 88 pliku operacje.hh.

4.14.3.12 void operacje::quick_sort (int *l*, int *p*)

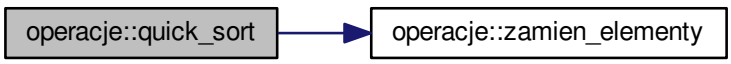
Metoda Dokonuje sortownaia szybkiego.

Parametry

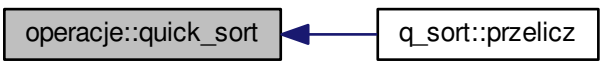
in	*l*	- pierwszy indeks tablicy
in	*p*	- ostatni indeks tablicy

Definicja w linii 77 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.14.3.13 bool operacje::zamien_elementy (int *i*, int *j*)

Metoda zamienia 2 elementy tablicy.

Parametry

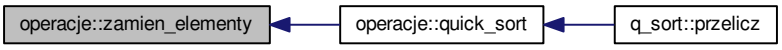
in	*i*	- element tablicy
in	*j*	- element tablicy

Zwraca

true - gdy elementy nie wykraczaja poza zakres tablicy false - w przeciwnym przypadku

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



4.14.4 Dokumentacja atrybutów składowych

4.14.4.1 int operacje::n

ilosc elementow w tablicy

Definicja w linii 16 pliku operacje.hh.

4.14.4.2 float* operacje::tab

tablica z liczbami

Definicja w linii 19 pliku operacje.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- operacje.hh
- operacje.cpp

4.15 Dokumentacja klasy q_sort

klasa reprezentuje dane poddane sortowaniu szybkiemu

`#include <algorytm.hh>`

Diagram dziedziczenia dla q_sort

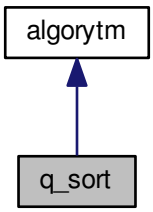
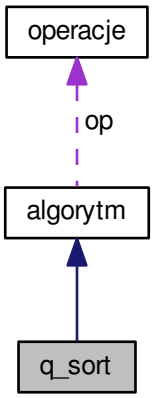


Diagram współpracy dla q_sort:



Metody publiczne

- q_sort (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor klasy
- float przelicz ()

  metoda dokonujaca sortowania danych

Dodatkowe Dziedziczone Składowe

4.15.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkiemu

Definicja w linii 200 pliku algorytm.hh.

4.15.2 Dokumentacja konstruktora i destruktora

4.15.2.1 q_sort::q_sort (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M*) [inline]

konstruktor klasy

Definicja w linii 203 pliku algorytm.hh.

4.15.3 Dokumentacja funkcji składowych

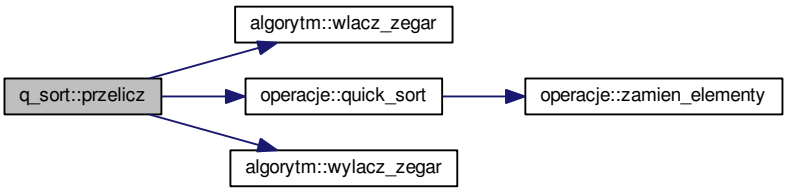
4.15.3.1 float q_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 171 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

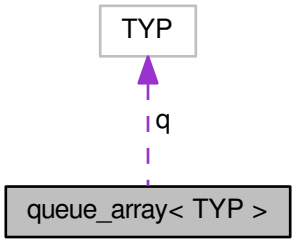
- algorytm.hh
- algorytm.cpp

4.16 Dokumentacja szablonu klasy queue_array< TYP >

Modeluje kolejke w oparciu o tablice.

`#include <kolejka.hh>`

Diagram współpracy dla queue_array< TYP >:



Metody publiczne

- queue_array ()

  konstruktor bezparametryczny
- queue_array (flag F)

  konstruktor parametryczny - ustawia flage na zadana pozycje
- int size ()
- bool is_empty ()
- void enqueue (TYP element)

  Dodaje element na poczatek kolejki w zaleznosci od wybranego trybu powiekszania tablicy.
- TYP dequeue ()

  usuwa element z konca kolejki
- void clear ()

  czysci kolejke

Atrybuty publiczne

- flag f

  flaga trybu zwiekszania pamieci , przyjmuje wartosc : plus1 - dla trybu kazdorazowego powiekszania pamieci x2 - dla trybu podwajania rozmiaru struktury

Atrybuty prywatne

- TYP * q
- int s
- int sp

4.16.1 Opis szczegółowy

template<typename TYP>class queue_array< TYP >

Modeluje kolejke w oparciu o tablice.

Definicja w linii 50 pliku kolejka.hh.

4.16.2 Dokumentacja konstruktora i destruktora

4.16.2.1 template<typename TYP> queue_array< TYP >::queue_array ( ) [inline]

konstruktor bezparametryczny

Definicja w linii 63 pliku kolejka.hh.

4.16.2.2 template<typename TYP> queue_array< TYP >::queue_array (flag *F*) [inline]

konstruktor parametryczny - ustawia flage na zadana pozycje

Definicja w linii 65 pliku kolejka.hh.

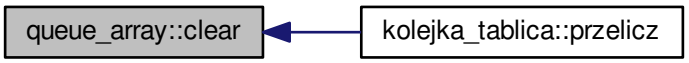
4.16.3 Dokumentacja funkcji składowych

4.16.3.1 template<typename TYP> void queue_array< TYP >::clear ( ) [inline]

czysci kolejke

Definicja w linii 173 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:

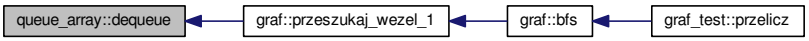


4.16.3.2 template<typename TYP> TYP queue_array< TYP >::dequeue ( ) [inline]

usuwa element z konca kolejki

Definicja w linii 129 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:

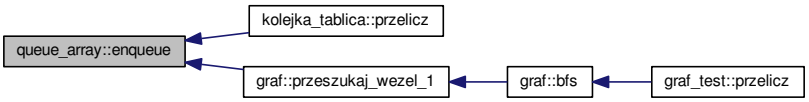


4.16.3.3 template<typename TYP> void queue_array< TYP >::enqueue (TYP *element*) [inline]

Dodaje element na poczatek kolejki w zaleznosci od wybranego trybu powiekszania tablicy.

Definicja w linii 82 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



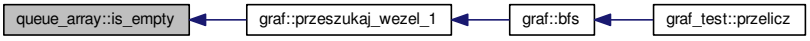
4.16.3.4 template<typename TYP> bool queue_array< TYP >::is_empty ( ) [inline]

Zwraca

false - gdy kolejka nie jest pusta, true , gdy pusta

Definicja w linii 75 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



4.16.3.5 template<typename TYP> int queue_array< TYP >::size ( ) [inline]

Zwraca

rozmiar kolejki

Definicja w linii 70 pliku kolejka.hh.

4.16.4 Dokumentacja atrybutów składowych

4.16.4.1 template<typename TYP> flag queue_array< TYP >::f

flaga trybu zwiekszania pamieci , przyjmuje wartosc : plus1 - dla trybu kazdorazowego powiekszania pamieci x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 59 pliku kolejka.hh.

4.16.4.2 template<typename TYP> TYP* queue_array< TYP >::q [private]

Definicja w linii 51 pliku kolejka.hh.

4.16.4.3 template<typename TYP> int queue_array< TYP >::s [private]

Definicja w linii 52 pliku kolejka.hh.

4.16.4.4 template<typename TYP> int queue_array< TYP >::sp [private]

Definicja w linii 52 pliku kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- kolejka.hh

4.17 Dokumentacja szablonu klasy queue_list< TYP >

Modeluje kolejke oparta na liscie STL.

`#include <kolejka.hh>`

Metody publiczne

- bool is_empty ()
- int size ()
- void enqueue (TYP &element)

  dodaje element
- TYP dequeue ()

  usuwa element
- void clear ()

  czysci stos

Atrybuty prywatne

- list< TYP > q

4.17.1 Opis szczegółowy

template<typename TYP>class queue_list< TYP >

Modeluje kolejke oparta na liscie STL.

Definicja w linii 19 pliku kolejka.hh.

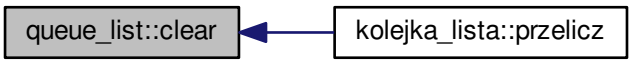
4.17.2 Dokumentacja funkcji składowych

4.17.2.1 template<typename TYP> void queue_list< TYP >::clear ( ) [inline]

czysci stos

Definicja w linii 41 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



4.17.2.2 template<typename TYP> TYP queue_list< TYP >::dequeue ( ) [inline]

usuwa element

Definicja w linii 35 pliku kolejka.hh.

4.17.2.3 template<typename TYP> void queue_list< TYP >::enqueue (TYP & *element*) [inline]

dodaje element

Definicja w linii 33 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:

4.17.2.4 template<typename TYP> bool queue_list< TYP >::is_empty ( ) [inline]

Zwraca

false - gdy kolejka nie jest pusta, true , gdy pusta

Definicja w linii 26 pliku kolejka.hh.

4.17.2.5 template<typename TYP> int queue_list< TYP >::size ( ) [inline]

Zwraca

rozmiar kolejki

Definicja w linii 31 pliku kolejka.hh.

4.17.3 Dokumentacja atrybutów składowych

4.17.3.1 template<typename TYP> list<TYP> queue_list< TYP >::q [private]

Definicja w linii 20 pliku kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

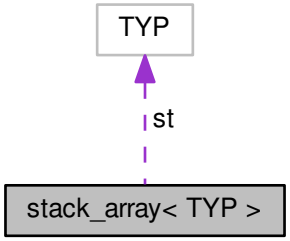
- kolejka.hh

4.18 Dokumentacja szablonu klasy stack_array< TYP >

Modeluje stos w oparciu o tablice.

`#include <stos.hh>`

Diagram współpracy dla stack_array< TYP >:



Metody publiczne

- stack_array ()
  konstruktor bezparametryczny
- stack_array (flag F)
  konstruktor parametryczny - ustawia flage na zadana pozycje
- bool is_empty ()
- int size ()
- void push (TYP element)
  Dodaje element na wierzch stosu w zaleznosci od wybranego trybu powiekszania tablicy.
- TYP pop ()
  zdejmuje element ze stosu
- void clear ()
  czysci stos

Atrybuty publiczne

- flag f
  flaga trybu zwiekszania pamieci , przyjmuje wartosc :
  plus1 - dla trybu kazdorazowego powiekszania pamieci
  x2 - dla trybu podwajania rozmiaru struktury

Atrybuty prywatne

- TYP * st
- int s
- int sp

4.18.1 Opis szczegółowy

template<typename TYP>class stack_array< TYP >

Modeluje stos w oparciu o tablice.

Definicja w linii 59 pliku stos.hh.

4.18.2 Dokumentacja konstruktora i destruktora

4.18.2.1 template<typename TYP> stack_array< TYP >::stack_array ( ) [inline]

konstruktor bezparametryczny

Definicja w linii 72 pliku stos.hh.

4.18.2.2 template<typename TYP> stack_array< TYP >::stack_array (flag *F*) [inline]

konstruktor parametryczny - ustawia flage na zadana pozycje

Definicja w linii 74 pliku stos.hh.

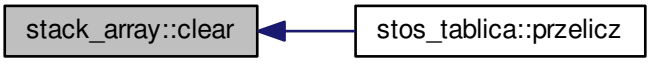
4.18.3 Dokumentacja funkcji składowych

4.18.3.1 template<typename TYP> void stack_array< TYP >::clear ( ) [inline]

czysci stos

Definicja w linii 183 pliku stos.hh.

Oto graf wywoływań tej funkcji:



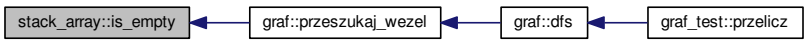
4.18.3.2 template<typename TYP> bool stack_array< TYP >::is_empty ( ) [inline]

Zwraca

false - gdy stos nie jest pusty, true , gdy pusty

Definicja w linii 79 pliku stos.hh.

Oto graf wywoływań tej funkcji:



4.18.3.3 template<typename TYP> TYP stack_array< TYP >::pop ( ) [inline]

zdejmuje element ze stosu

Definicja w linii 140 pliku stos.hh.

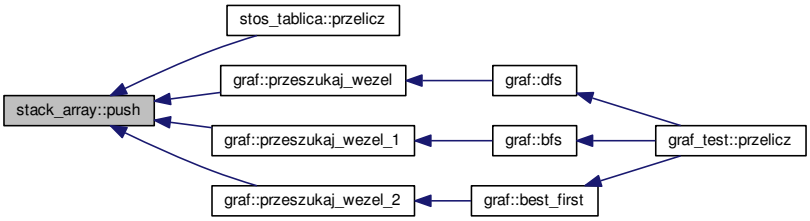
Oto graf wywoływań tej funkcji:

4.18.3.4 template<typename TYP> void stack_array< TYP >::push (TYP *element*) [inline]

Dodaje element na wierzch stosu w zaleznosci od wybranego trybu powiekszania tablicy.

Definicja w linii 91 pliku stos.hh.

Oto graf wywoływań tej funkcji:



4.18.3.5 template<typename TYP> int stack_array< TYP >::size ( ) [inline]

Zwraca

rozmiar ztosu

Definicja w linii 87 pliku stos.hh.

4.18.4 Dokumentacja atrybutów składowych

4.18.4.1 template<typename TYP> flag stack_array< TYP >::f

flaga trybu zwiekszania pamieci , przyjmuje wartosc :

plus1 - dla trybu kazdorazowego powiekszania pamieci

x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 68 pliku stos.hh.

4.18.4.2 template<typename TYP> int stack_array< TYP >::s [private]

Definicja w linii 61 pliku stos.hh.

4.18.4.3 template<typename TYP> int stack_array< TYP >::sp [private]

Definicja w linii 61 pliku stos.hh.

4.18.4.4 template<typename TYP> TYP* stack_array< TYP >::st [private]

Definicja w linii 60 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stos.hh

4.19 Dokumentacja szablonu klasy stack_list< TYP >

Modeluje stos oparty na liscie STL.

`#include <stos.hh>`

Metody publiczne

- bool is_empty ()
- int size ()
- void push (TYP &element)

  Dodaje element na wierzch stosu.
- TYP pop ()

  zdejmuje element z wierzchu stosu
- void clear ()

  czysci stos

Atrybuty prywatne

- list< TYP > st

4.19.1 Opis szczegółowy

template<typename TYP>class stack_list< TYP >

Modeluje stos oparty na liscie STL.

Definicja w linii 22 pliku stos.hh.

4.19.2 Dokumentacja funkcji składowych

4.19.2.1 template<typename TYP> void stack_list< TYP >::clear ( ) [inline]

czysci stos

Definicja w linii 50 pliku stos.hh.

Oto graf wywoływań tej funkcji:

4.19.2.2 template<typename TYP> bool stack_list< TYP >::is_empty ( ) [inline]

Zwraca

false - gdy stos nie jest pusty, true , gdy pusty

Definicja w linii 29 pliku stos.hh.

4.19.2.3 template<typename TYP> TYP stack_list< TYP >::pop ( ) [inline]

zdejmuje element z wierzchu stosu

Definicja w linii 42 pliku stos.hh.

4.19.2.4 template<typename TYP> void stack_list< TYP >::push (TYP & *element*) [inline]

Dodaje element na wierzch stosu.

Definicja w linii 38 pliku stos.hh.

Oto graf wywoływań tej funkcji:

4.19.2.5 template<typename TYP> int stack_list< TYP >::size ( ) [inline]

Zwraca

rozmiar ztosu

Definicja w linii 34 pliku stos.hh.

4.19.3 Dokumentacja atrybutów składowych

4.19.3.1 template<typename TYP> list<TYP> stack_list< TYP >::st [private]

Definicja w linii 23 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stos.hh

4.20 Dokumentacja klasy stos_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

`#include <algorytm.hh>`

Diagram dziedziczenia dla stos_lista

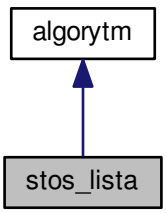
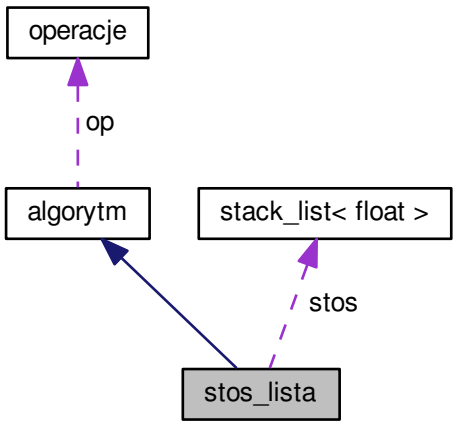


Diagram współpracy dla stos_lista:



Metody publiczne

- stos_lista (ifstream &plik1, ifstream &plik2, int N, int M)
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- stack_list< float > stos

Dodatkowe Dziedziczone Składowe

4.20.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 170 pliku algorytm.hh.

4.20.2 Dokumentacja konstruktora i destruktora

4.20.2.1 stos_lista::stos_lista (ifstream & *plik1*, ifstream & *plik2*, int *N*, int *M*) [inline]

Definicja w linii 173 pliku algorytm.hh.

4.20.3 Dokumentacja funkcji składowych

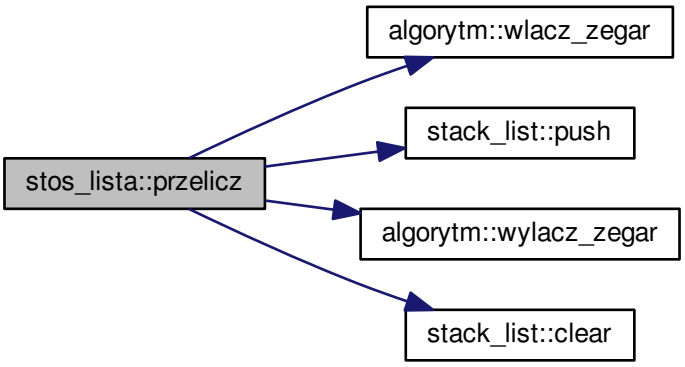
4.20.3.1 float stos_lista::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 136 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.20.4 Dokumentacja atrybutów składowych

4.20.4.1 stack_list<float> stos_lista::stos [private]

Definicja w linii 171 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.21 Dokumentacja klasy stos_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

`#include <algorytm.hh>`

Diagram dziedziczenia dla stos_tablica

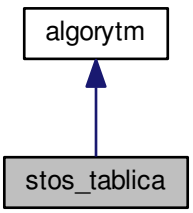
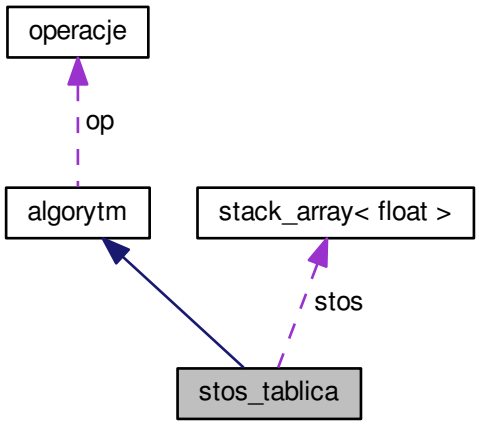


Diagram współpracy dla stos_tablica:



Metody publiczne

- stos_tablica (ifstream &plik1, ifstream &plik2, int N, int M, flag F)

  konstruktor - ustawia flage w zadany stan
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- stack_array< float > stos

Dodatkowe Dziedziczone Składowe

4.21.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 158 pliku algorytm.hh.

4.21.2 Dokumentacja konstruktora i destruktora

4.21.2.1 stos_tablica::stos_tablica (ifstream & *plik1,* ifstream & *plik2,* int *N,* int *M,* flag *F*) [inline]

konstruktor - ustawia flage w zadany stan

Definicja w linii 164 pliku algorytm.hh.

4.21.3 Dokumentacja funkcji składowych

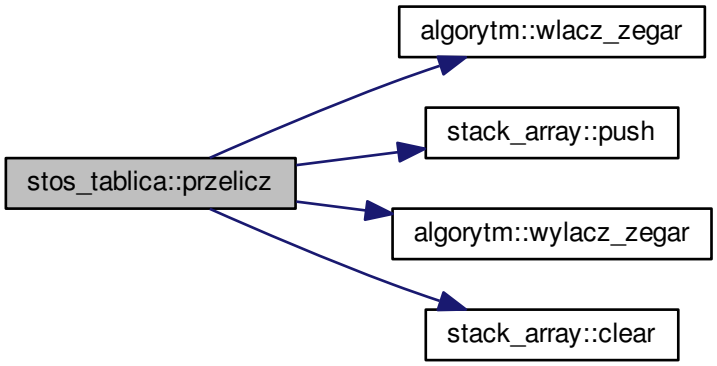
4.21.3.1 float stos_tablica::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 125 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.21.4 Dokumentacja atrybutów składowych

4.21.4.1 stack_array<float> stos_tablica::stos [private]

Definicja w linii 159 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.22 Dokumentacja klasy tab_aso

Modeluje tablice asocjacyjna przeznaczona do testowania szybkosci wyszukiwnaia.

`#include <algorytm.hh>`

Diagram dziedziczenia dla tab_aso

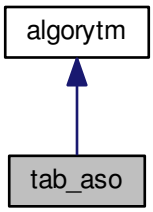
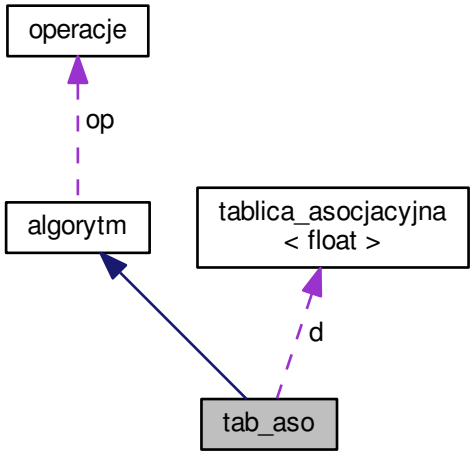


Diagram współpracy dla tab_aso:



Metody publiczne

- void wczytaj_klucze (ifstream &plik)

  wczytywanie kluczy
- tab_aso (ifstream &plik1, ifstream &plik2, ifstream &plik3, int N, int M)

  konstruktor
- float przelicz ()

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- tablica_asocjacyjna< float > d

  tablica asocjacyjna

- string * klucze

  wczytywanie kluczy

Dodatkowe Dziedziczone Składowe

4.22.1 Opis szczegółowy

Modeluje tablice asocjacyjna przeznaczona do testowania szybkosci wyszukiwnaia.

Definicja w linii 265 pliku algorytm.hh.

4.22.2 Dokumentacja konstruktora i destruktora

4.22.2.1 tab_aso::tab_aso (ifstream & *plik1,* ifstream & *plik2,* ifstream & *plik3,* int *N,* int *M*) [inline]

konstruktor

Definicja w linii 278 pliku algorytm.hh.

4.22.3 Dokumentacja funkcji składowych

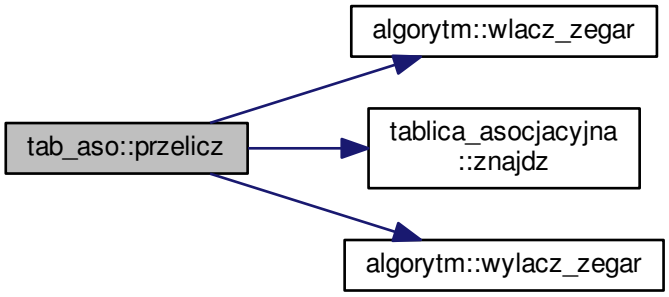
4.22.3.1 float tab_aso::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 250 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.22.3.2 void tab_aso::wczytaj_klucze (ifstream & *plik*)

wczytywanie kluczy

Parametry

in	*plik*	- strumien z kluczmai uzytymi podczas testow

Definicja w linii 241 pliku algorytm.cpp.

4.22.4 Dokumentacja atrybutów składowych

4.22.4.1 tablica_asocjacyjna<float> tab_aso::d [private]

tablica asocjacyjna

Definicja w linii 267 pliku algorytm.hh.

4.22.4.2 string* tab_aso::klucze [private]

wczytywanie kluczy

Parametry

in	*plik*	- strumien z kluczmai uzytymi podczas testow

Definicja w linii 271 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.23 Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjna.

`#include <tablica_asocjacyjna.hh>`

Metody publiczne

- tablica_asocjacyjna ()

  Konstruktor klasy; ustawia nastepujace parametry.
- void dodaj (string k, TYP v)

  Metoda dodaje element do struktury. Gdy (uwzgledniajac porzadek alfabetyczny) element ma stac w skrajnym miejscu tablicy, dodawany jest od razu. W przeciwnym razie funkcja `wstaw` `szuka` odpowiedniego miejsca. Ponadto metoda tworzy pamiec dla struktury, gdy uprzednio jest ona pusta.
- void usun (string k)

  Metoda usuwa zadany element, korzystajac z funkcji znajdz.
- TYP pobierz (string k)

  Metoda zwraca uzytkownikowi szukany element, pod warunkiem, ze jest on w zbiorze.
- bool znajdz (string k)
- bool czy_pusta ()
- int zlicz_elementy ()
- void wypisz ()
- TYP wez (int ind)

  Metoda wprost odnosi sie do konkretnego elementu tablicy - metoda uzywana przy grafie.
- string wez_id (int ind)

  metoda wprost odnosi sie do klucza, konkretnego leemnetu tablicy - metoda uzywana przy grafie

- bool czy_blokada ()

  metoda sprawdza, czy dostep do tablicy jest mozliwy
- void zablokuj ()

  metoda blokuje dostep do tablicy
- void odblokuj ()

  metoda zezwala na dostep do tablicy
- void ustaw (string k, TYP v)

  metoda zmienia wartosc w miejscu, ktore wskazuje klucz k

Metody prywatne

- void insert (int ind, string k, TYP v)

  Metoda ktora umieszcza wartosc oraz jej klucz w zadanym mejscu. Gdy wartosc z kluczem jest dodawana w srodek struktury, dane na prawo od niej przesuwane sa o jeden w prawo. Gdy istnieje potrzeba powiekszenia tablicy, stosuje sie znany juz radzaj gospodarowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.
- void wstaw (string k, TYP v, int ind_l, int ind_r)

  Metoda szuka pozycji, w ktora nalezy dodac element, aby tablica byla posortowana alfabetycznie.
- int znajdz (string k, int ind_l, int ind_r)

  Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, tzn jest zawarty w strukturze, flaga found ustawiana jest na wartosc true.

Atrybuty prywatne

- string * key

  Tablica zawierajaca klucze poszukiwan.
- TYP * value

  Tablica zawierajaca wartosci.
- int s

  rozmiar tablicy
- int sp

  rozmiar danych zapelniajacych tablice
- bool found

  flaga informujaca o tym, czy dany klucz znaleziono w zbiorze
- bool blok

4.23.1 Opis szczegółowy

template<typename TYP>class tablica_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjna.

Definicja w linii 18 pliku tablica_asocjacyjna.hh.

4.23.2 Dokumentacja konstruktora i destruktora

4.23.2.1 template<typename TYP> tablica_asocjacyjna< TYP >::tablica_asocjacyjna ( ) [inline]

Konstruktor klasy; ustawia nastepujace parametry.

```
s = 0 newline
sp = 0 newline
found = false
```

Definicja w linii 122 pliku tablica_asocjacyjna.hh.

4.23.3 Dokumentacja funkcji składowych

4.23.3.1 template<typename TYP> bool tablica_asocjacyjna< TYP >::czy_blokada ( ) [inline]

metoda sprawdza, czy dostep do tablicy jest mozliwy

Zwraca

true, gdy tablica zablokowana, false, gdy dostep jest mozliwy

Definicja w linii 218 pliku tablica_asocjacyjna.hh.

4.23.3.2 template<typename TYP> bool tablica_asocjacyjna< TYP >::czy_pusta ( ) [inline]

Zwraca

true, gdy stos jest pusty, false w przeciwnym wypadku

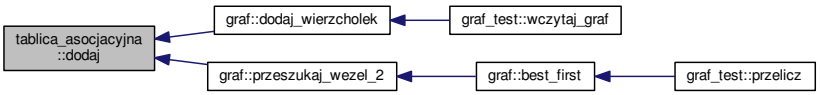
Definicja w linii 186 pliku tablica_asocjacyjna.hh.

4.23.3.3 template<typename TYP> void tablica_asocjacyjna< TYP >::dodaj (string *k,* TYP *v*) [inline]

Metoda dodaje element do struktury. Gdy (uwzgledniajac porzadek alfabetyczny) element ma stac w skrajnym miejscu tablicy, dodawany jest od razu. W przeciwnym razie funkcja `wstaw szuka` odpowiedniego miejsca. Ponadto metoda tworzy pamiec dla struktury, gdy uprzednio jest ona pusta.

Definicja w linii 128 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.4 template<typename TYP> void tablica_asocjacyjna< TYP >::insert (int *ind,* string *k,* TYP *v*) [inline], [private]

Metoda ktora umieszcza wartosc oraz jej klucz w zadanym mejscu. Gdy wartosc z kluczem jest dodawana w srodek struktury, dane na prawo od niej przesuwane sa o jeden w prawo. Gdy istnieje potrzeba powiekszenia tablicy, stosuje sie znany juz radzaj gospodarowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.

Definicja w linii 35 pliku tablica_asocjacyjna.hh.

4.23.3.5 template<typename TYP> void tablica_asocjacyjna< TYP >::odblokuj ( ) [inline]

metoda zezwala na dostep do tablicy

Definicja w linii 226 pliku tablica_asocjacyjna.hh.

4.23.3.6 template<typename TYP> TYP tablica_asocjacyjna< TYP >::pobierz (string *k*) [inline]

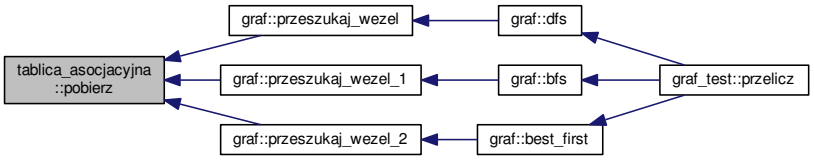
Metoda zwraca uzytkownikowi szukany element, pod warunkiem, ze jest on w zbiorze.

Zwraca

szukany element Gdy slownik jest pusty lub szukany element nie istnieje, uzytkownik zostaje o tym poinformowany

Definicja w linii 168 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.7 template<typename TYP> void tablica_asocjacyjna< TYP >::ustaw (string *k*, TYP *v*) [inline]

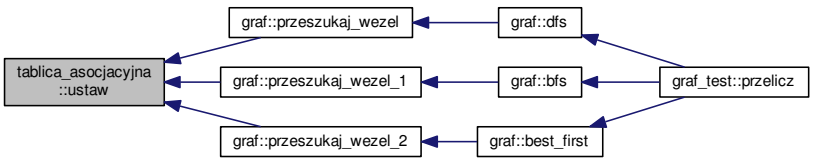
metoda zmienia wartosc w miejscu, ktore wskazuje klucz k

Parametry

in	*k*	- klucz
in	*v*	wartosc, ktora ma zastapic dotychczasowa wartosc w tablicy

Definicja w linii 233 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.8 template<typename TYP> void tablica_asocjacyjna< TYP >::usun (string *k*) [inline]

Metoda usuwa zadany element, korzystajac z funkcji znajdz.

Definicja w linii 143 pliku tablica_asocjacyjna.hh.

4.23.3.9 template<typename TYP> TYP tablica_asocjacyjna< TYP >::wez (int *ind*) [inline]

Metoda wprost odnosi sie do konkretnego elementu tablicy - metoda uzywana przy grafie.

Parametry

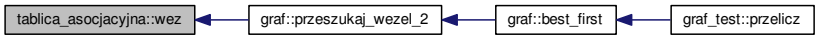
in	*ind*	- indeks tablicy

Zwraca

value[ind] - wartosc mieszczaca sie pod zadanym indeksem

Definicja w linii 202 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.10 template<typename TYP> string tablica_asocjacyjna< TYP >::wez_id (int *ind*) [inline]

metoda wprost odnosi sie do klucza, konkretnego leemnetu tablicy - metoda uzywana przy grafie

Parametry

in	*ind*	- indeks tablicy

Zwraca

key[ind] - klucz mieszczacy sie pod zadanym indeksem

Definicja w linii 210 pliku tablica_asocjacyjna.hh.

4.23.3.11 template<typename TYP> void tablica_asocjacyjna< TYP >::wstaw (string *k,* TYP *v,* int *ind_l,* int *ind_r*) [inline], [private]

Metoda szuka pozycji, w ktora nalezy dodac element, aby tablica byla posortowana alfabetycznie.

Definicja w linii 83 pliku tablica_asocjacyjna.hh.

4.23.3.12 template<typename TYP> void tablica_asocjacyjna< TYP >::wypisz ( ) [inline]

Definicja w linii 193 pliku tablica_asocjacyjna.hh.

4.23.3.13 template<typename TYP> void tablica_asocjacyjna< TYP >::zablokuj ( ) [inline]

metoda blokuje dostep do tablicy

Definicja w linii 222 pliku tablica_asocjacyjna.hh.

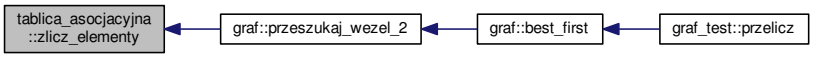
4.23.3.14 template<typename TYP> int tablica_asocjacyjna< TYP >::zlicz_elementy ( ) [inline]

Zwraca

ilosc elementow w strukturze

Definicja w linii 191 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.15 template<typename TYP> int tablica_asocjacyjna< TYP >::znajdz (string *k*, int *ind_l*, int *ind_r*) [inline],[private]

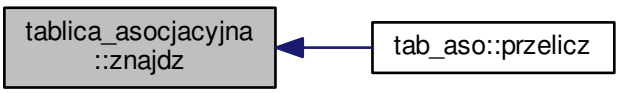
Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, tzn jest zawarty w strukturze, flaga found ustawiana jest na wartosc true.

Zwraca

indeks szukanego elementu

Definicja w linii 100 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.23.3.16 template<typename TYP> bool tablica_asocjacyjna< TYP >::znajdz (string *k*) [inline]

Definicja w linii 178 pliku tablica_asocjacyjna.hh.

4.23.4 Dokumentacja atrybutów składowych

4.23.4.1 template<typename TYP> bool tablica_asocjacyjna< TYP >::blok [private]

Definicja w linii 29 pliku tablica_asocjacyjna.hh.

4.23.4.2 template<typename TYP> bool tablica_asocjacyjna< TYP >::found [private]

flaga informujaca o tym, czy dany klucz znaleziono w zbiorze

Definicja w linii 28 pliku tablica_asocjacyjna.hh.

4.23.4.3 template<typename TYP> string* tablica_asocjacyjna< TYP >::key [private]

Tablica zawierajaca klucze poszukiwan.

Definicja w linii 20 pliku tablica_asocjacyjna.hh.

4.23.4.4 template<typename TYP> int tablica_asocjacyjna< TYP >::s [private]

rozmiar tablicy

Definicja w linii 24 pliku tablica_asocjacyjna.hh.

4.23.4.5 template<typename TYP> int tablica_asocjacyjna< TYP >::sp [private]

rozmiar danych zapelniajacych tablice

Definicja w linii 26 pliku tablica_asocjacyjna.hh.

4.23.4.6 template<typename TYP> TYP* tablica_asocjacyjna< TYP >::value [private]

Tablica zawierajaca wartosci.

Definicja w linii 22 pliku tablica_asocjacyjna.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

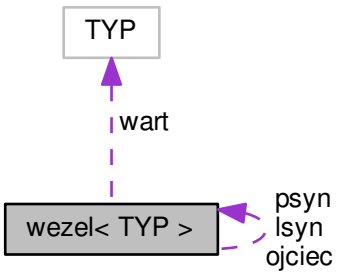
- tablica_asocjacyjna.hh

4.24 Dokumentacja szablonu klasy wezel< TYP >

modeluje pojedynczy wezel drzewa

`#include <drzewo.hh>`

Diagram współpracy dla wezel< TYP >:



Metody publiczne

- wezel ()

  konstruktor bezparametryczny

- wezel (string k, TYP v)
  konstruktor parametryczny
- ~wezel ()
  destruktor
- string wez_klucz ()
- TYP wez_wart ()
- void dodaj_syna (wezel *w)
  dodaje syna do danego wezla
- wezel< TYP > * znajdz_nast ()

Atrybuty publiczne

- syn flag
  okresla, czyim synem jest wezel
- wezel * ojciec
  wskaznik na ojca danego wezla
- wezel * lsyn
  wskaznik na lewego syna wezla
- wezel * psyn
  wskaznik na prawego syna wezla

Atrybuty prywatne

- string klucz
  klucz sluzacy do wyszukiwania
- TYP wart
  wartosc wezla

4.24.1 Opis szczegółowy

template<typename TYP>class wezel< TYP >

modeluje pojedynczy wezel drzewa

Definicja w linii 14 pliku drzewo.hh.

4.24.2 Dokumentacja konstruktora i destruktora

4.24.2.1 template<typename TYP> wezel< TYP >::wezel ( ) [inline]

konstruktor bezparametryczny

Definicja w linii 29 pliku drzewo.hh.

4.24.2.2 template<typename TYP> wezel< TYP >::wezel (string *k*, TYP *v*) [inline]

konstruktor parametryczny

Parametry

in	*k*	- klucz
in	*v*	- wartosc

Definicja w linii 35 pliku drzewo.hh.

4.24.2.3 template<typename TYP> wezel< TYP >::~wezel ( ) [inline]

destruktor

Definicja w linii 37 pliku drzewo.hh.

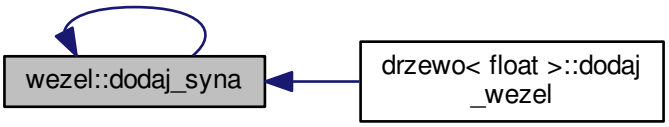
4.24.3 Dokumentacja funkcji składowych

4.24.3.1 template<typename TYP> void wezel< TYP >::dodaj_syna (wezel< TYP > * *w*) [inline]

dodaje syna do danego wezla

Definicja w linii 49 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



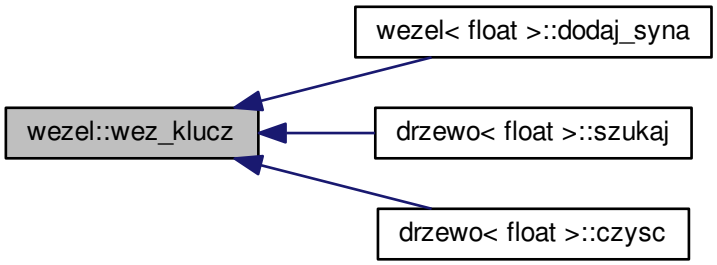
4.24.3.2 template<typename TYP> string wezel< TYP >::wez_klucz ( ) [inline]

Zwraca

klucz wezla

Definicja w linii 41 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



4.24.3.3 template<typename TYP> TYP wezel< TYP >::wez_wart ( ) [inline]

Zwraca

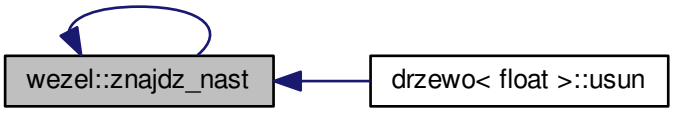
wartosc wezla

Definicja w linii 45 pliku drzewo.hh.

4.24.3.4 template<typename TYP> wezel<TYP>* wezel< TYP >::znajdz_nast ( ) [inline]

Definicja w linii 61 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



4.24.4 Dokumentacja atrybutów składowych

4.24.4.1 template<typename TYP> syn wezel< TYP >::flag

okresla, czyim synem jest wezel

Definicja w linii 21 pliku drzewo.hh.

4.24.4.2 template<typename TYP> string wezel< TYP >::klucz [private]

klucz sluzacy do wyszukiwania

Definicja w linii 16 pliku drzewo.hh.

4.24.4.3 template<typename TYP> wezel* wezel< TYP >::lsyn

wskaznik na lewego syna wezla

Definicja w linii 25 pliku drzewo.hh.

4.24.4.4 template<typename TYP> wezel* wezel< TYP >::ojciec

wskaznik na ojca danego wezla

Definicja w linii 23 pliku drzewo.hh.

4.24.4.5 template<typename TYP> wezel* wezel< TYP >::psyn

wskaznik na prawego syna wezla

Definicja w linii 27 pliku drzewo.hh.

4.24.4.6 template<typename TYP> TYP wezel< TYP >::wart [private]

wartosc wezla

Definicja w linii 18 pliku drzewo.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- drzewo.hh

4.25 Dokumentacja klasy wierzcholek

Klasa modeluje pojecie wierzcholka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakie interpertowanie wierzcholka grafu, wedle zyczen uzytkownika.

`#include <graf.hh>`

Metody publiczne

- wierzcholek (int wz, int wg)

  konstruktor

Atrybuty prywatne

- int id

  numer indentyfikacyjny wierzcholka
- int waga

  waga wezla, uzywana w stosunku do wierzcholka, z ktorym ow wierzcholek jest incydentny

Przyjaciele

- class graf

  klasa zparzyjazniona

4.25.1 Opis szczegółowy

Klasa modeluje pojecie wierzcholka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakie interpertowanie wierzcholka grafu, wedle zyczen uzytkownika.

Definicja w linii 17 pliku graf.hh.

4.25.2 Dokumentacja konstruktora i destruktora

4.25.2.1 wierzcholek::wierzcholek (int *wz,* int *wg*) [inline]

konstruktor

Parametry

in	*wz*	- id wierzcholka
in	*wg*	- waga

Definicja w linii 30 pliku graf.hh.

4.25.3 Dokumentacja przyjaciół i funkcji związanych

4.25.3.1 friend class graf [friend]

klasa zparzyjazniona

Definicja w linii 19 pliku graf.hh.

4.25.4 Dokumentacja atrybutów składowych

4.25.4.1 int wierzcholek::id [private]

numer indentyfikacyjny wierzcholka

Definicja w linii 22 pliku graf.hh.

4.25.4.2 int wierzcholek::waga [private]

waga wezla, uzywana w stosunku do wierzcholka, z ktorym ow wierzcholek jest incydentny

Definicja w linii 24 pliku graf.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- graf.hh

Rozdział 5

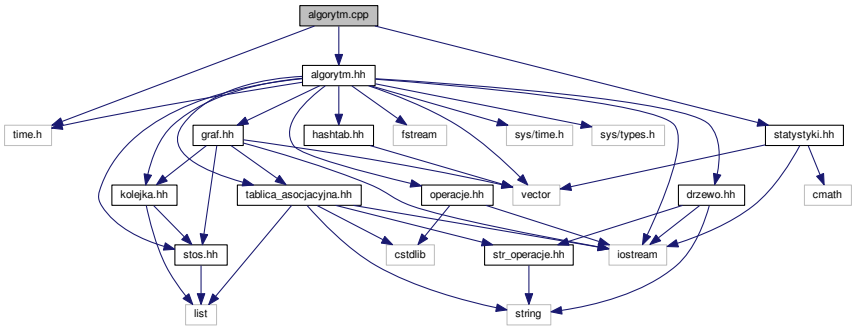
Dokumentacja plików

5.1 Dokumentacja pliku algorytm.cpp

plik zawiera definicje metod klas zdefiniowanych w pliku algorytm.hh

```
#include "algorytm.hh"
#include "statystyki.hh"
#include <time.h>
```

Wykres zależności załączania dla algorytm.cpp:



5.1.1 Opis szczegółowy

plik zawiera definicje metod klas zdefiniowanych w pliku algorytm.hh

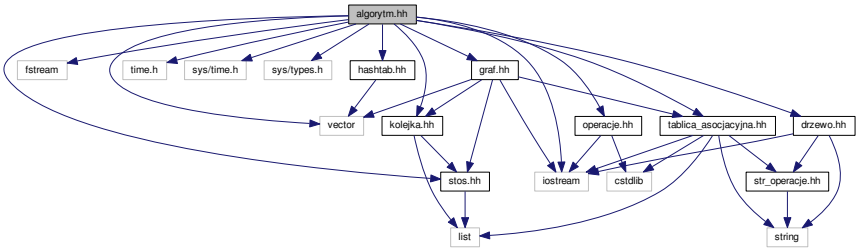
Definicja w pliku algorytm.cpp.

5.2 Dokumentacja pliku algorytm.hh

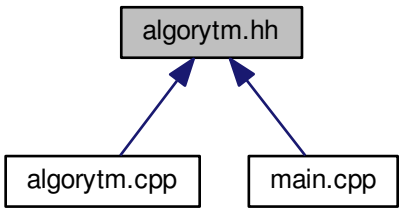
Definicja klas wykonujacych operacje na zestawie danych wejsciowych.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <time.h>
#include <sys/time.h>
#include <sys/types.h>
#include "operacje.hh"
#include "stos.hh"
#include "kolejka.hh"
#include "drzewo.hh"
#include "hashtab.hh"
#include "tablica_asocjacyjna.hh"
#include "graf.hh"
```

Wykres zależności załączania dla algorytm.hh:



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



Komponenty

- class algorytm

  Definicja klasy algorytm Jest to klasa bazowa, ktora ma za zadanie wczytac, przetworzyc i porownac dane z plikiem wzorcowym.
- class mnozenie

  modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2
- class stos_tablica

  klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury
- class stos_lista

  klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury
- class kolejka_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

- class kolejka_lista

  klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

- class q_sort

  klasa reprezentuje dane poddane sortowaniu szybkiemu

- class h_sort

  klasa reprezentuje dane poddane sortowaniu przez kopcowanie

- class m_sort

  klasa reprezentuje dane poddane sortowaniu przez scalanie

- class bst

  Modeluje drzewo binarne przeznaczone do testowania szybkosci wyszukiwnaia.

- class h_table

  Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwnaia.

- class tab_aso

  Modeluje tablice asocjacyjna przeznaczona do testowania szybkosci wyszukiwnaia.

- class graf_test

  modeluje strukture grafów uzytych do badan

5.2.1 Opis szczegółowy

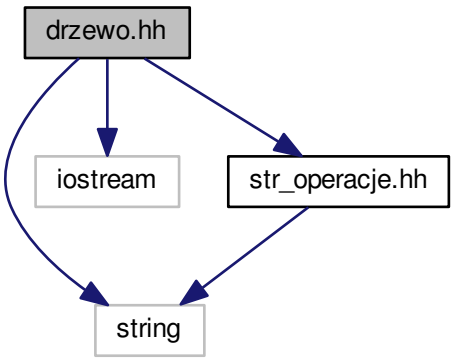
Definicja klas wykonujacych operacje na zestawie danych wejsciowych.

Definicja w pliku algorytm.hh.

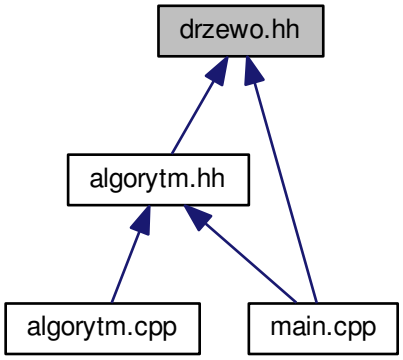
5.3 Dokumentacja pliku drzewo.hh

```
#include <string>
#include <iostream>
#include "str_operacje.hh"
```

Wykres zależności załączania dla drzewo.hh:



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



Komponenty

- class wezel< TYP >

  modeluje pojedynczy wezel drzewa
- class drzewo< TYP >

  modeluje binarne drzewo przeszukiwan

Wyliczenia

- enum syn { lewy, zaden, prawy }

  typ wyliczeniowy, określa, czyim synem jest dany element drzewa

5.3.1 Opis szczegółowy

Plik zawiera definicje klasy reprezentujacej drzewo binarne

Definicja w pliku drzewo.hh.

5.3.2 Dokumentacja typów wyliczanych

5.3.2.1 enum syn

typ wyliczeniowy, określa, czyim synem jest dany element drzewa

Wartości wyliczeń

lewy

zaden

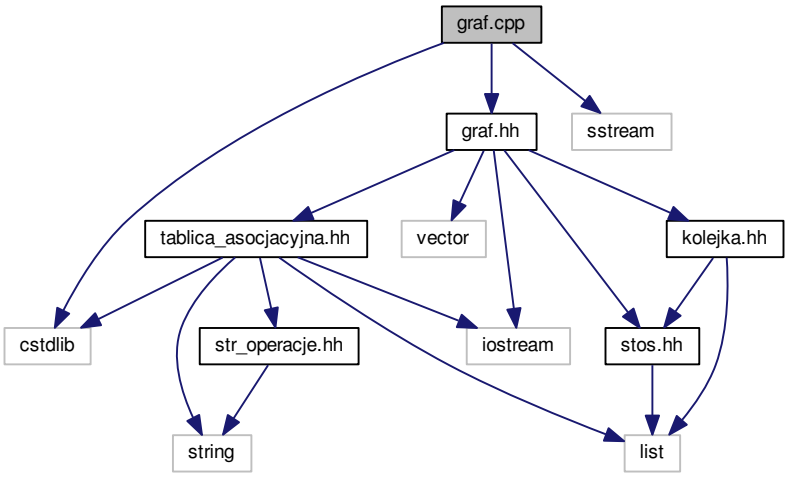
prawy

Definicja w linii 11 pliku drzewo.hh.

5.4 Dokumentacja pliku graf.cpp

```
#include "graf.hh"
#include <sstream>
#include <cstdlib>
```

Wykres zależności załączania dla graf.cpp:



Zmienne

- tablica_asocjacyjna< int > vec

5.4.1 Dokumentacja zmiennych

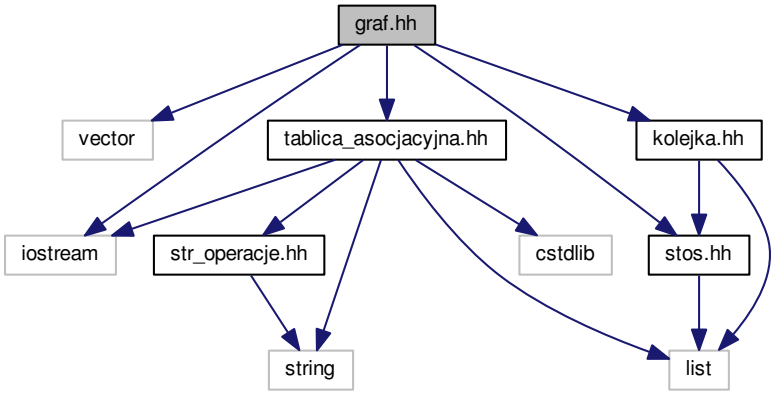
5.4.1.1 tablica_asocjacyjna<int> vec

Definicja w linii 4 pliku graf.cpp.

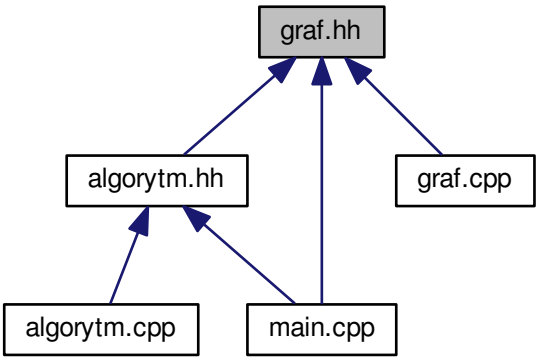
5.5 Dokumentacja pliku graf.hh

```
#include <vector>
#include <iostream>
#include "tablica_asocjacyjna.hh"
#include "stos.hh"
#include "kolejka.hh"
```

Wykres zależności załączania dla graf.hh:



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



Komponenty

- class wierzcholek

  Klasa modeluje pojecie wierzcholka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakie interpertowanie wierzcholka grafu, wedle zyczen uzytkownika.

- class graf

  Klasa modeluje pojecie grafu w oparciu o liste incydencji, Operacje na grafie mozliwe sa na dwa sposoby \n.

5.5.1 Opis szczegółowy

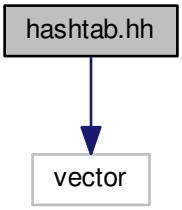
Plik zawiera definicje klasy wierzcholek i klasy graf

Definicja w pliku graf.hh.

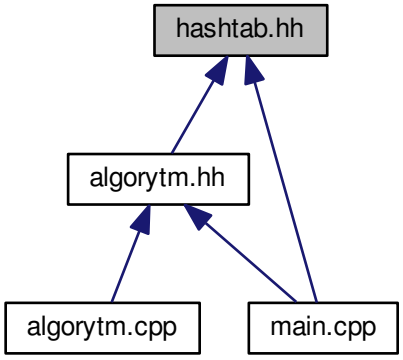
5.6 Dokumentacja pliku hashtab.hh

```
#include <vector>
```

Wykres zależności załączania dla hashtab.hh:



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



Komponenty

- class el_tab< TYP >

  pojedynczy element tablicy haszujacej
- class hashtab< TYP >

  modeluje tablice haszujca w oparciu o kontener klasy el_tab

5.6.1 Opis szczegółowy

Plik zawiera definicje klasy reprezentujacej tablice haszujaca

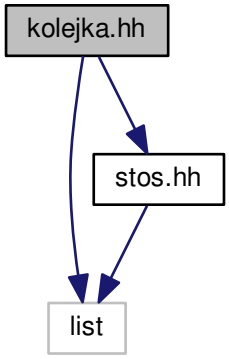
Definicja w pliku hashtab.hh.

5.7 Dokumentacja pliku kolejka.hh

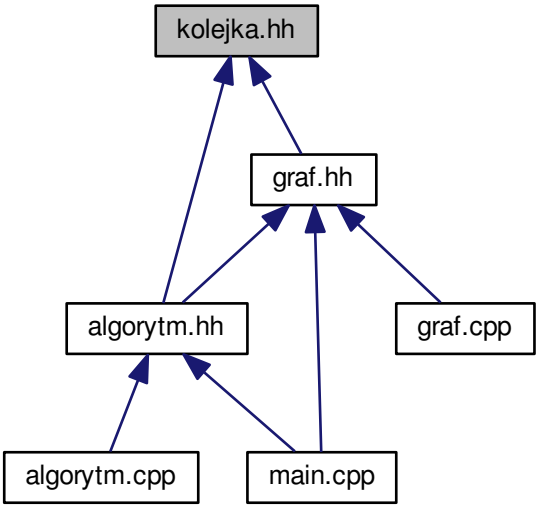
Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby.

```
#include <list>
#include "stos.hh"
```

Wykres zależności załączania dla kolejka.hh:



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



Komponenty

- class queue_list< TYP >

Modeluje kolejke oparta na liscie STL.

- class queue_array< TYP >

  Modeluje kolejke w oparciu o tablice.

5.7.1 Opis szczegółowy

Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby.

1. Za pomocą listy.
2. Za pomocą tablicy a. kazdorazowo powiekszajacej swoj rozmiar b. powiekszajacej swoj rozmiar dwukrotnie, gdy kolejka sie przepelni

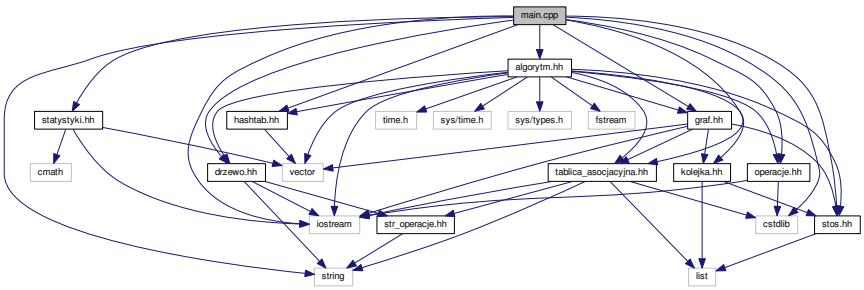
Definicja w pliku kolejka.hh.

5.8 Dokumentacja pliku main.cpp

plik glowny

```
#include <iostream>
#include "algorytm.hh"
#include "statystyki.hh"
#include "operacje.hh"
#include "stos.hh"
#include "tablica_asocjacyjna.hh"
#include "drzewo.hh"
#include "hashtab.hh"
#include "graf.hh"
#include <cstdlib>
#include <string>
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main ()

5.8.1 Opis szczegółowy

plik glowny

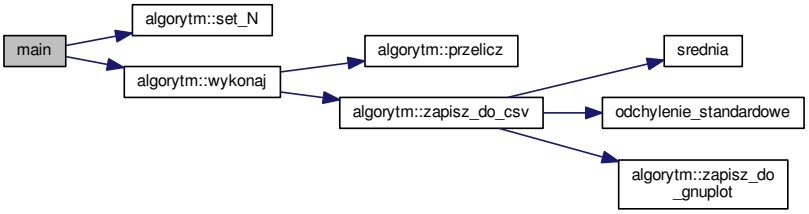
Definicja w pliku main.cpp.

5.8.2 Dokumentacja funkcji

5.8.2.1 int main ( )

Definicja w linii 20 pliku main.cpp.

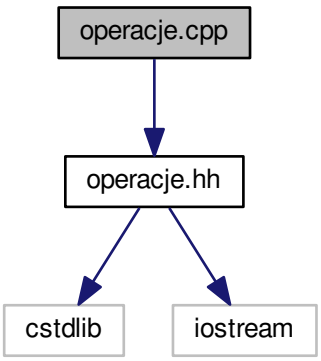
Oto graf wywołań dla tej funkcji:



5.9 Dokumentacja pliku operacje.cpp

```
#include "operacje.hh"
```

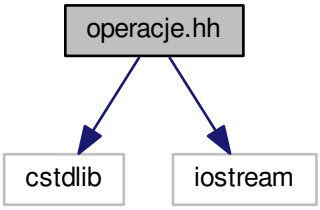
Wykres zależności załączania dla operacje.cpp:



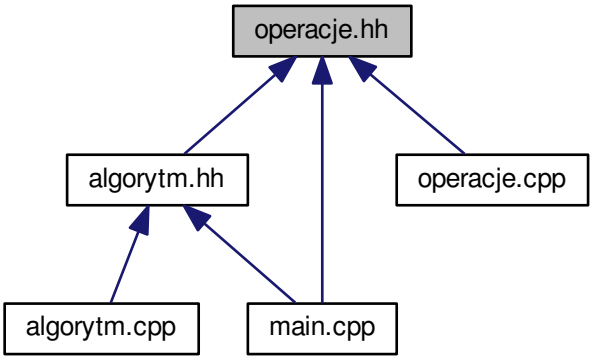
5.10 Dokumentacja pliku operacje.hh

```
#include <cstdlib>
#include <iostream>
```

Wykres zależności załączania dla operacje.hh:



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



Komponenty

- class operacje

  Klasa modeluje tablice z danymi i metody sluzace do operacji na niej.

Definicje

- #define ROZMIAR 9

5.10.1 Dokumentacja definicji

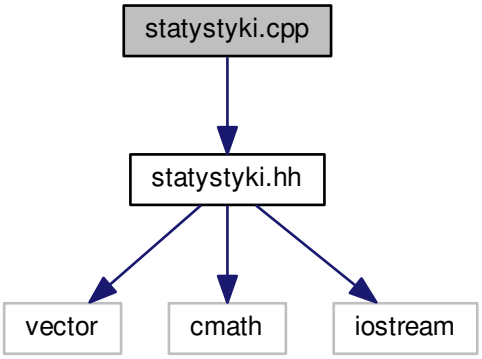
5.10.1.1 #define ROZMIAR 9

Definicja w linii 3 pliku operacje.hh.

5.11 Dokumentacja pliku statystyki.cpp

```
#include "statystyki.hh"
```

Wykres zależności załączania dla statystyki.cpp:



Funkcje

- float srednia (float ∗tab, int rozmiar)

  funckja oblicza wartosc srednia
- float odchylenie_standardowe (float srednia, float ∗tab, int rozmiar)

  funckja oblicza odchylenie standardowe

5.11.1 Dokumentacja funkcji

5.11.1.1 float odchylenie_standardowe (float *srednia,* float ∗ *tab,* int *rozmiar*)

funckja oblicza odchylenie standardowe

Parametry

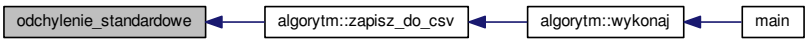
tab	- kontener zawierajacy czasy wykonania algorytmu
srednia	- wartosc srednia
rozmiar	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.11.1.2 float srednia (float * *tab*, int *rozmiar*)

funckja oblicza wartosc srednia

Parametry

tab	- kontener zawierajacy czasy wykonania algorytmu
rozmiar	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

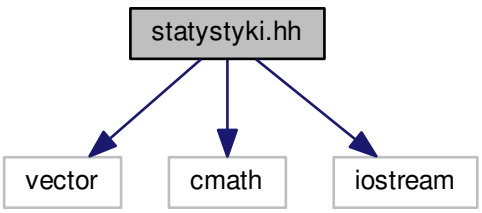
Oto graf wywoływań tej funkcji:

5.12 Dokumentacja pliku statystyki.hh

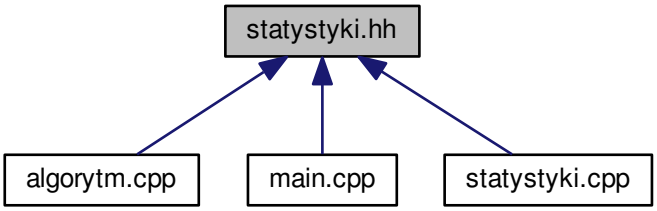
plik zawiera dekalracje funkcji odpowiedzialnych za przeprowadznaie statystyk

```
#include <vector>
#include <cmath>
#include <iostream>
```

Wykres zależności załączania dla statystyki.hh:



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



Funkcje

- float srednia (float ∗tab, int rozmiar)

  funckja oblicza wartosc srednia
- float odchylenie_standardowe (float srednia, float ∗tab, int rozmiar)

  funckja oblicza odchylenie standardowe

5.12.1 Opis szczegółowy

plik zawiera dekalracje funkcji odpowiedzialnych za przeprowadznaie statystyk

Definicja w pliku statystyki.hh.

5.12.2 Dokumentacja funkcji

5.12.2.1 float odchylenie_standardowe (float *srednia,* float ∗ *tab,* int *rozmiar*)

funckja oblicza odchylenie standardowe

Parametry

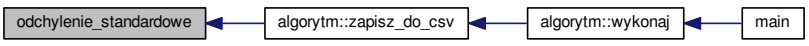
tab	- kontener zawierajacy czasy wykonania algorytmu
srednia	- wartosc srednia
rozmiar	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.12.2.2 float srednia (float * *tab,* int *rozmiar*)

funckja oblicza wartosc srednia

Parametry

tab	- kontener zawierajacy czasy wykonania algorytmu
rozmiar	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

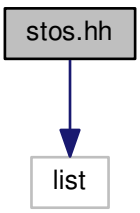
Oto graf wywoływań tej funkcji:

5.13 Dokumentacja pliku stos.hh

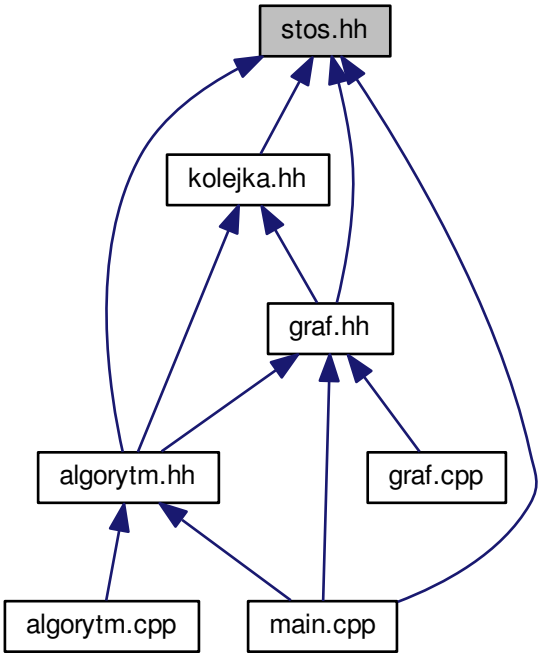
Plik zawiera definicje klasy `Stos` Zaimplementowana na 2 sposoby.

```
#include <list>
```

Wykres zależności załączania dla stos.hh:



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



Komponenty

- class stack_list< TYP >

  Modeluje stos oparty na liscie STL.
- class stack_array< TYP >

  Modeluje stos w oparciu o tablice.

Wyliczenia

- enum flag { plus1, x2 }

  typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

5.13.1 Opis szczegółowy

Plik zawiera definicje klasy `Stos` Zaimplementowana na 2 sposoby.

1. Za pomocą listy.
2. Za pomocą tablicy a. kazdorazowo powiekszajacej swoj rozmiar b. powiekszajacej swoj rozmiar dwukrotnie, gdy stos sie przepelni

Definicja w pliku stos.hh.

5.13.2 Dokumentacja typów wyliczanych

5.13.2.1 enum flag

typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

Wartości wyliczeń

plus1

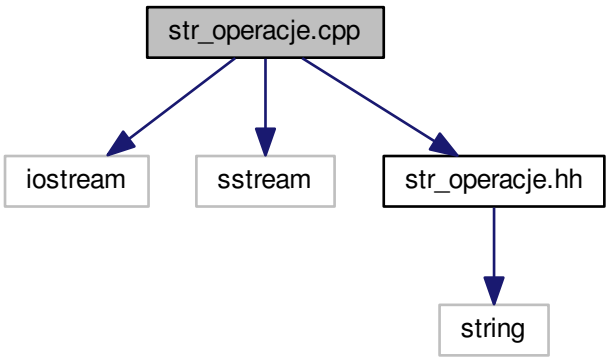
x2

Definicja w linii 17 pliku stos.hh.

5.14 Dokumentacja pliku str_operacje.cpp

```
#include <iostream>
#include <sstream>
#include "str_operacje.hh"
```

Wykres zależności załączania dla str_operacje.cpp:



Funkcje

- bool operator< (string s1, string s2)

  funkcja sluzaca do alfabetycznego porzadkowania napisow
- bool operator> (string s1, string s2)

  funkcja sluzaca do alfabetycznego porzadkowania napisow
- bool operator<= (string s1, string s2)
- bool operator>= (string s1, string s2)
- bool operator== (string s1, string s2)

5.14.1 Dokumentacja funkcji

5.14.1.1 bool operator< (string *s1*, string *s2*)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

true, gdy s1 wyzej w porzadku alfabetycznym niz s2, false w przeciwnym przypadku

Definicja w linii 7 pliku str_operacje.cpp.

5.14.1.2 bool operator<= (string *s1,* string *s2*)

Zwraca

true, gdy s1 jest wyzej w porzadku alfabrtycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s2 jest wyzej w porzadku alfabetycznym niz s1

Definicja w linii 34 pliku str_operacje.cpp.

5.14.1.3 bool operator== (string *s1,* string *s2*)

Zwraca

true, gdy łańcuchy są identyczne

Definicja w linii 42 pliku str_operacje.cpp.

5.14.1.4 bool operator> (string *s1,* string *s2*)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

true, gdy s1 nizej w porzadku alfabetycznym niz s2, false w przeciwnym przypadku

Definicja w linii 21 pliku str_operacje.cpp.

5.14.1.5 bool operator>= (string *s1,* string *s2*)

Zwraca

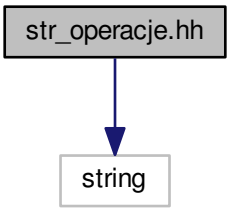
true, gdy s1 jest nizej w porzadku alfabrtycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s1 jest wyzej w porzadku alfabetycznym niz s2

Definicja w linii 38 pliku str_operacje.cpp.

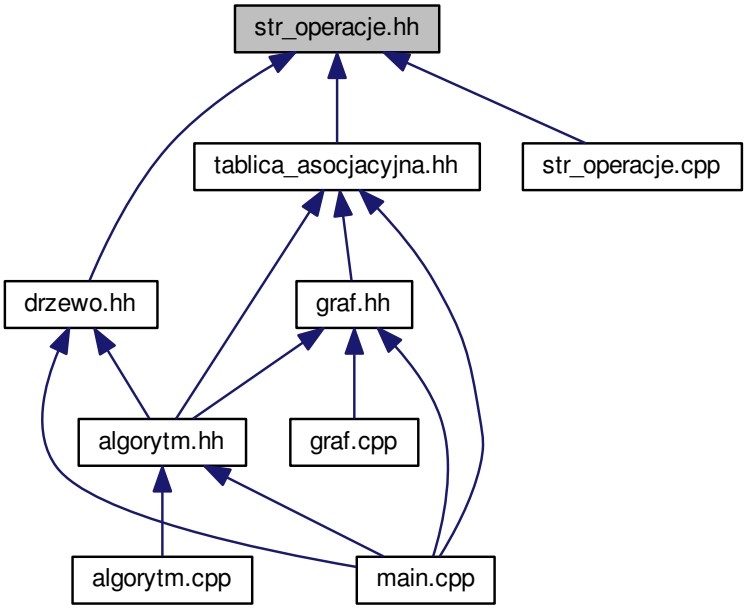
5.15 Dokumentacja pliku str_operacje.hh

```
#include <string>
```

Wykres zależności załączania dla str_operacje.hh:



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



Funkcje

- bool operator< (string s1, string s2)

  funkcja sluzaca do alfabetycznego porzadkowania napisow
- bool operator> (string s1, string s2)

  funkcja sluzaca do alfabetycznego porzadkowania napisow
- bool operator<= (string s1, string s2)
- bool operator>= (string s1, string s2)
- bool operator== (string s1, string s2)

5.15.1 Dokumentacja funkcji

5.15.1.1 bool operator< (string *s1*, string *s2*)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

true, gdy s1 wyzej w porzadku alfabetycznym niz s2, false w przeciwnym przypadku

Definicja w linii 7 pliku str_operacje.cpp.

5.15.1.2 bool operator<= (string *s1*, string *s2*)

Zwraca

true, gdy s1 jest wyzej w porzadku alfabrtycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s2 jest wyzej w porzadku alfabetycznym niz s1

Definicja w linii 34 pliku str_operacje.cpp.

5.15.1.3 bool operator== (string *s1*, string *s2*)

Zwraca

true, gdy łańcuchy są identyczne

Definicja w linii 42 pliku str_operacje.cpp.

5.15.1.4 bool operator> (string *s1*, string *s2*)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

true, gdy s1 nizej w porzadku alfabetycznym niz s2, false w przeciwnym przypadku

Definicja w linii 21 pliku str_operacje.cpp.

5.15.1.5 bool operator>= (string *s1*, string *s2*)

Zwraca

true, gdy s1 jest nizej w porzadku alfabrtycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s1 jest wyzej w porzadku alfabetycznym niz s2

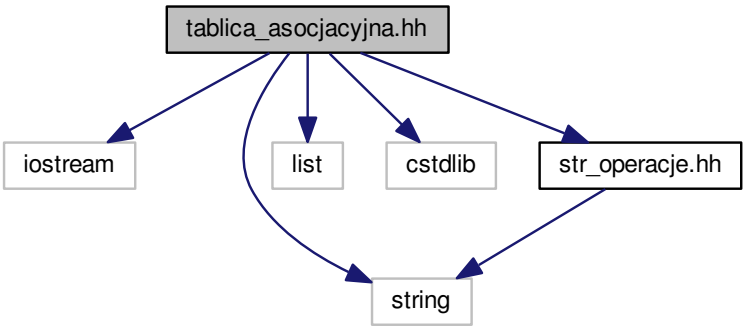
Definicja w linii 38 pliku str_operacje.cpp.

5.16 Dokumentacja pliku strona.dox

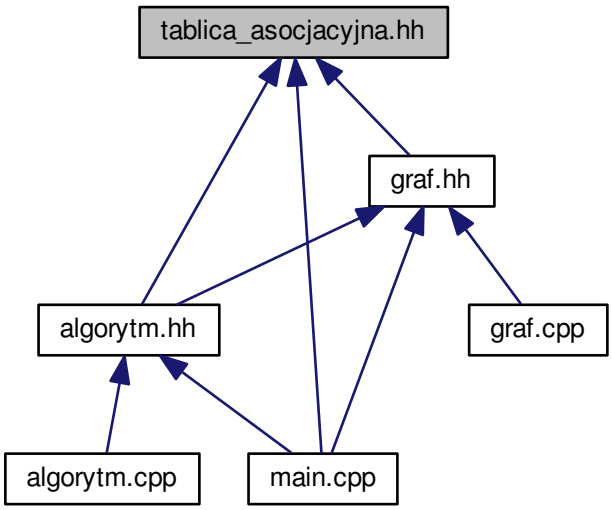
5.17 Dokumentacja pliku tablica_asocjacyjna.hh

```
#include <iostream>
#include <string>
#include <list>
#include <cstdlib>
#include "str_operacje.hh"
```

Wykres zależności załączania dla tablica_asocjacyjna.hh:



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



Komponenty

- class tablica_asocjacyjna< TYP >

  Klasa modeluje tablice asocjacyjna.

5.17.1 Opis szczegółowy

Plik zawiera definicje klasy tablica_asocjacyjna, oraz definicje funkcji pomocniczych jako przeciazen operatorow porownania dla klasy typu string

Definicja w pliku tablica_asocjacyjna.hh.

Skorowidz