

PAMSI - pwilkosz
1.0

Wygenerowano przez Doxygen 1.8.6

Wt, 6 maj 2014 23:38:56

Spis treści

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy algorytm	7
4.1.1 Opis szczegółowy	10
4.1.2 Dokumentacja konstruktora i destruktora	10
4.1.2.1 algorytm	10
4.1.3 Dokumentacja funkcji składowych	10
4.1.3.1 ile_danych	10
4.1.3.2 jaki_czas	10
4.1.3.3 porownaj	11
4.1.3.4 przelicz	11
4.1.3.5 set_N	11
4.1.3.6 wczytaj	11
4.1.3.7 wczytaj_wzor	12
4.1.3.8 wlacz_zegar	12
4.1.3.9 wykonaj	13
4.1.3.10 wylacz_zegar	14
4.1.3.11 zapisz_do_csv	15
4.1.3.12 zapisz_do_gnuplot	16
4.1.4 Dokumentacja atrybutów składowych	16
4.1.4.1 czas	16
4.1.4.2 czas1	16
4.1.4.3 czas2	17
4.1.4.4 dane	17

4.1.4.5	dane_wz	17
4.1.4.6	m	17
4.1.4.7	n	17
4.1.4.8	op	17
4.2	Dokumentacja klasy bst	17
4.2.1	Opis szczegółowy	19
4.2.2	Dokumentacja konstruktora i destruktora	19
4.2.2.1	bst	19
4.2.2.2	~bst	19
4.2.3	Dokumentacja funkcji składowych	19
4.2.3.1	przelicz	19
4.2.3.2	wczytaj_klucze	20
4.2.4	Dokumentacja atrybutów składowych	20
4.2.4.1	d	20
4.2.4.2	klucze	20
4.3	Dokumentacja szablonu klasy drzewo< TYP >	20
4.3.1	Opis szczegółowy	21
4.3.2	Dokumentacja konstruktora i destruktora	21
4.3.2.1	drzewo	21
4.3.2.2	drzewo	21
4.3.3	Dokumentacja funkcji składowych	21
4.3.3.1	czyszc	21
4.3.3.2	dodaj	22
4.3.3.3	dodaj_wezel	22
4.3.3.4	szukaj	22
4.3.3.5	usun	22
4.3.3.6	wyczyszc	22
4.3.3.7	znajdz	22
4.3.4	Dokumentacja atrybutów składowych	23
4.3.4.1	korzen	23
4.3.4.2	znaleziony	23
4.4	Dokumentacja szablonu klasy el_tab< TYP >	23
4.4.1	Opis szczegółowy	24
4.4.2	Dokumentacja konstruktora i destruktora	24
4.4.2.1	el_tab	24
4.4.2.2	~el_tab	24
4.4.3	Dokumentacja atrybutów składowych	25
4.4.3.1	klucz	25
4.4.3.2	wart	25
4.4.3.3	zajety	25

4.5	Dokumentacja klasy graf	25
4.5.1	Opis szczegółowy	27
4.5.2	Dokumentacja konstruktora i destruktora	27
4.5.2.1	graf	27
4.5.3	Dokumentacja funkcji składowych	27
4.5.3.1	best_first	27
4.5.3.2	bfs	28
4.5.3.3	czy_sasiad	29
4.5.3.4	czy_sasiad	29
4.5.3.5	dfs	29
4.5.3.6	dodaj_krawedz	30
4.5.3.7	dodaj_krawedz	31
4.5.3.8	dodaj_wierzcholek	31
4.5.3.9	dodaj_wierzcholek	32
4.5.3.10	przeszukaj_wezel	32
4.5.3.11	przeszukaj_wezel_1	33
4.5.3.12	przeszukaj_wezel_2	34
4.5.3.13	sasiedztwo	35
4.5.3.14	sasiedztwo	36
4.5.3.15	usun_krawedz	36
4.5.3.16	usun_krawedz	36
4.5.3.17	usun_wierzcholek	37
4.5.3.18	usun_wierzcholek	37
4.5.3.19	wyczysc	38
4.5.3.20	wypisz_liste	38
4.5.4	Dokumentacja atrybutów składowych	38
4.5.4.1	lista_incydencji	38
4.5.4.2	Q	38
4.5.4.3	Q0	38
4.5.4.4	tab	38
4.6	Dokumentacja klasy graf_test	38
4.6.1	Opis szczegółowy	40
4.6.2	Dokumentacja konstruktora i destruktora	40
4.6.2.1	graf_test	40
4.6.3	Dokumentacja funkcji składowych	40
4.6.3.1	przelicz	40
4.6.3.2	wczytaj_graf	41
4.6.4	Dokumentacja atrybutów składowych	41
4.6.4.1	G1	41
4.6.4.2	G2	41

4.6.4.3	G3	42
4.6.4.4	G4	42
4.6.4.5	G5	42
4.6.4.6	G6	42
4.6.4.7	typ	42
4.7	Dokumentacja klasy <code>h_sort</code>	42
4.7.1	Opis szczegółowy	43
4.7.2	Dokumentacja konstruktora i destruktora	43
4.7.2.1	<code>h_sort</code>	43
4.7.3	Dokumentacja funkcji składowych	43
4.7.3.1	<code>przelicz</code>	43
4.8	Dokumentacja klasy <code>h_table</code>	44
4.8.1	Opis szczegółowy	45
4.8.2	Dokumentacja konstruktora i destruktora	45
4.8.2.1	<code>h_table</code>	45
4.8.3	Dokumentacja funkcji składowych	46
4.8.3.1	<code>przelicz</code>	46
4.8.3.2	<code>wczytaj_klucze</code>	46
4.8.4	Dokumentacja atrybutów składowych	46
4.8.4.1	<code>klucze</code>	46
4.9	Dokumentacja szablonu klasy <code>hashtab< TYP ></code>	46
4.9.1	Opis szczegółowy	47
4.9.2	Dokumentacja konstruktora i destruktora	47
4.9.2.1	<code>hashtab</code>	47
4.9.2.2	<code>hashtab</code>	47
4.9.3	Dokumentacja funkcji składowych	48
4.9.3.1	<code>dodaj</code>	48
4.9.3.2	<code>hash</code>	48
4.9.3.3	<code>ustaw_dlugosc</code>	49
4.9.3.4	<code>usun</code>	49
4.9.3.5	<code>wypisz</code>	49
4.9.3.6	<code>znajdz</code>	50
4.9.4	Dokumentacja atrybutów składowych	50
4.9.4.1	<code>dlugosc</code>	50
4.9.4.2	<code>tab</code>	50
4.10	Dokumentacja klasy <code>kolejka_lista</code>	51
4.10.1	Opis szczegółowy	52
4.10.2	Dokumentacja konstruktora i destruktora	52
4.10.2.1	<code>kolejka_lista</code>	52
4.10.3	Dokumentacja funkcji składowych	52

4.10.3.1	przelicz	52
4.10.4	Dokumentacja atrybutów składowych	52
4.10.4.1	qu	52
4.11	Dokumentacja klasy kolejka_tablica	53
4.11.1	Opis szczegółowy	54
4.11.2	Dokumentacja konstruktora i destruktora	54
4.11.2.1	kolejka_tablica	54
4.11.3	Dokumentacja funkcji składowych	54
4.11.3.1	przelicz	54
4.11.4	Dokumentacja atrybutów składowych	54
4.11.4.1	qu	54
4.12	Dokumentacja klasy m_sort	55
4.12.1	Opis szczegółowy	56
4.12.2	Dokumentacja konstruktora i destruktora	56
4.12.2.1	m_sort	56
4.12.3	Dokumentacja funkcji składowych	56
4.12.3.1	przelicz	56
4.13	Dokumentacja klasy mnozenie	56
4.13.1	Opis szczegółowy	57
4.13.2	Dokumentacja konstruktora i destruktora	57
4.13.2.1	mnozenie	58
4.13.3	Dokumentacja funkcji składowych	59
4.13.3.1	przelicz	59
4.14	Dokumentacja klasy operacje	59
4.14.1	Opis szczegółowy	60
4.14.2	Dokumentacja konstruktora i destruktora	60
4.14.2.1	operacje	60
4.14.2.2	operacje	60
4.14.3	Dokumentacja funkcji składowych	61
4.14.3.1	dodaj_element	61
4.14.3.2	dodaj_elementy	62
4.14.3.3	heap_sort	62
4.14.3.4	make_heap	62
4.14.3.5	make_node	63
4.14.3.6	merge	63
4.14.3.7	merge_sort	64
4.14.3.8	odwroc_tablice	64
4.14.3.9	operator=	65
4.14.3.10	operator==	66
4.14.3.11	operator[]	66

4.14.3.12 quick_sort	66
4.14.3.13 zamien_elementy	67
4.14.4 Dokumentacja atrybutów składowych	67
4.14.4.1 n	67
4.14.4.2 tab	67
4.15 Dokumentacja klasy q_sort	67
4.15.1 Opis szczegółowy	68
4.15.2 Dokumentacja konstruktora i destruktora	69
4.15.2.1 q_sort	69
4.15.3 Dokumentacja funkcji składowych	69
4.15.3.1 przelicz	69
4.16 Dokumentacja szablonu klasy queue_array< TYP >	69
4.16.1 Opis szczegółowy	70
4.16.2 Dokumentacja konstruktora i destruktora	71
4.16.2.1 queue_array	71
4.16.2.2 queue_array	71
4.16.3 Dokumentacja funkcji składowych	71
4.16.3.1 clear	71
4.16.3.2 dequeue	71
4.16.3.3 enqueue	71
4.16.3.4 is_empty	72
4.16.3.5 size	72
4.16.4 Dokumentacja atrybutów składowych	72
4.16.4.1 f	72
4.16.4.2 q	72
4.16.4.3 s	72
4.16.4.4 sp	73
4.17 Dokumentacja szablonu klasy queue_list< TYP >	73
4.17.1 Opis szczegółowy	73
4.17.2 Dokumentacja funkcji składowych	73
4.17.2.1 clear	73
4.17.2.2 dequeue	74
4.17.2.3 enqueue	74
4.17.2.4 is_empty	74
4.17.2.5 size	74
4.17.3 Dokumentacja atrybutów składowych	75
4.17.3.1 q	75
4.18 Dokumentacja szablonu klasy stack_array< TYP >	75
4.18.1 Opis szczegółowy	76
4.18.2 Dokumentacja konstruktora i destruktora	76

4.18.2.1	stack_array	76
4.18.2.2	stack_array	76
4.18.3	Dokumentacja funkcji składowych	76
4.18.3.1	clear	76
4.18.3.2	is_empty	76
4.18.3.3	pop	77
4.18.3.4	push	77
4.18.3.5	size	77
4.18.4	Dokumentacja atrybutów składowych	78
4.18.4.1	f	78
4.18.4.2	s	78
4.18.4.3	sp	78
4.18.4.4	st	78
4.19	Dokumentacja szablonu klasy stack_list< TYP >	78
4.19.1	Opis szczegółowy	79
4.19.2	Dokumentacja funkcji składowych	79
4.19.2.1	clear	79
4.19.2.2	is_empty	79
4.19.2.3	pop	79
4.19.2.4	push	79
4.19.2.5	size	80
4.19.3	Dokumentacja atrybutów składowych	80
4.19.3.1	st	80
4.20	Dokumentacja klasy stos_lista	80
4.20.1	Opis szczegółowy	81
4.20.2	Dokumentacja konstruktora i destruktor	81
4.20.2.1	stos_lista	81
4.20.3	Dokumentacja funkcji składowych	81
4.20.3.1	przelicz	81
4.20.4	Dokumentacja atrybutów składowych	82
4.20.4.1	stos	82
4.21	Dokumentacja klasy stos_tablica	82
4.21.1	Opis szczegółowy	83
4.21.2	Dokumentacja konstruktora i destruktor	83
4.21.2.1	stos_tablica	83
4.21.3	Dokumentacja funkcji składowych	83
4.21.3.1	przelicz	84
4.21.4	Dokumentacja atrybutów składowych	84
4.21.4.1	stos	84
4.22	Dokumentacja klasy tab_aso	84

4.22.1	Opis szczegółowy	86
4.22.2	Dokumentacja konstruktora i destruktora	86
4.22.2.1	tab_aso	86
4.22.3	Dokumentacja funkcji składowych	86
4.22.3.1	przelicz	86
4.22.3.2	wczytaj_klucze	86
4.22.4	Dokumentacja atrybutów składowych	87
4.22.4.1	d	87
4.22.4.2	klucze	87
4.23	Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >	87
4.23.1	Opis szczegółowy	88
4.23.2	Dokumentacja konstruktora i destruktora	88
4.23.2.1	tablica_asocjacyjna	88
4.23.3	Dokumentacja funkcji składowych	89
4.23.3.1	czy_blokada	89
4.23.3.2	czy_pusta	89
4.23.3.3	dodaj	89
4.23.3.4	insert	89
4.23.3.5	odblokuj	89
4.23.3.6	pobierz	90
4.23.3.7	ustaw	90
4.23.3.8	usun	90
4.23.3.9	wez	90
4.23.3.10	wez_id	91
4.23.3.11	wstaw	91
4.23.3.12	wypisz	91
4.23.3.13	zablokuj	91
4.23.3.14	zlicz_elementy	91
4.23.3.15	znajdz	92
4.23.3.16	znajdz	92
4.23.4	Dokumentacja atrybutów składowych	92
4.23.4.1	blok	92
4.23.4.2	found	92
4.23.4.3	key	93
4.23.4.4	s	93
4.23.4.5	sp	93
4.23.4.6	value	93
4.24	Dokumentacja szablonu klasy wezel< TYP >	93
4.24.1	Opis szczegółowy	94
4.24.2	Dokumentacja konstruktora i destruktora	94

4.24.2.1	wezel	94
4.24.2.2	wezel	94
4.24.2.3	~wezel	95
4.24.3	Dokumentacja funkcji składowych	95
4.24.3.1	dodaj_syna	95
4.24.3.2	wez_klucz	95
4.24.3.3	wez_wart	96
4.24.3.4	znajdz_nast	96
4.24.4	Dokumentacja atrybutów składowych	96
4.24.4.1	flag	96
4.24.4.2	klucz	97
4.24.4.3	lsyn	97
4.24.4.4	ojciec	97
4.24.4.5	psyn	97
4.24.4.6	wart	97
4.25	Dokumentacja klasy wierzcholek	97
4.25.1	Opis szczegółowy	98
4.25.2	Dokumentacja konstruktora i destruktora	98
4.25.2.1	wierzcholek	98
4.25.3	Dokumentacja przyjaciół i funkcji związanych	98
4.25.3.1	graf	98
4.25.4	Dokumentacja atrybutów składowych	98
4.25.4.1	id	98
4.25.4.2	waga	98
5	Dokumentacja plików	99
5.1	Dokumentacja pliku algorytm.cpp	99
5.1.1	Opis szczegółowy	99
5.2	Dokumentacja pliku algorytm.hh	99
5.2.1	Opis szczegółowy	101
5.3	Dokumentacja pliku drzewo.hh	101
5.3.1	Opis szczegółowy	102
5.3.2	Dokumentacja typów wyliczanych	102
5.3.2.1	syn	102
5.4	Dokumentacja pliku graf.cpp	103
5.4.1	Dokumentacja zmiennych	103
5.4.1.1	vec	103
5.5	Dokumentacja pliku graf.hh	103
5.5.1	Opis szczegółowy	104
5.6	Dokumentacja pliku hashtab.hh	105

5.6.1	Opis szczegółowy	105
5.7	Dokumentacja pliku kolejka.hh	106
5.7.1	Opis szczegółowy	107
5.8	Dokumentacja pliku main.cpp	107
5.8.1	Opis szczegółowy	107
5.8.2	Dokumentacja funkcji	108
5.8.2.1	main	108
5.9	Dokumentacja pliku operacje.cpp	108
5.10	Dokumentacja pliku operacje.hh	108
5.10.1	Dokumentacja definicji	109
5.10.1.1	ROZMIAR	109
5.11	Dokumentacja pliku statystyki.cpp	110
5.11.1	Dokumentacja funkcji	110
5.11.1.1	odchylenie_standardowe	110
5.11.1.2	srednia	111
5.12	Dokumentacja pliku statystyki.hh	111
5.12.1	Opis szczegółowy	112
5.12.2	Dokumentacja funkcji	112
5.12.2.1	odchylenie_standardowe	112
5.12.2.2	srednia	113
5.13	Dokumentacja pliku stos.hh	114
5.13.1	Opis szczegółowy	115
5.13.2	Dokumentacja typów wyliczanych	116
5.13.2.1	flag	116
5.14	Dokumentacja pliku str_operacje.cpp	116
5.14.1	Dokumentacja funkcji	116
5.14.1.1	operator<	116
5.14.1.2	operator<=	117
5.14.1.3	operator==	117
5.14.1.4	operator>	117
5.14.1.5	operator>=	117
5.15	Dokumentacja pliku str_operacje.hh	118
5.15.1	Dokumentacja funkcji	119
5.15.1.1	operator<	119
5.15.1.2	operator<=	119
5.15.1.3	operator==	119
5.15.1.4	operator>	119
5.15.1.5	operator>=	119
5.16	Dokumentacja pliku strona.dox	120
5.17	Dokumentacja pliku tablica_asocjacyjna.hh	120

5.17.1 Opis szczegółowy	121
Indeks	122

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

algorytm	7
bst	17
graf_test	38
h_sort	42
h_table	44
kolejka_lista	51
kolejka_tablica	53
m_sort	55
mnozenie	56
q_sort	67
stos_lista	80
stos_tablica	82
tab_aso	84
drzewo< TYP >	20
drzewo< float >	20
el_tab< TYP >	23
graf	25
hashtab< TYP >	46
operacje	59
queue_array< TYP >	69
queue_array< float >	69
queue_array< string >	69
queue_list< TYP >	73
queue_list< float >	73
stack_array< TYP >	75
stack_array< float >	75
stack_array< int >	75
stack_list< TYP >	78
stack_list< float >	78
tablica_asocjacyjna< TYP >	87
tablica_asocjacyjna< bool >	87
tablica_asocjacyjna< float >	87
wezel< TYP >	93
wezel< float >	93
wierzcholek	97

Rozdział 2

Indeks klas

2.1 Lista klas

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

algorytm	Definicja klasy algorytm Jest to klasa bazowa, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym	7
bst	Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania	17
drzewo< TYP >	Modeluje binarne drzewo przeszukiwania	20
el_tab< TYP >	Pojedynczy element tablicy haszującej	23
graf	Klasa modeluje pojęcie grafu w oparciu o listę incydencji, Operacje na grafie możliwe są na dwa sposoby \n	25
graf_test	Modeluje strukturę grafów użytych do badań	38
h_sort	Klasa reprezentuje dane poddane sortowaniu przez kopcowanie	42
h_table	Modeluje tablicę haszującą przeznaczoną do testowania szybkości wyszukiwania	44
hashtab< TYP >	Modeluje tablicę haszującą w oparciu o kontener klasy el_tab	46
kolejka_lista	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	51
kolejka_tablica	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	53
m_sort	Klasa reprezentuje dane poddane sortowaniu przez scalanie	55
mnozenie	Modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2	56
operacje	Klasa modeluje tablicę z danymi i metody służące do operacji na niej	59
q_sort	Klasa reprezentuje dane poddane sortowaniu szybkemu	67
queue_array< TYP >	Modeluje kolejkę w oparciu o tablicę	69
queue_list< TYP >	Modeluje kolejkę opartą na liście STL	73
stack_array< TYP >	Modeluje stos w oparciu o tablicę	75

stack_list< TYP >	
Modeluje stos oparty na liscie STL	78
stos_lista	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	80
stos_tablica	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	82
tab_aso	
Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania	84
tablica_asocjacyjna< TYP >	
Klasa modeluje tablice asocjacyjna	87
wezel< TYP >	
Modeluje pojedynczy węzeł drzewa	93
wierzcholek	
Klasa modeluje pojęcie wierzchołka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojaki interpretowanie wierzchołka grafu, wedle życzeń użytkownika	97

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

algorytm.cpp	Plik zawiera definicje metod klas zdefiniowanych w pliku algorytm.hh	99
algorytm.hh	Definicja klas wykonujących operacje na zestawie danych wejściowych	99
drzewo.hh	101
graf.cpp	103
graf.hh	103
hashtab.hh	105
kolejka.hh	Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby	106
main.cpp	Plik glowny	107
operacje.cpp	108
operacje.hh	108
statystyki.cpp	110
statystyki.hh	Plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk	111
stos.hh	Plik zawiera definicje klasy Stos Zaimplementowana na 2 sposoby	114
str_operacje.cpp	116
str_operacje.hh	118
tablica_asocjacyjna.hh	120

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy algorytm

Definicja klasy algorytm Jest to klasa bazowa, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym.

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

Diagram dziedziczenia dla algorytm

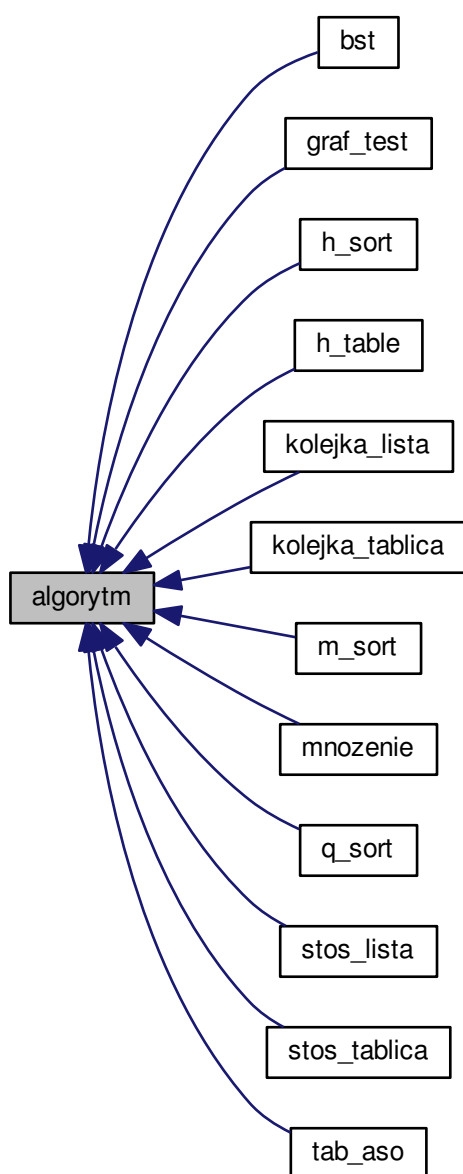
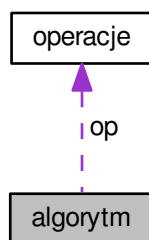


Diagram współpracy dla algorytm:



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 wejscowym, wywołuje metody odpowiedzialne za pomiar czasu oraz za porownanie wyniku operacji z plikiem wzorcowym
- bool **wczytaj** (ifstream &plik)
Metoda wczytuje plik wejscowy 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 wejscowych zgodnie z zadany algorytmem.
- bool **porownaj** ()
porownuje przetworzony dane z danymi wzorcowymi
- int **ile_danych** ()
- float * **jaki_czas** ()
- void **wlacz zegar** ()
Metoda włącza pomiar czasu poprzez włączenie funkcji gettimeofday i przechowanie czasu w zmiennej start.
- void **wylacz zegar** ()
Metoda wyłącza pomiar czasu poprzez włączenie 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, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym.

Definicja w linii 37 pliku algorytm.hh.

4.1.2 Dokumentacja konstruktora i destruktor

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

konstruktor kopiujący - przekazuje informacje o nazwach plików, które zapisywane są do pol klasy

Parametry

in	<i>plik1</i>	- plik wejściowy
in	<i>plik2</i>	- plik wzorcowy
in	<i>N</i>	- ilość danych wejściowych
in	<i>M</i>	- ilość powtórzeń

Definicja w linii 80 pliku algorytm.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 `int algorytm::ile_danych ()`

Zwraca

ilość liczb wejściowych

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 ()

porównuje przetworzony dane z danymi wzorcowymi

Zwraca

true - gdy pliki zgodne false - w przeciwnym przypadku

Definicja w linii 99 pliku algorytm.cpp.

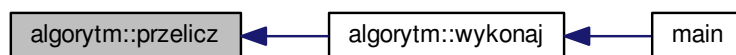
4.1.3.4 float algorytm::przelicz () [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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 wywołań tej funkcji:

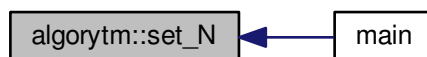


4.1.3.5 void algorytm::set_N (int wart) [inline]

metoda ustawia wartosc n

Definicja w linii 94 pliku algorytm.hh.

Oto graf wywołań tej funkcji:



4.1.3.6 bool algorytm::wczytaj (ifstream & plik)

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

Parametry

<i>in</i>	<i>plik</i>	- strumień pliku wejściowego
-----------	-------------	------------------------------

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

<i>in</i>	<i>plik</i>	- strumień pliku wejściowego
-----------	-------------	------------------------------

Definicja w linii 21 pliku algorytm.cpp.

4.1.3.8 void algorytm::wlacz zegar ()

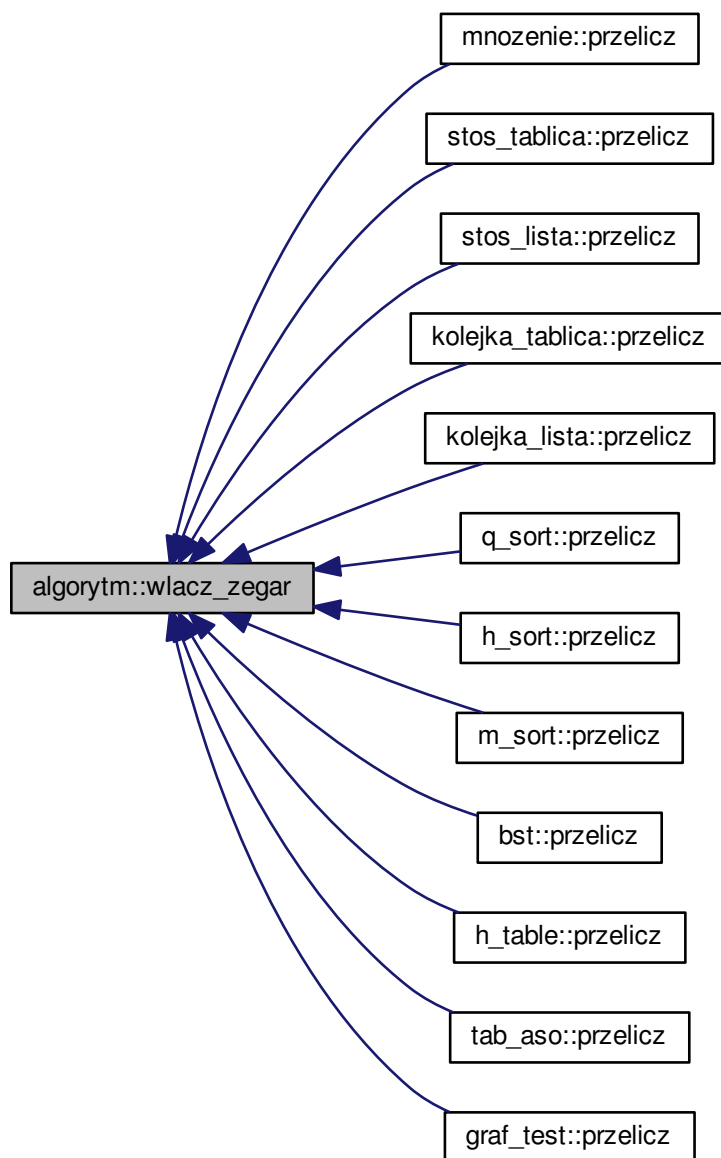
Metoda włącza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `start`.

Zwraca

`start` - zmienna pamiętająca czas poprzedzający wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywołań tej funkcji:

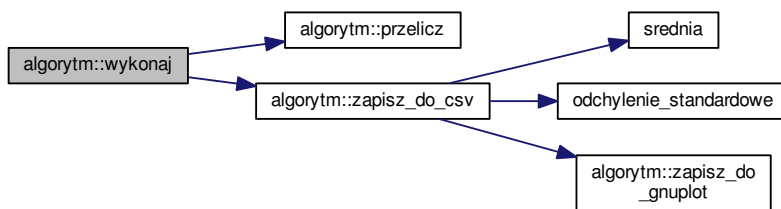


4.1.3.9 void algorytm::wykonaj (ofstream & out)

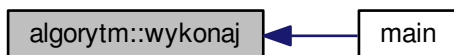
funkcja dokonuje operacji na pliku wejściowym, wywołuje metody odpowiedzialne za pomiar czasu oraz za porównanie wyniku operacji z plikiem wzorcowym

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 ()

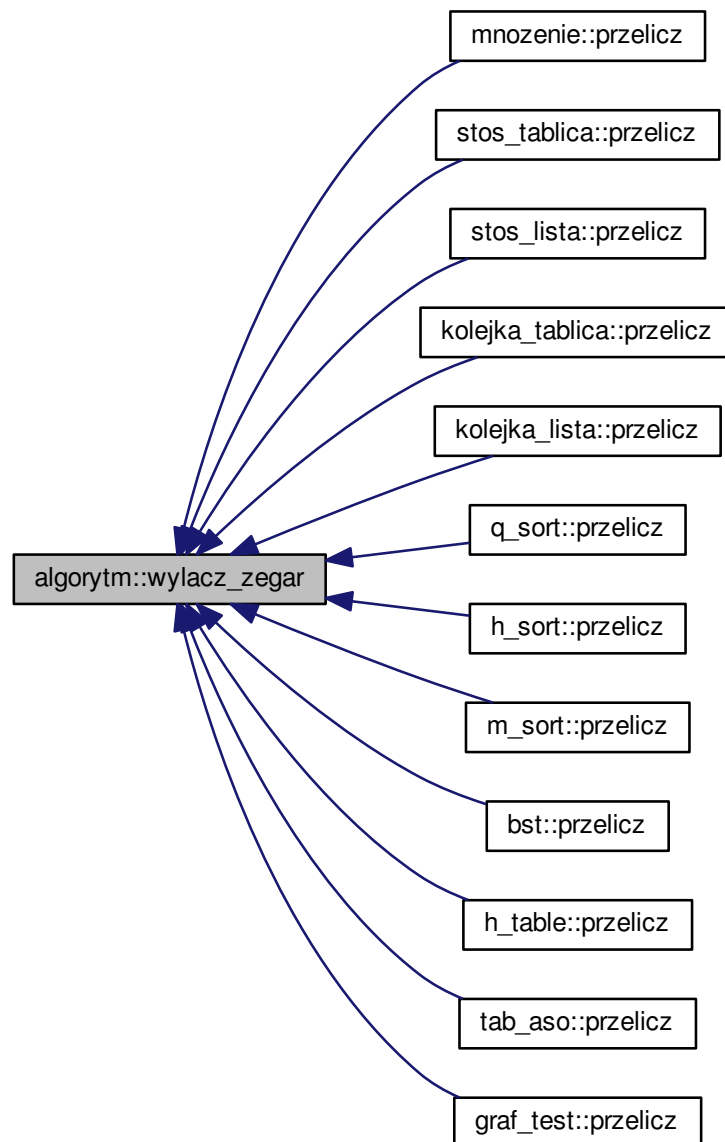
Metoda wyacza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `end`.

Zwraca

`end` - zmienna pamiętajaca czas poprzedzający wykonanie algorytmu

Definicja w linii 49 pliku `algorytm.cpp`.

Oto graf wywołań tej funkcji:

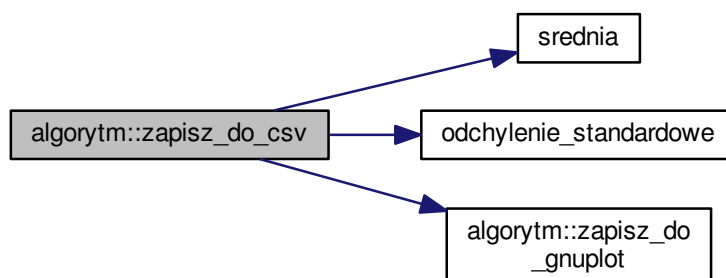


4.1.3.11 void algorytm::zapisz_do_csv (ofstream & out)

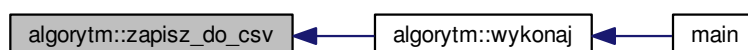
Metoda zapisuje tablice `czas` do pliku `wyjście.csv`.

Definicja w linii 62 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

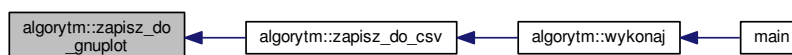


4.1.3.12 void algorytm::zapisz_do_gnuplot (ofstream & out, float sr, float od)

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 działania 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 szybkości wyszukiwania.

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

Diagram dziedziczenia dla bst

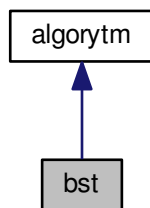
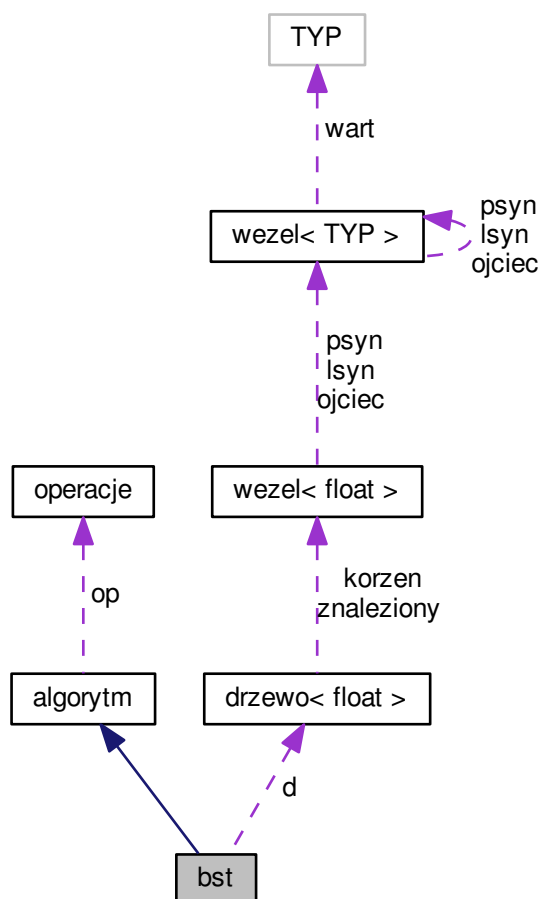


Diagram współpracy dla bst:



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 zadany algorytmem.

Atrybuty prywatne

- `drzewo` < float > `d`
- string * `klucze`

Dodatkowe Dziedziczone Składowe

4.2.1 Opis szczegółowy

Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.

Definicja w linii 227 pliku `algorytm.hh`.

4.2.2 Dokumentacja konstruktora i destruktor

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

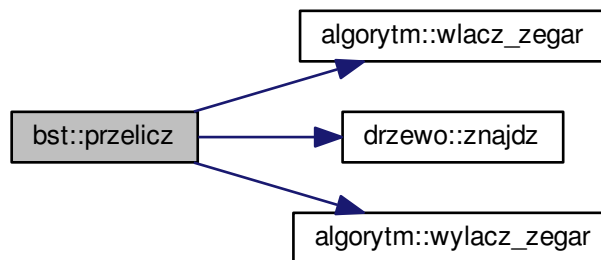
4.2.3.1 `float bst::przelicz ()` [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadany 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 zadany klucz
- 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 znaleziony
- void `czyszc` (`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 destruktor

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 >::czyszc (wezel< TYP > * w)` `[inline]`

rekursywne czyszczenie wezla

Parametry

<i>in</i>	<i>w</i>	- 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

<i>in</i>	<i>k</i>	- klucz wezla
<i>in</i>	<i>v</i>	= 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

<i>in</i>	<i>W</i>	- 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

<i>in</i>	<i>k</i>	- klucz
<i>in</i>	<i>w</i>	- wezel, w którym 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

<i>in</i>	<i>k</i>	- klucz wezla, który należy 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 zadany kluczu

Parametry

<code>in</code>	<code>k</code>	- 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:

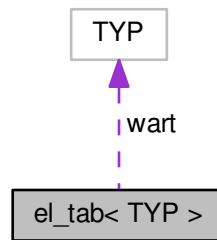
- [drzewo.hh](#)

4.4 Dokumentacja szablonu klasy `el_tab< TYP >`

pojedynczy element tablicy haszującej

```
#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 hashtable.hh.

4.4.2 Dokumentacja konstruktora i destruktoru

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

Definicja w linii 26 pliku hashtable.hh.

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

Definicja w linii 27 pliku hashtable.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 hashtable.hh.

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

wartosc pola

Definicja w linii 21 pliku hashtable.hh.

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

flaga informujaca, czy pole jest zajete

Definicja w linii 25 pliku hashtable.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

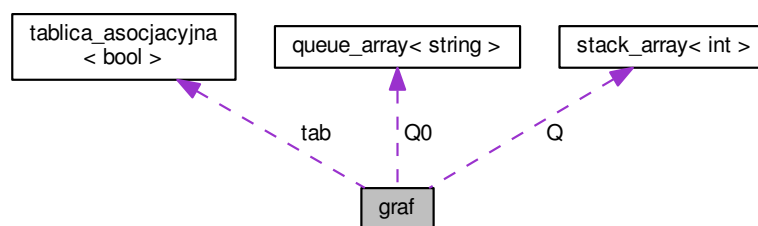
- [hashtab.hh](#)

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, wierzchołkom 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 wierzchołki 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 wierzchoikiem - odwolanie poprzez obiekt typu wierzcholek
 - void `dodaj_wierzcholek` (`wierzcholek w`)
Dodaje wierzcholek do wezla, wierzchoikom 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 wierzchoлков.
 - 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 wierzchoikiem - odwolanie poprzez identyfikator wierzcholka
 - void `usun_krawedz` (unsigned int id1, unsigned int id2)
usuwa krawedz spomiedzy 2 wierzchołkow - odwolanie poprzez identyfikatory wierzchołkow
 - void `usun_krawedz` (`wierzcholek w1`, `wierzcholek w2`)
usuwa krawedz spomiedzy 2 wierzchołkow - 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 wszere 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 destruktor

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)

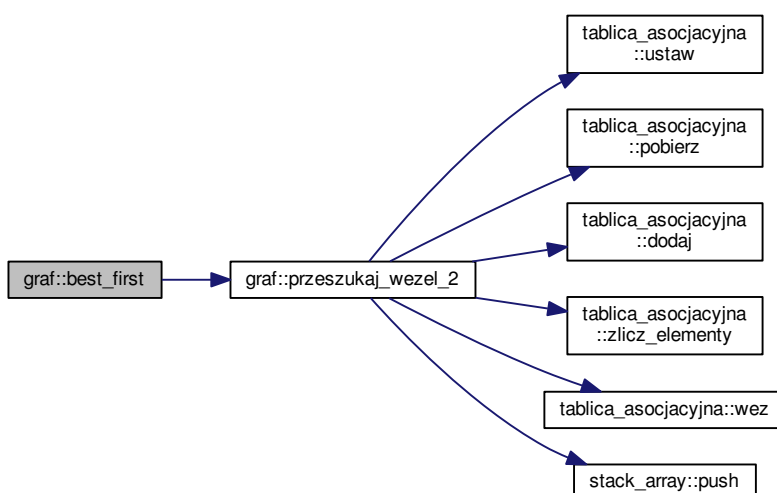
Metoda przeszukuje graf, poczynajac od najkrotszej sciezki.

Parametry

in	id	- wezel, ktorego szukamy
----	----	--------------------------

Definicja w linii 222 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań tej funkcji:



4.5.3.2 void graf::bfs (int id)

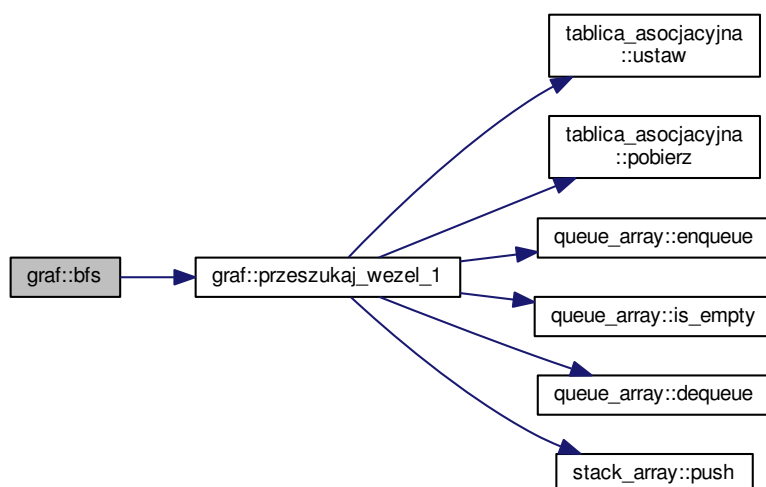
Metoda przeszukuje wszerek cały graf.

Parametry

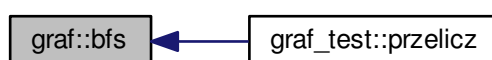
in	id	- wezel, ktorego szukamy
----	----	--------------------------

Definicja w linii 172 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań tej funkcji:



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

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

Parametry

in	<i>id1</i>	- id 1. wierzcholka
in	<i>id2</i>	- 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 - odwołanie poprzez obiekt klasy wierzcholek.

Parametry

in	<i>w1</i>	- pierwszy wierzcholek
in	<i>w2</i>	- 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*)

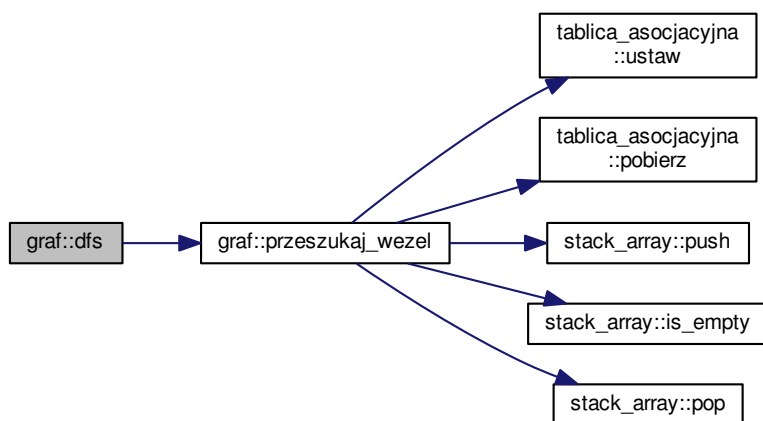
Metoda przeszukuje wglab cały graf.

Parametry

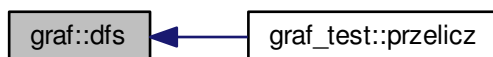
in	id	- wezel, ktorego szukamy
----	----	--------------------------

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)

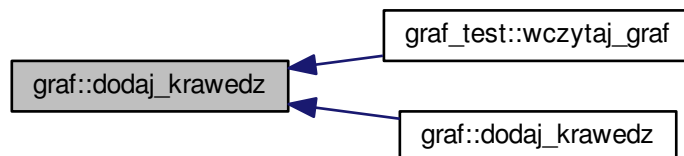
Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez identyfikatory wierzchołkow.

Parametry

in	id1	- id 1. wierzchołka
in	id2	- id 2. wierzchołka
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 pomiędzy 2 wezły - odwołanie 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:

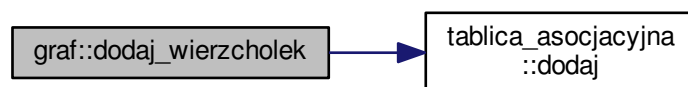


4.5.3.8 void graf::dodaj_wierzcholek ()

Dodaje wierzcholek do wezła, wierzchołkom przypisuje się 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:

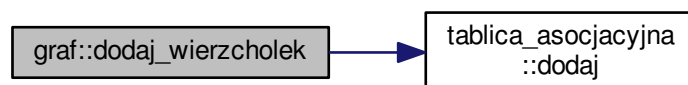


4.5.3.9 void graf::dodaj_wierzcholek (wierzcholek w)

Dodaje wierzcholek do wezła, wierzchołkom przypisuje się identyfikatory będące kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawędzi 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 wezeł nie był odwiedzony, odkłada na stos wszystkie jego nieodwiedzone następniki i rekurencyjnie je przeszukuje.

Parametry

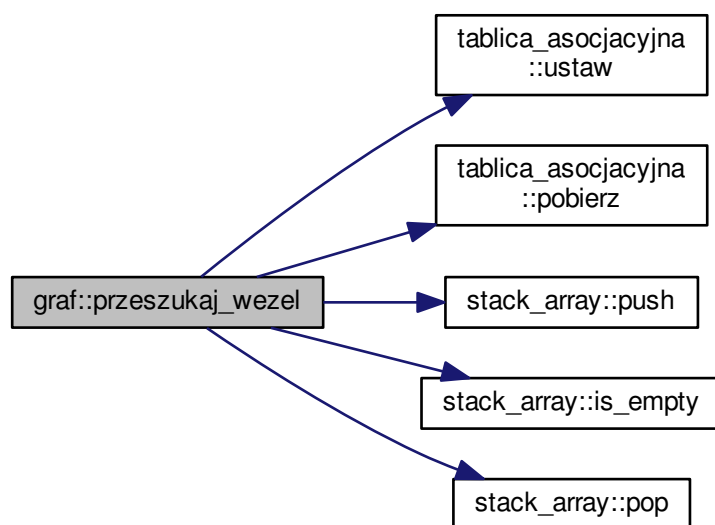
in	id	- id wezla, ktory ma byc przeszukany
in	wzor	- id wezla, ktorego szukamy

Zwraca

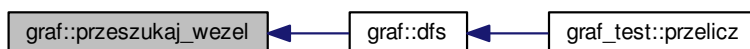
informacja, czy wezel zostal znaleziony

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 nie był odwiedzony, odkłada do kolejki wszystkie jego nieodwiedzone następniki i rekurencyjnie je przeszukuje.

Parametry

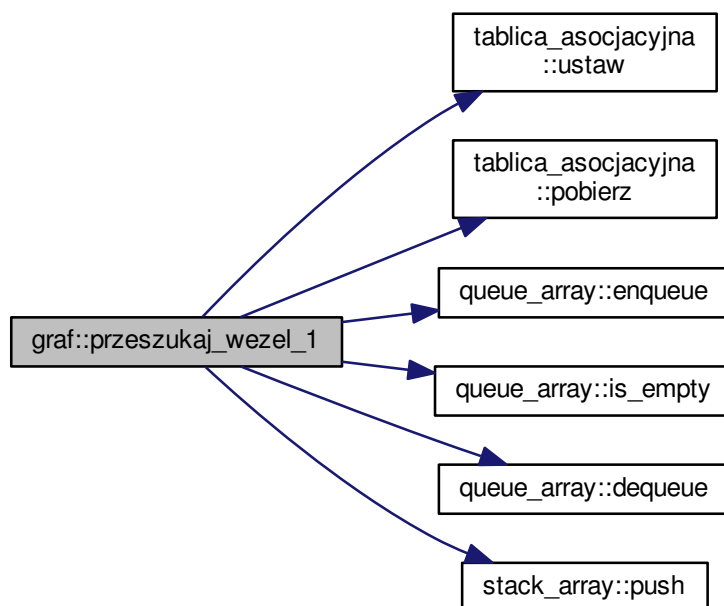
in	<i>id</i>	- id wezla, który ma być przeszukany
in	<i>wzor</i>	- id wezla, którego szukamy

Zwraca

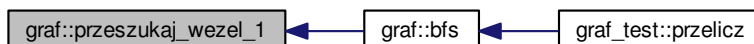
informacja, czy wezel został znaleziony

Definicja w linii 138 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań tej funkcji:



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

Jeżeli wezel nie był odwiedzony, odkłada do tablicy asocjacyjnej wszystkie jego nieodwiedzone następniki i rekurencyjnie je przeszukuje, poczynając od tego, do którego mamy najkrótszą ścieżkę.

Parametry

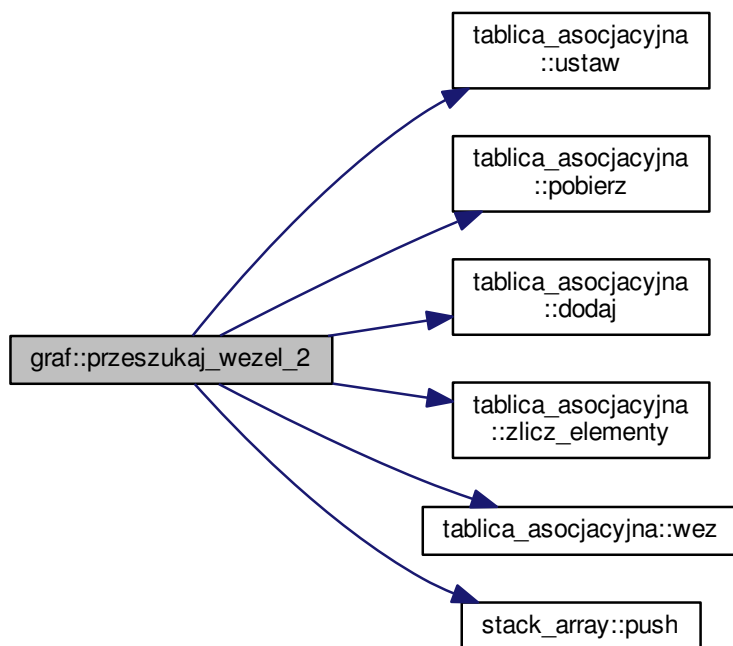
in	id	- id wezła, który ma być przeszukany
in	wzor	- id wezła, którego szukamy

Zwraca

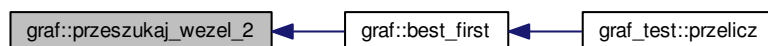
informacja, czy wezeł został znaleziony

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 wierzchołki połączone krawędzią z podanym wierzchołkiem - odwołanie poprzez obiekt typu wierzcholek

Parametry

<i>in</i>	<i>w</i>	- 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 wierzchoikiem - odwołanie poprzez identyfikator wierzcholka

Parametry

<i>in</i>	<i>id</i>	- 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 wierzchołkow - odwołanie poprzez identyfikatory wierzchołkow

Parametry

<i>in</i>	<i>id1</i>	- id 1. wierzcholka
<i>in</i>	<i>id2</i>	- id 2. wierzcholka

Definicja w linii 73 pliku graf.cpp.

Oto graf wywoływać tej funkcji:

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

usuwa krawedz spomiedzy 2 wierzchołkow - odwołanie poprzez obiekt typu wierzcholek

Parametry

<i>in</i>	<i>w1</i>	- pierwszy wierzcholek
<i>in</i>	<i>w2</i>	- 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

<i>in</i>	<i>id</i>	- 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

<i>in</i>	<i>w</i>	- 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 incydencji 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

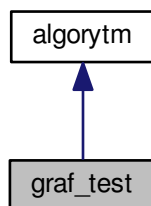
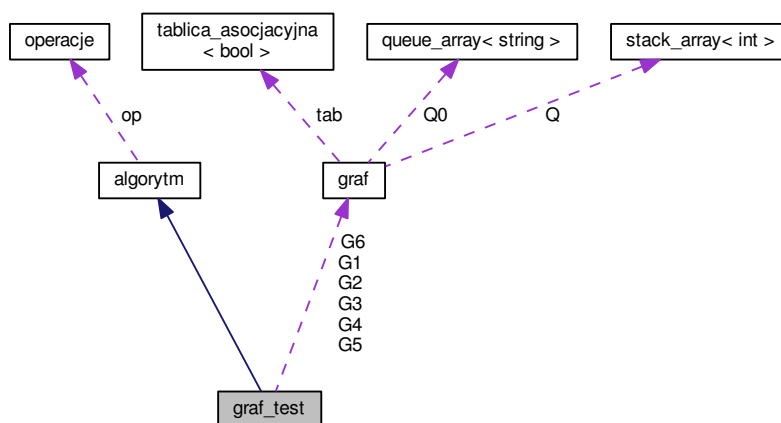


Diagram współpracy dla graf_test:



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 zadany 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 użytych do badań

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

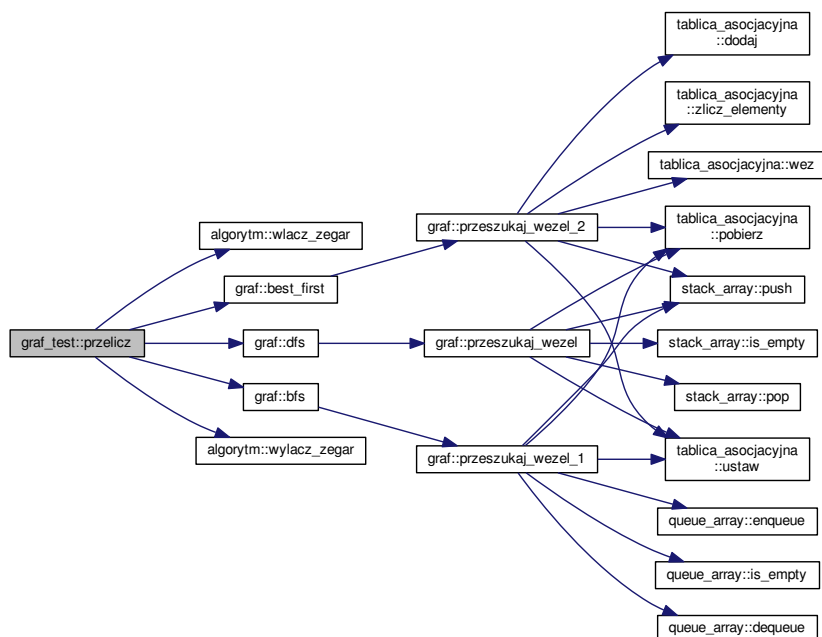
4.6.3.1 `float graf_test::przelicz ()` `[virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 329 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:

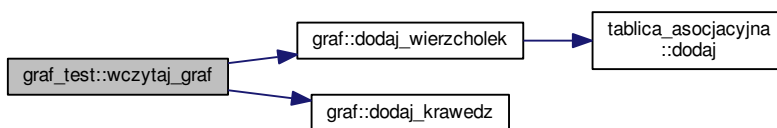


4.6.3.2 void graf_test::wczytaj_graf ()

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 zastosować

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`

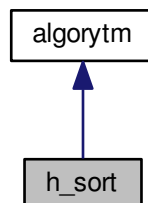
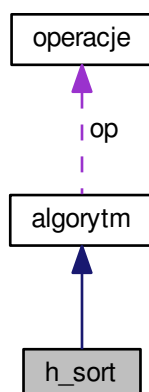


Diagram współpracy dla h_sort:



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 destruktor

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

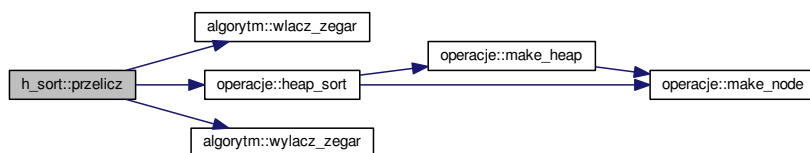
4.7.3.1 `float h_sort::przelicz ()` [virtual]

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 haszująca przeznaczona do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla `h_table`

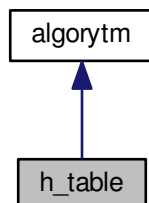
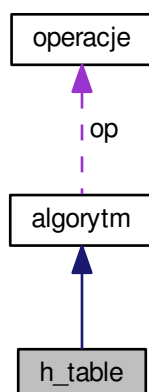


Diagram współpracy dla h_table:



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 wejściowych zgodnie z zadany algorytmem.

Atrybuty prywatne

- string * `klucze`
tablica kluczy

Dodatkowe Dziedziczone Składowe

4.8.1 Opis szczegółowy

Modeluje tablice haszująca przeznaczona do testowania szybkości wyszukiwania.

Definicja w linii 247 pliku algorytm.hh.

4.8.2 Dokumentacja konstruktora i destruktor

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

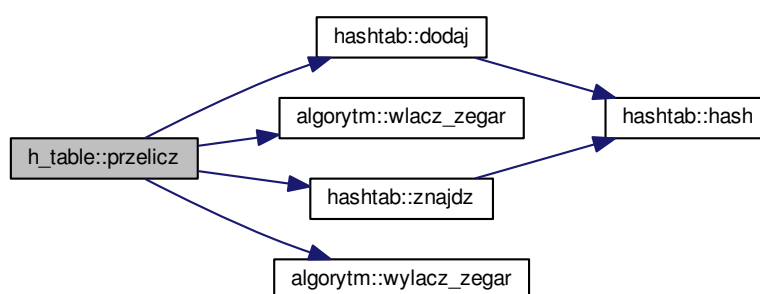
4.8.3.1 float h_table::przelicz () [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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	- strumień z kluczami użytymi podczas testów
----	------	--

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 haszująca 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 haszujacej
- `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 hashtable< TYP >`

modeluje tablice haszujca w oparciu o kontener klasy `el_tab`

Definicja w linii 34 pliku hashtable.hh.

4.9.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 43 pliku hashtable.hh.

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

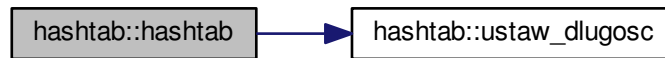
konsruktor parametryczny

Parametry

in	N	- rozmiar tablicy
----	---	-------------------

Definicja w linii 47 pliku hashtable.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 haszującej

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]`

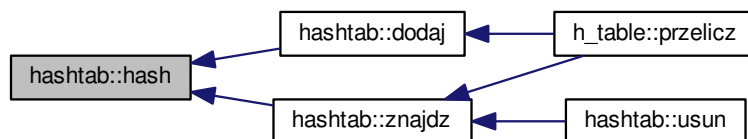
funkcja haszująca

Zwraca

h - liczba, która po kompresji będzie indeksem danego elementu

Definicja w linii 51 pliku `hashtab.hh`.

Oto graf wywołań tej funkcji:



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

ustawia dlugosc tablicy

Definicja w linii 41 pliku hashtable.hh.

Oto graf wywołań tej funkcji:



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

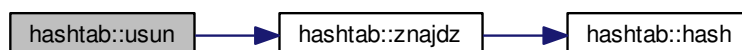
usuwa element jesli znajduje sie w tablicy

Parametry

<code>in</code>	<code>k</code>	- klucz
-----------------	----------------	---------

Definicja w linii 89 pliku hashtable.hh.

Oto graf wywołań dla tej funkcji:



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

Definicja w linii 93 pliku hashtable.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

<code>in</code>	<code>k</code>	- klucz elementu
-----------------	----------------	------------------

Zwraca

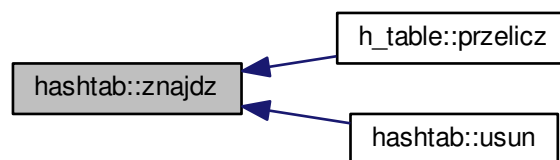
znaleziony element

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

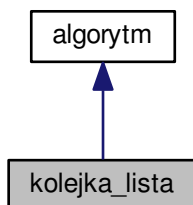
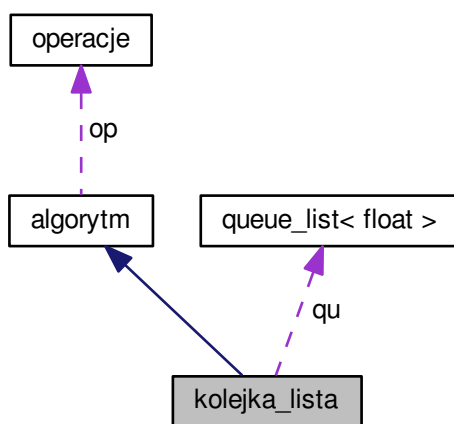


Diagram współpracy dla kolejka_lista:



Metody publiczne

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

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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

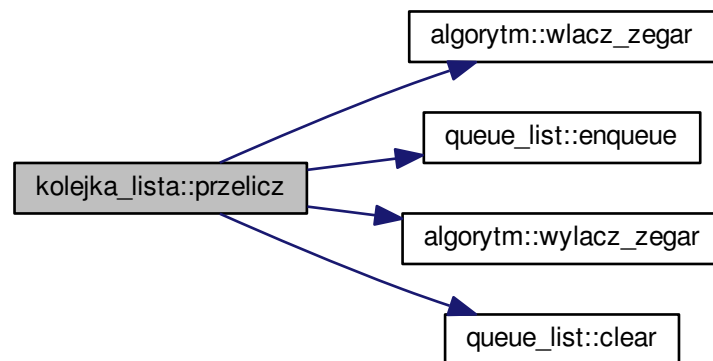
4.10.3.1 `float kolejka_lista::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadaniem 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

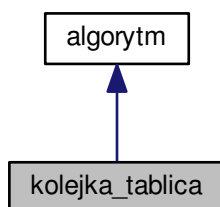
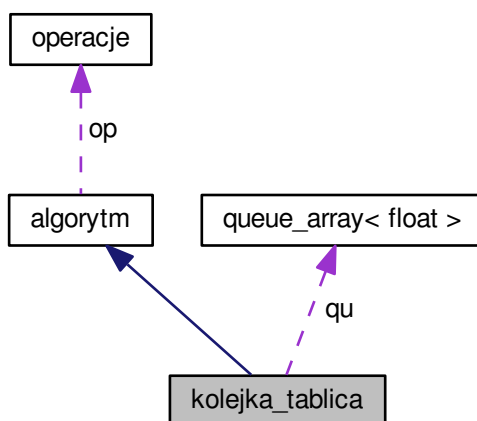


Diagram współpracy dla kolejka_tablica:



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 zadany 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 destruktor

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

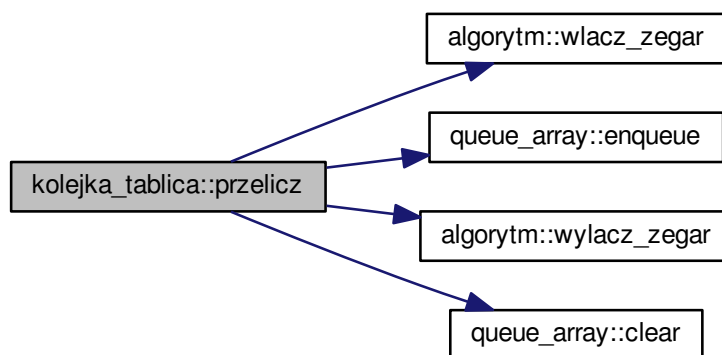
4.11.3.1 `float kolejka_tablica::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytm.

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

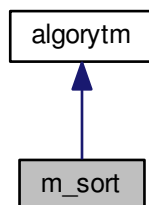
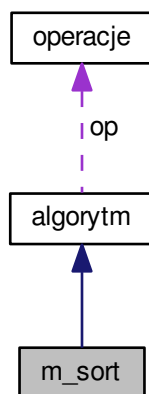


Diagram współpracy dla m_sort:



Metody publiczne

- `m_sort` (ifstream &plik1, ifstream &plik2, int N, int M)
konstruktor
- float `przelicz` ()
metoda dokonująca 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 destruktor

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

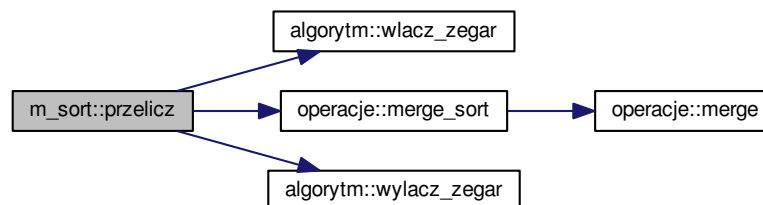
4.12.3.1 `float m_sort::przelicz ()` [virtual]

metoda dokonująca 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 dokonujący mnożenia każdego elementu pliku wejściowego przez 2

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

Diagram dziedziczenia dla mnozenie

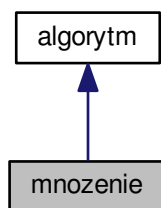
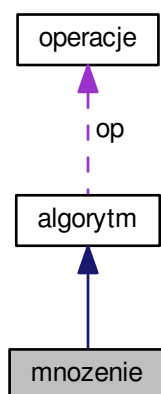


Diagram współpracy dla mnozenie:



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 destruktor

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 wejscowego i wzorcowego

Parametry

in	<i>plik1</i>	- plik wejsciowy
in	<i>plik2</i>	- plik wzorcowy
in	<i>N</i>	- ilosc danych wejsciowych
in	<i>M</i>	- ilosc powtorzen

Definicja w linii 148 pliku algorytm.hh.

4.13.3 Dokumentacja funkcji składowych

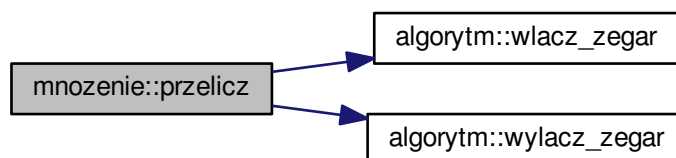
4.13.3.1 float mnozenie::przelicz () [virtual]

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 sortowniaa 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 destruktoru

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

<code>in</code>	<code>N</code>	- ilosc elementow w tablicy; parametr przypisywany do pola <code>n</code> 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, alokując dodatkową pamięć

Parametry

<i>in</i>	<i>e</i>	- element, który należy dołączyć 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

<i>in</i>	<i>tab2</i>	- tablica, która należy dołączyć
<i>in</i>	<i>rozm</i>	- rozmiar tablicy <i>tab2</i>

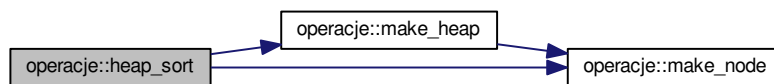
Definicja w linii 46 pliku operacje.cpp.

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 ()

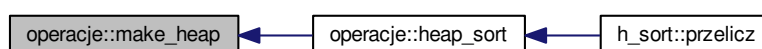
Metoda tworzy kopiec binarny.

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	<i>rozmiar</i>	- rozmiar tablicy
in	<i>i</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, jednocześnie je porzadkujac.

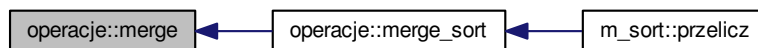
Parametry

in	<i>poczatek</i>	- pierwszy indeks tablicy
in	<i>srodek</i>	- srodkowy indeks tablicy

<i>in</i>	<i>koniec</i>	- ostatni indeks tablicy
-----------	---------------	--------------------------

Definicja w linii 130 pliku operacje.cpp.

Oto graf wywołań tej funkcji:



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

\ brief Metoda dokonuje sortowania poprzez rekurencyjne wywołanie dla obu połow tablic, następnie metoda dokonuje scalenia danych

Parametry

<i>in</i>	<i>poczatek</i>	- pierwszy indeks tablicy
<i>in</i>	<i>koniec</i>	- ostatni indeks tablicy

Definicja w linii 166 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań 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*)

Przeciążenie operatora przypisania; przypisuje elementy tablicy `tab1` do tablicy będącej polem klasy.

Parametry

<i>in</i>	<i>tab1</i>	- 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

<i>in</i>	<i>tab1</i>	- tablica, ktorej wartosci porownujemy
-----------	-------------	--

Zwraca

true - gdy zawartosc 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

<i>in</i>	<i>l</i>	- pierwszy indeks tablicy
<i>in</i>	<i>p</i>	- 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

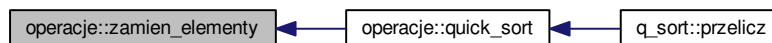
in	<i>i</i>	- element tablicy
in	<i>j</i>	- element tablicy

Zwraca

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

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywołań 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 szybkemu

```
#include <algorithm>
```

Diagram dziedziczenia dla q_sort

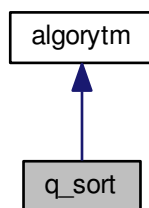
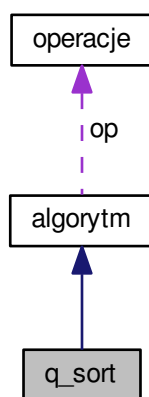


Diagram współpracy dla q_sort:



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 szybkemu

Definicja w linii 200 pliku `algorytm.hh`.

4.15.2 Dokumentacja konstruktora i destruktor

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

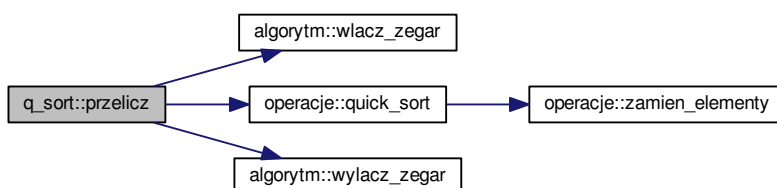
4.15.3.1 `float q_sort::przelicz () [virtual]`

metoda dokonująca 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:

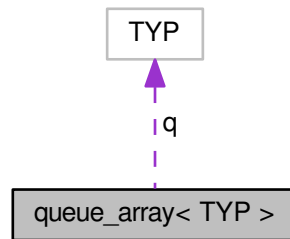
- [algorytm.hh](#)
- [algorytm.cpp](#)

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 destruktoru

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:

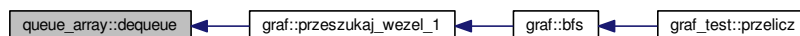


4.16.3.2 `template<typename TYP> TYP queue_array< TYP >::dequeue () [inline]`

usuwa element z konca kolejki

Definicja w linii 129 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:

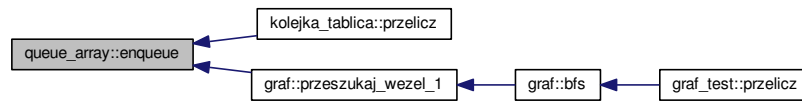


4.16.3.3 `template<typename TYP> void queue_array< TYP >::enqueue (TYP element) [inline]`

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

Definicja w linii 82 pliku kolejka.hh.

Oto graf wywołań tej funkcji:



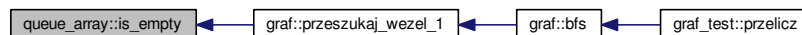
4.16.3.4 `template<typename TYP> bool queue_array< TYP >::is_empty () [inline]`

Zwraca

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

Definicja w linii 75 pliku kolejka.hh.

Oto graf wywołań 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 zwiększania pamięci , przyjmuje wartość : plus1 - dla trybu każdorazowego powiększania pamięci 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 kolejkę opartą na liście STL.

```
#include <kolejka.hh>
```

Metody publiczne

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

dodaje element

- `TYP dequeue ()`

usuwa element

- `void clear ()`

czyszczy stos

Atrybuty prywatne

- `list< TYP > q`

4.17.1 Opis szczegółowy

```
template<typename TYP>class queue_list< TYP >
```

Modeluje kolejkę opartą na liście 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]`

czyszczy 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:

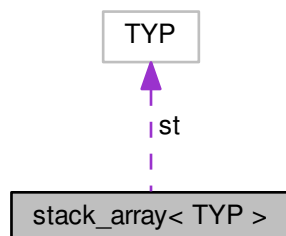
- [kolejka.hh](#)

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 destruktor

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:



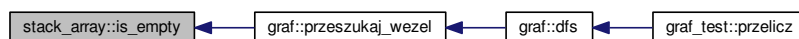
4.18.3.2 `template<typename TYP> bool stack_array< TYP >::is_empty () [inline]`

Zwraca

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

Definicja w linii 79 pliku `stos.hh`.

Oto graf wywołań tej funkcji:

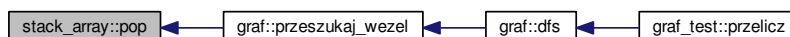


4.18.3.3 `template<typename TYP> TYP stack_array< TYP >::pop () [inline]`

zdejmuje element ze stosu

Definicja w linii 140 pliku `stos.hh`.

Oto graf wywołań tej funkcji:

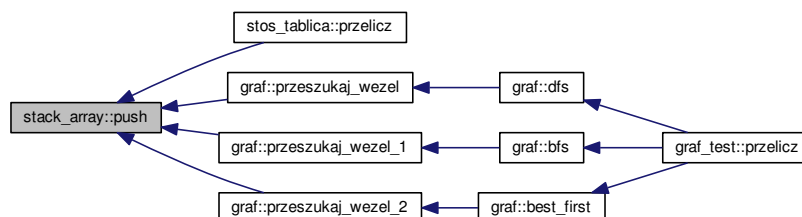


4.18.3.4 `template<typename TYP> void stack_array< TYP >::push (TYP element) [inline]`

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

Definicja w linii 91 pliku `stos.hh`.

Oto graf wywołań 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 zwiększania pamięci , przyjmuje wartość :

plus1 - dla trybu każdorazowego powiększania pamięci

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 liście 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` ()
czyszczy stos

Atrybuty prywatne

- list< TYP > `st`

4.19.1 Opis szczegółowy

```
template<typename TYP>class stack_list< TYP >
```

Modeluje stos oparty na liście 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]`

czyszczy stos

Definicja w linii 50 pliku `stos.hh`.

Oto graf wywołań 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

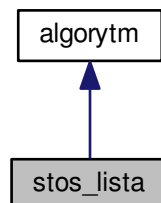
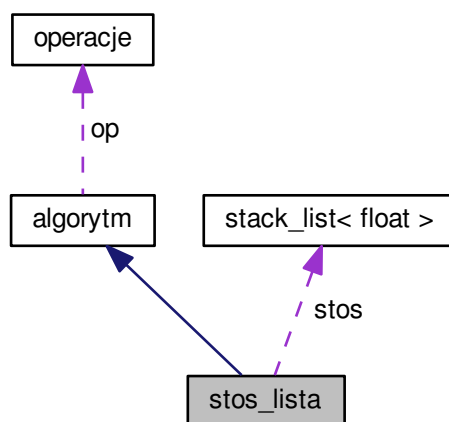


Diagram współpracy dla stos_lista:



Metody publiczne

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

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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 destruktor

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

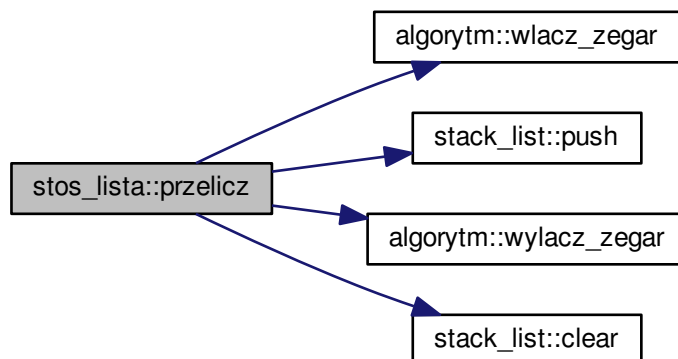
4.20.3.1 `float stos_lista::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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`

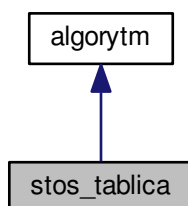
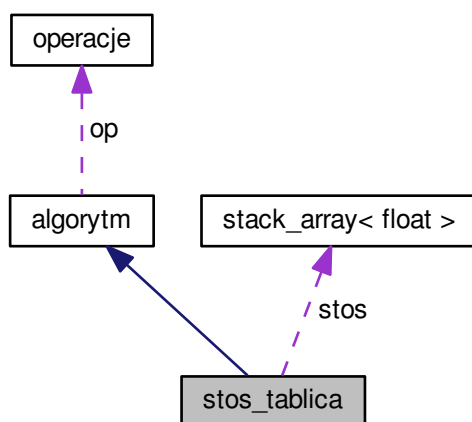


Diagram współpracy dla stos_tablica:



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 zadany 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

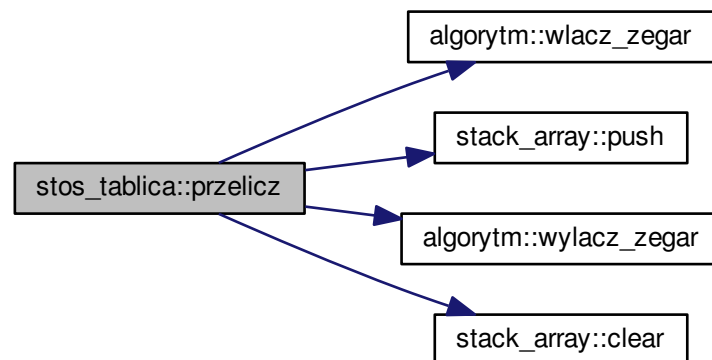
4.21.3.1 `float stos_tablica::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany 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 asocjacyjną przeznaczoną do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla tab_aso

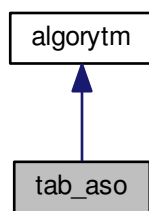
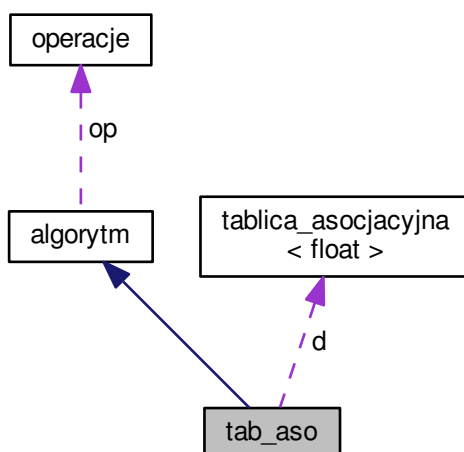


Diagram współpracy dla tab_aso:



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 zadany 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 szybkości wyszukiwania.

Definicja w linii 265 pliku `algorytm.hh`.

4.22.2 Dokumentacja konstruktora i destruktor

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

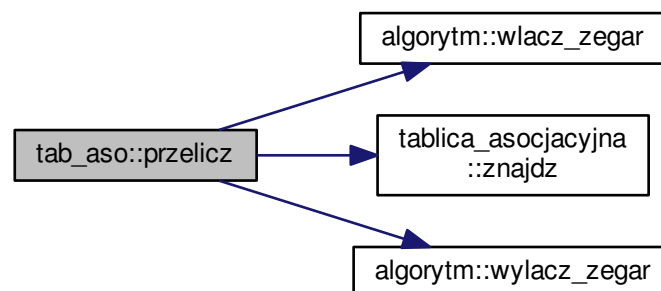
4.22.3.1 `float tab_aso::przelicz ()` `[virtual]`

Metoda odpowiada za przetworzenie danych wejściowych 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

<code>in</code>	<code>plik</code>	- strumień z kluczami użytymi podczas testów
-----------------	-------------------	--

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

<code>in</code>	<code>plik</code>	- strumień z kluczami użytymi podczas testów
-----------------	-------------------	--

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 następujące parametry.
- `void dodaj (string k, TYP v)`
Metoda dodaje element do struktury. Gdy (uwzględniając porządek alfabetyczny) element ma stać w skrajnym miejscu tablicy, dodawany jest od razu. W przeciwnym razie funkcja `wstaw` szuka odpowiedniego miejsca. Ponadto metoda tworzy pamięć dla struktury, gdy uprzednio jest ona pusta.
- `void usun (string k)`
Metoda usuwa zadany element, korzystając z funkcji `znajdz`.
- `TYP pobierz (string k)`
Metoda zwraca użytkownikowi szukany element, pod warunkiem, że jest on w zbiorze.
- `bool znajdz (string k)`
- `bool czy_pusta ()`
- `int zlicz_elementy ()`
- `void wypisz ()`
- `TYP wez (int ind)`
Metoda wprost odnosi się do konkretnego elementu tablicy - metoda używana przy grafie.
- `string wez_id (int ind)`
metoda wprost odnosi się do klucza, konkretnego elementu tablicy - metoda używana przy grafie

- bool `czy_blokada` ()
metoda sprawdza, czy dostęp do tablicy jest możliwy
- void `zablokuj` ()
metoda blokuje dostęp do tablicy
- void `odblokuj` ()
metoda zezwala na dostęp do tablicy
- void `ustaw` (string k, TYP v)
metoda zmienia wartość w miejscu, które wskazuje klucz k

Metody prywatne

- void `insert` (int ind, string k, TYP v)
Metoda która umieszcza wartość oraz jej klucz w zadanym miejscu. Gdy wartość z kluczem jest dodawana w środek struktury, dane na prawo od niej przesuwane są o jeden w prawo. Gdy istnieje potrzeba powiększenia tablicy, stosuje się znany już rodzaj gospodarowania pamięcią, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze względu na złożoność obliczeniową.
- void `wstaw` (string k, TYP v, int ind_l, int ind_r)
Metoda szuka pozycji, w którą należy dodać element, aby tablica była posortowana alfabetycznie.
- int `znajdz` (string k, int ind_l, int ind_r)
Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, zwróci wartość w strukturze, flaga found ustawiana jest na wartość true.

Atrybuty prywatne

- string * `key`
Tablica zawierająca klucze poszukiwane.
- TYP * `value`
Tablica zawierająca wartości.
- int `s`
rozmiar tablicy
- int `sp`
rozmiar danych zapełniających tablice
- bool `found`
flaga informująca 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 asocjacyjną.

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 następujące 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 dostęp do tablicy jest możliwy

Zwraca

true, gdy tablica zablokowana, false, gdy dostęp jest możliwy

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

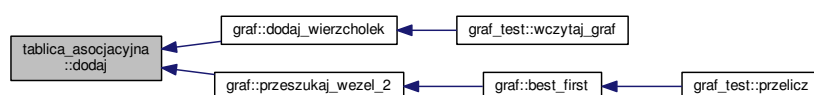
Definicja w linii 186 pliku `tablica_asocjacyjna.hh`.

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

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

Definicja w linii 128 pliku `tablica_asocjacyjna.hh`.

Oto graf wywołań tej funkcji:



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

Metoda która umieszcza wartość oraz jej klucz w zadanym miejscu. Gdy wartość z kluczem jest dodawana w środek struktury, dane na prawo od niej przesuwane są o jeden w prawo. Gdy istnieje potrzeba powiększenia tablicy, stosuje się znany już rodzaj gospodarowania pamięcią, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze względu na złożoność obliczeniową.

Definicja w linii 35 pliku `tablica_asocjacyjna.hh`.

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

metoda zezwala na dostęp 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]`

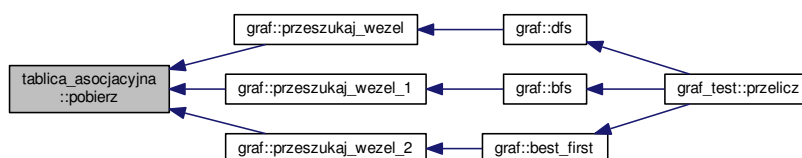
Metoda zwraca uzytkownikowi szukany element, pod warunkiem, ze jest on w zbiorze.

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 wywoływów tej funkcji:



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

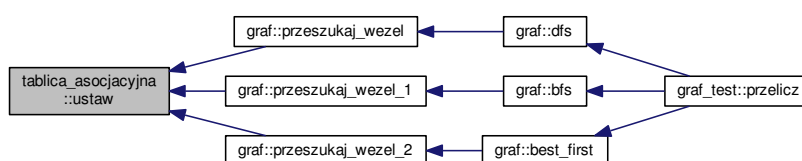
metoda zmienia wartosc w miejscu, ktore wskazuje klucz k

Parametry

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

Definicja w linii 233 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływów 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

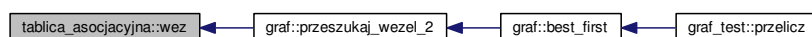
<code>in</code>	<code>ind</code>	- indeks tablicy
-----------------	------------------	------------------

Zwraca

`value[ind]` - wartosc mieszcza sie pod zadany indeks

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

<code>in</code>	<code>ind</code>	- indeks tablicy
-----------------	------------------	------------------

Zwraca

`key[ind]` - klucz mieszcza sie pod zadany indeks

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`.

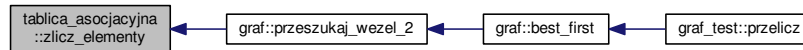
4.23.3.14 `template<typename TYP> int tablica_asocjacyjna< TYP >::zlicz_elementy () [inline]`

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]`

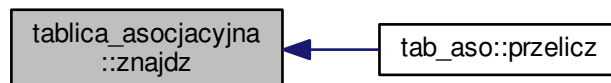
Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, tzn jest zawarty w strukturze, flaga found ustawiana jest na wartosc true.

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 zawierająca 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 zawierająca wartości.

Definicja w linii 22 pliku tablica_asocjacyjna.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

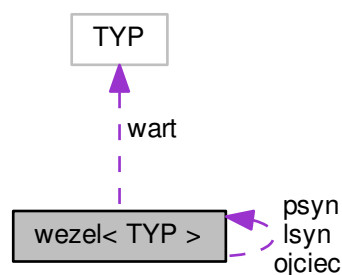
- [tablica_asocjacyjna.hh](#)

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.

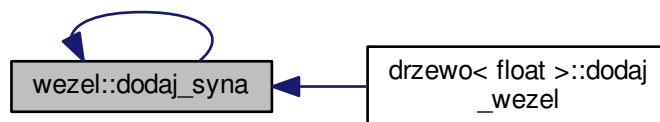
4.24.3 Dokumentacja funkcji składowych

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:



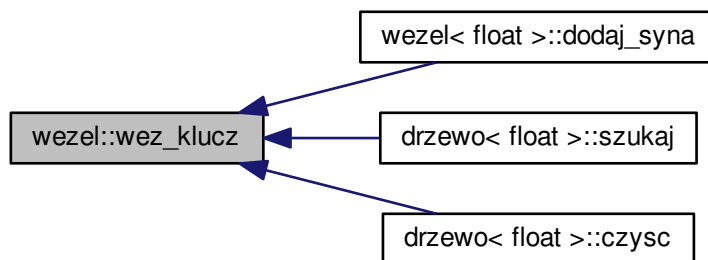
4.24.3.2 `template<typename TYP> string wezel< TYP >::wez_klucz () [inline]`

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

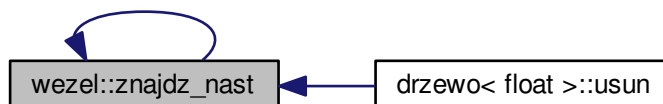
wartosc wezla

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`

określa, 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 dwojake interpretowanie 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 zparzyjzazniona

4.25.1 Opis szczegółowy

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

Definicja w linii 17 pliku graf.hh.

4.25.2 Dokumentacja konstruktora i destruktor

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 zparzyjzazniona

Definicja w linii 19 pliku graf.hh.

4.25.4 Dokumentacja atrybutów składowych

4.25.4.1 int wierzcholek::id [private]

numer identyfikacyjny 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

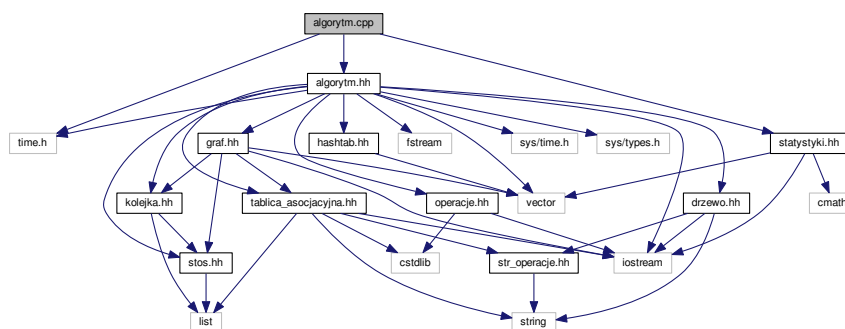
Dokumentacja plików

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

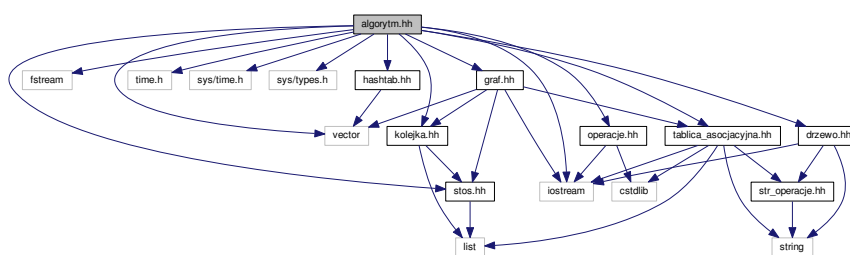
Definicja klas wykonujących operacje na zestawie danych wejściowych.

```

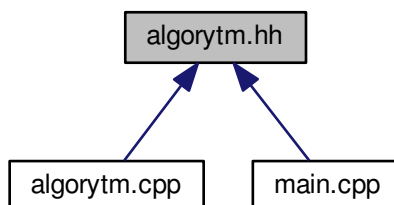
#include <iostream>
#include <fstream>
#include <vector>
#include <time.h>
#include <sys/time.h>
#include <sys/types.h>
#include "operacje.hh"
#include "stos.hh"
#include "kolejka.hh"
#include "drzewo.hh"
#include "hashtab.hh"
#include "tablica_asocjacyjna.hh"
#include "graf.hh"

```

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, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym.
- class `mnozenie`
modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego 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 szybkemu*
- 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 szybkości wyszukiwania.*
- class [h_table](#)
 - Modeluje tablice haszująca przeznaczona do testowania szybkości wyszukiwania.*
- class [tab_aso](#)
 - Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.*
- class [graf_test](#)
 - modeluje struktury grafów użytych do badań*

5.2.1 Opis szczegółowy

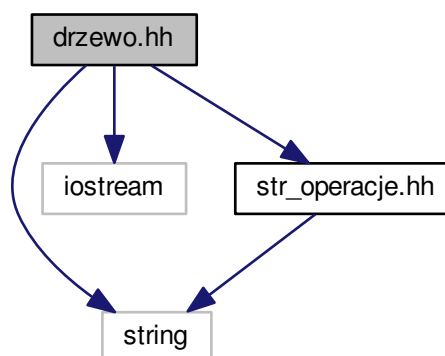
Definicja klas wykonujących operacje na zestawie danych wejściowych.

Definicja w pliku [algorytm.hh](#).

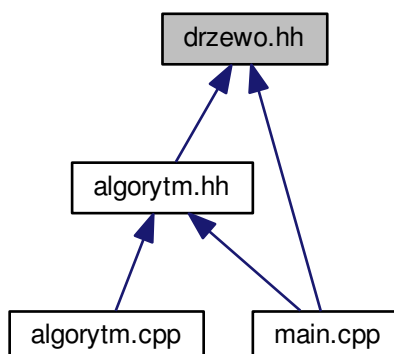
5.3 Dokumentacja pliku drzewo.hh

```
#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 wezeł drzewa
- class `drzewo< TYP >`
modeluje binarne drzewo przeszukiwan

Wyliczenia

- enum `syn { lewy, zaden, prawy }`
typ wyliczeniowy, określa, czym synem jest dany element drzewa

5.3.1 Opis szczegółowy

Plik zawiera definicje klasy reprezentującej drzewo binarne

Definicja w pliku `drzewo.hh`.

5.3.2 Dokumentacja typów wyliczanych

5.3.2.1 enum `syn`

typ wyliczeniowy, określa, czym synem jest dany element drzewa

Wartości wyliczeń

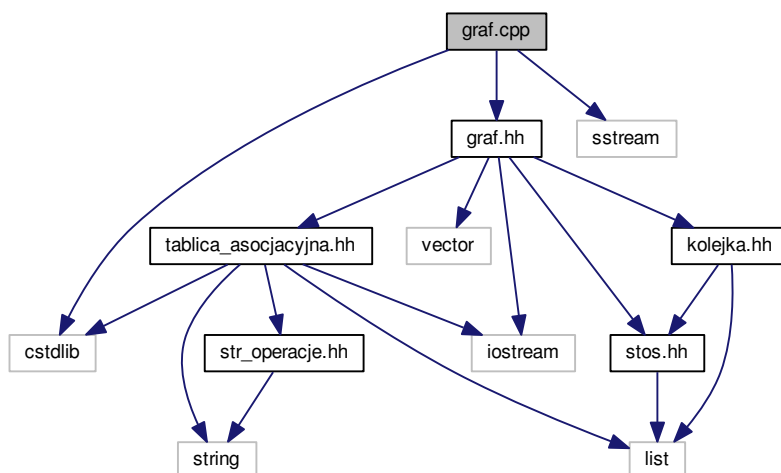
lewy
zaden
prawy

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

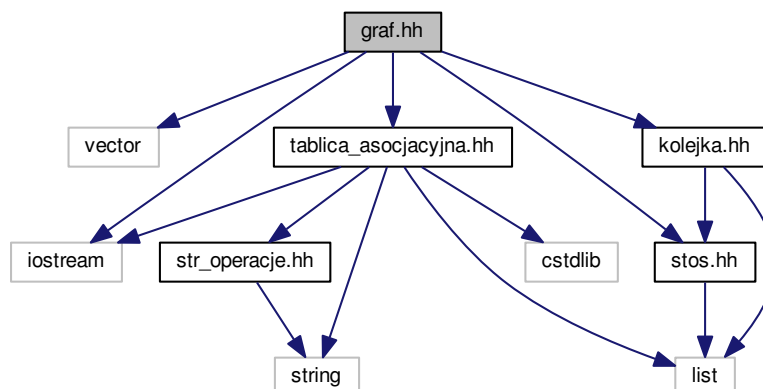
5.4.1.1 `tablica_asocjacyjna<int> vec`

Definicja w linii 4 pliku graf.cpp.

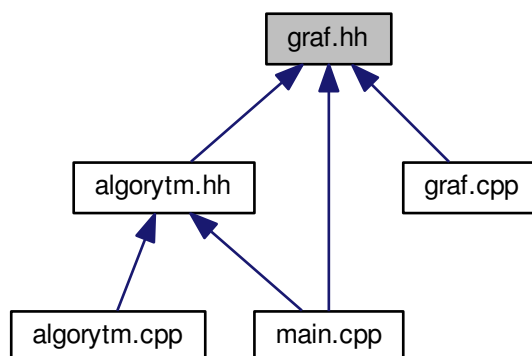
5.5 Dokumentacja pliku graf.hh

```
#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 pojęcie wierzchołka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakie interpretowanie wierzchołka grafu, wedle życzeń użytkownika.

- class [graf](#)

Klasa modeluje pojęcie grafu w oparciu o listę incydencji, Operacje na grafie możliwe są na dwa sposoby \n.

5.5.1 Opis szczegółowy

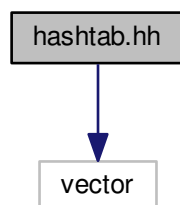
Plik zawiera definicje klasy wierzcholek i klasy graf

Definicja w pliku [graf.hh](#).

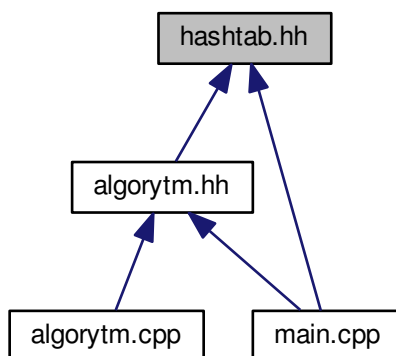
5.6 Dokumentacja pliku hashtab.hh

```
#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 haszującej
- class `hashtab< TYP >`
modeluje tablice haszująca w oparciu o kontener klasy `el_tab`

5.6.1 Opis szczegółowy

Plik zawiera definicje klasy reprezentującej tablice haszująca

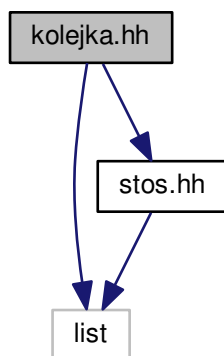
Definicja w pliku [hashtab.hh](#).

5.7 Dokumentacja pliku kolejka.hh

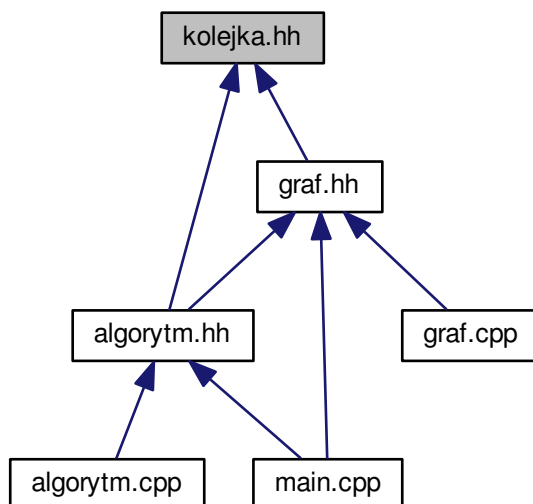
Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby.

```
#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 kolejkę opartą na liście STL.

- class `queue_array< TYP >`

Modeluje kolejkę 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. każdorazowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy kolejka się przepelni

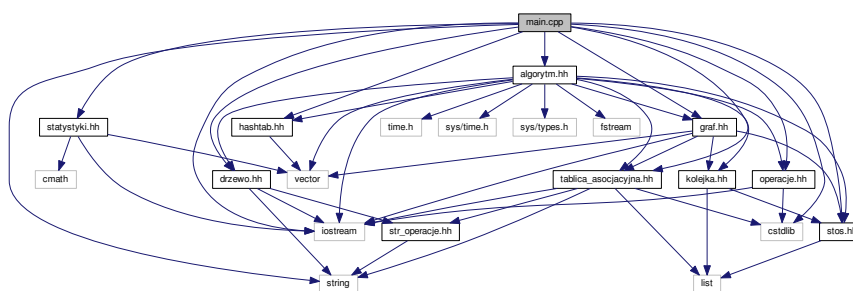
Definicja w pliku `kolejka.hh`.

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

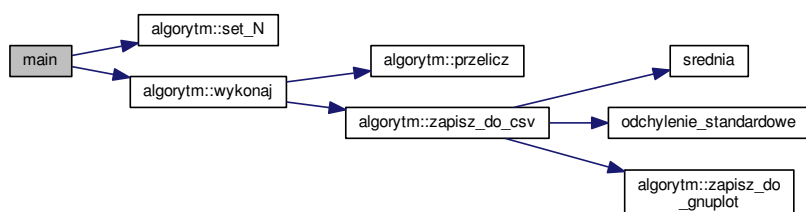
Definicja w pliku `main.cpp`.

5.8.2 Dokumentacja funkcji

5.8.2.1 `int main ()`

Definicja w linii 20 pliku `main.cpp`.

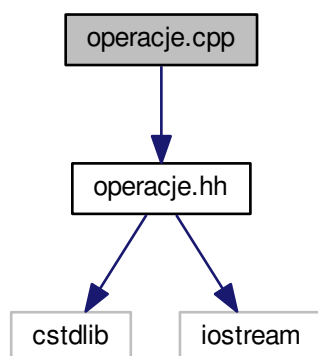
Oto graf wywołań dla tej funkcji:



5.9 Dokumentacja pliku `operacje.cpp`

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

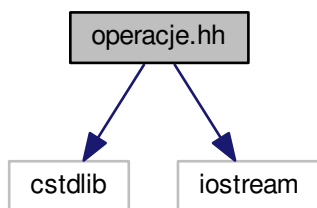
Wykres zależności załączania dla `operacje.cpp`:



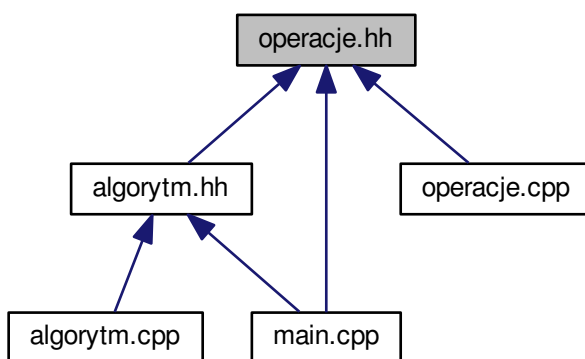
5.10 Dokumentacja pliku `operacje.hh`

```
#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 służące do operacji na niej.

Definicje

- `#define ROZMIAR 9`

5.10.1 Dokumentacja definicji

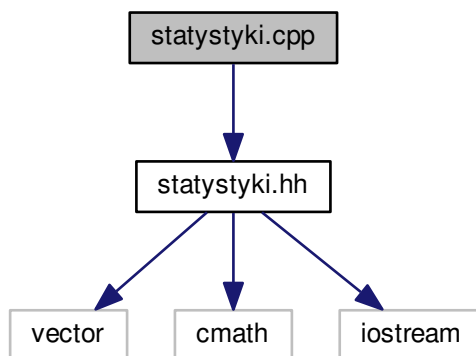
5.10.1.1 `#define ROZMIAR 9`

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

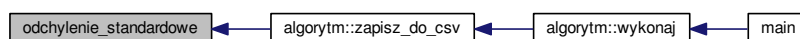
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

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)

funkcja oblicza wartosc srednia

Parametry

<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:

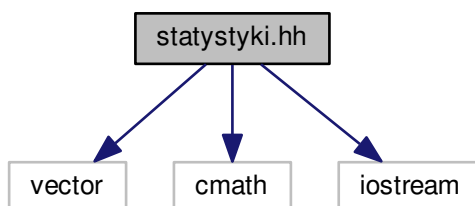


5.12 Dokumentacja pliku statystyki.hh

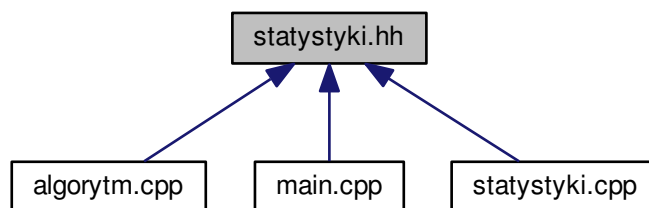
plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk

```
#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 deklaracje funkcji odpowiedzialnych za przeprowadzanie 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

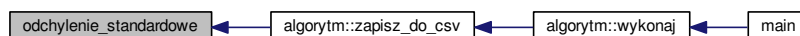
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

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

<i>tab</i>	- kontener zawierający czasy wykonania algorytmu
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

Oto graf wywołań tej funkcji:

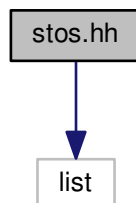


5.13 Dokumentacja pliku stos.hh

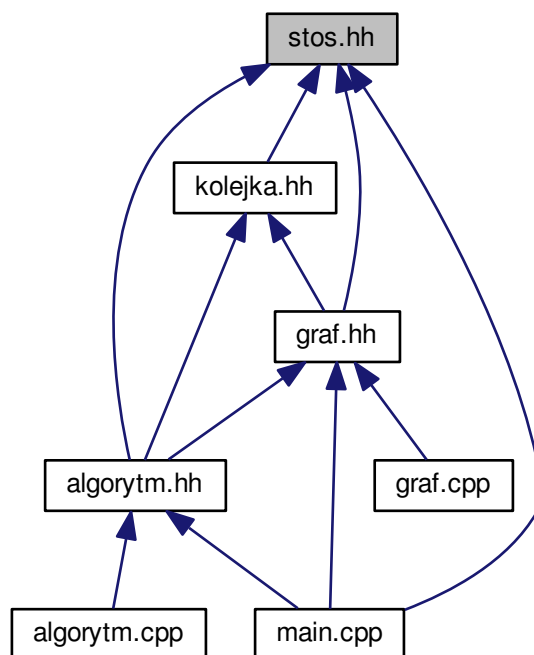
Plik zawiera definicje klasy `Stos` Zaimplementowana na 2 sposoby.

```
#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 liście STL.
- class `stack_array< TYP >`
Modeluje stos w oparciu o tablice.

Wyliczenia

- enum `flag { plus1, x2 }`
typ wyliczeniowy służący do ustawienia sposobu zwiększania pamięci

5.13.1 Opis szczegółowy

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

1. Za pomocą listy.
2. Za pomocą tablicy a. każdorazowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy stos się przepełni

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

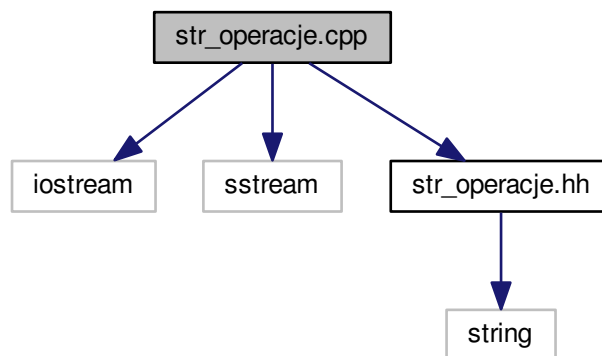
plus1
x2

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 wyżej w porządku alfabetycznym niż 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 wyżej w porządku alfabetycznym niż s2 lub gdy oba stringi są sobie równe, false, gdy s2 jest wyżej w porządku alfabetycznym niż 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 służąca do alfabetycznego porządkowania napisów

Zwraca

true, gdy s1 niżej w porządku alfabetycznym niż s2, false w przeciwnym przypadku

Definicja w linii 21 pliku str_operacje.cpp.

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

Zwraca

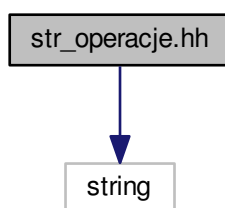
true, gdy s1 jest niżej w porządku alfabetycznym niż s2 lub gdy oba stringi są sobie równe, false, gdy s1 jest wyżej w porządku alfabetycznym niż s2

Definicja w linii 38 pliku str_operacje.cpp.

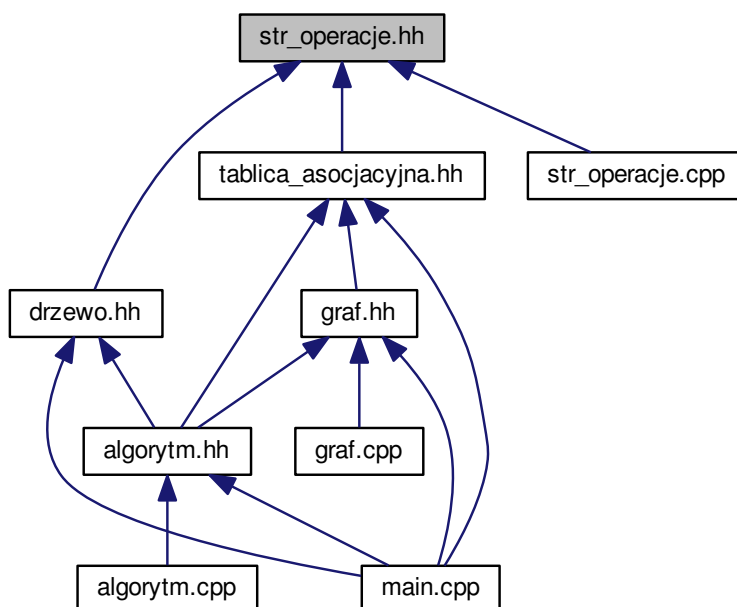
5.15 Dokumentacja pliku str_operacje.hh

```
#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 alfabetycznym 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 niziej 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 niziej w porzadku alfabetycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s1 jest wyzej w porzadku alfabetycznym niz s2

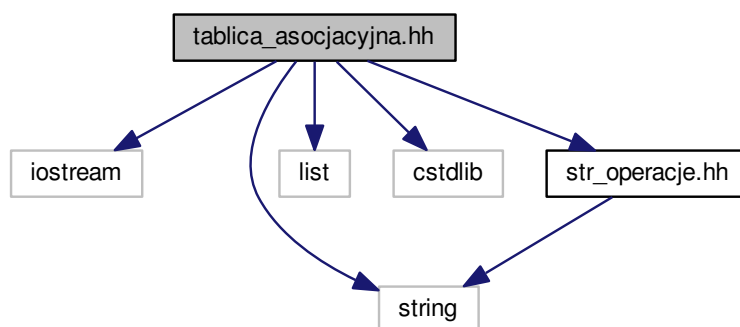
Definicja w linii 38 pliku str_operacje.cpp.

5.16 Dokumentacja pliku strona.dox

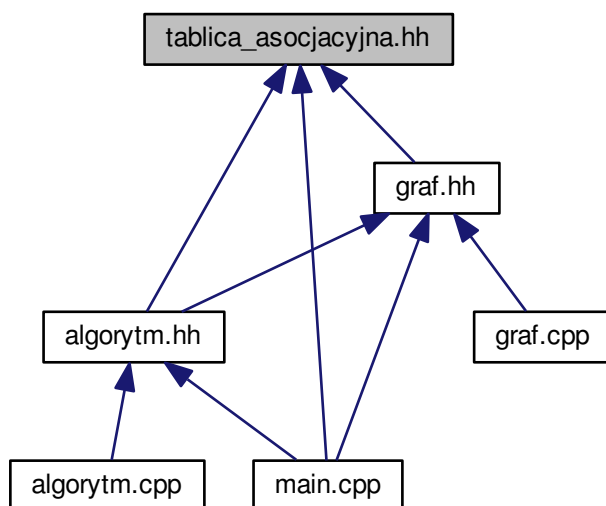
5.17 Dokumentacja pliku tablica_asocjacyjna.hh

```
#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

- ~bst
 - bst, [19](#)
- ~el_tab
 - el_tab, [24](#)
- ~wezel
 - wezel, [95](#)
- algorytm, [7](#)
 - algorytm, [10](#)
 - czas, [16](#)
 - czas1, [16](#)
 - czas2, [16](#)
 - dane, [17](#)
 - dane_wz, [17](#)
 - ile_danych, [10](#)
 - jaki_czas, [10](#)
 - m, [17](#)
 - n, [17](#)
 - op, [17](#)
 - porownaj, [10](#)
 - przelicz, [11](#)
 - set_N, [11](#)
 - wczytaj, [11](#)
 - wczytaj_wzor, [12](#)
 - wlacz_zegar, [12](#)
 - wykonaj, [13](#)
 - wylacz_zegar, [14](#)
 - zapisz_do_csv, [15](#)
 - zapisz_do_gnuplot, [16](#)
- algorytm.cpp, [99](#)
- algorytm.hh, [99](#)
- best_first
 - graf, [27](#)
- bfs
 - graf, [28](#)
- blok
 - tablica_asocjacyjna, [92](#)
- bst, [17](#)
 - ~bst, [19](#)
 - bst, [19](#)
 - d, [20](#)
 - klucze, [20](#)
 - przelicz, [19](#)
 - wczytaj_klucze, [20](#)
- clear
 - queue_array, [71](#)
 - queue_list, [73](#)
 - stack_array, [76](#)
- stack_list, [79](#)
- czas
 - algorytm, [16](#)
- czas1
 - algorytm, [16](#)
- czas2
 - algorytm, [16](#)
- czy_blokada
 - tablica_asocjacyjna, [89](#)
- czy_pusta
 - tablica_asocjacyjna, [89](#)
- czy_sasiad
 - graf, [28](#), [29](#)
- czyisc
 - drzewo, [21](#)
- d
 - bst, [20](#)
 - tab_aso, [87](#)
- dane
 - algorytm, [17](#)
- dane_wz
 - algorytm, [17](#)
- dequeue
 - queue_array, [71](#)
 - queue_list, [74](#)
- dfs
 - graf, [29](#)
- dlugosc
 - hashtab, [50](#)
- dodaj
 - drzewo, [22](#)
 - hashtab, [48](#)
 - tablica_asocjacyjna, [89](#)
- dodaj_element
 - operacje, [61](#)
- dodaj_elementy
 - operacje, [62](#)
- dodaj_krawedz
 - graf, [30](#), [31](#)
- dodaj_syna
 - wezel, [95](#)
- dodaj_wezel
 - drzewo, [22](#)
- dodaj_wierzcholek
 - graf, [31](#), [32](#)
- drzewo
 - czyisc, [21](#)
 - dodaj, [22](#)
 - dodaj_wezel, [22](#)

- drzewo, 21
- korzen, 23
- szukaj, 22
- usun, 22
- wyczysc, 22
- znajdz, 22
- znaleziony, 23
- drzewo< TYP >, 20
- drzewo.hh
 - lewy, 102
 - prawy, 102
 - zaden, 102
- drzewo.hh, 101
- syn, 102
- el_tab
 - ~el_tab, 24
 - el_tab, 24
 - el_tab, 24
 - klucz, 25
 - wart, 25
 - zajety, 25
- el_tab< TYP >, 23
- enqueue
 - queue_array, 71
 - queue_list, 74
- f
 - queue_array, 72
 - stack_array, 78
- flag
 - stos.hh, 116
 - wezel, 96
- found
 - tablica_asocjacyjna, 92
- G1
 - graf_test, 41
- G2
 - graf_test, 41
- G3
 - graf_test, 41
- G4
 - graf_test, 42
- G5
 - graf_test, 42
- G6
 - graf_test, 42
- graf, 25
 - best_first, 27
 - bfs, 28
 - czy_sasiad, 28, 29
 - dfs, 29
 - dodaj_krawedz, 30, 31
 - dodaj_wierzcholek, 31, 32
 - graf, 27
 - lista_incydencji, 38
 - przeszukaj_wezel, 32
 - przeszukaj_wezel_1, 33
 - przeszukaj_wezel_2, 34
 - Q, 38
 - Q0, 38
 - sasiedztwo, 35, 36
 - tab, 38
 - usun_krawedz, 36
 - usun_wierzcholek, 37
 - wierzcholek, 98
 - wyczysc, 37
 - wypisz_liste, 38
- graf.cpp, 103
- vec, 103
- graf.hh, 103
- graf_test, 38
 - G1, 41
 - G2, 41
 - G3, 41
 - G4, 42
 - G5, 42
 - G6, 42
 - graf_test, 40
 - graf_test, 40
 - przelicz, 40
 - typ, 42
 - wczytaj_graf, 41
- h_sort, 42
 - h_sort, 43
 - h_sort, 43
 - przelicz, 43
- h_table, 44
 - h_table, 45
 - h_table, 45
 - klucze, 46
 - przelicz, 46
 - wczytaj_klucze, 46
- hash
 - hashtab, 48
- hashtab
 - dlugosc, 50
 - dodaj, 48
 - hash, 48
 - hashtab, 47
 - tab, 50
 - ustaw_dlugosc, 49
 - usun, 49
 - wypisz, 49
 - znajdz, 49
- hashtab< TYP >, 46
- hashtab.hh, 105
- heap_sort
 - operacje, 62
- id
 - wierzcholek, 98
- ile_danych
 - algorytm, 10
- insert
 - tablica_asocjacyjna, 89

- is_empty
 - queue_array, [72](#)
 - queue_list, [74](#)
 - stack_array, [76](#)
 - stack_list, [79](#)
- jaki_czas
 - algorytm, [10](#)
- key
 - tablica_asocjacyjna, [92](#)
- klucz
 - el_tab, [25](#)
 - wezel, [96](#)
- klucze
 - bst, [20](#)
 - h_table, [46](#)
 - tab_aso, [87](#)
- kolejka.hh, [106](#)
- kolejka_lista, [51](#)
 - kolejka_lista, [52](#)
 - kolejka_lista, [52](#)
 - przelicz, [52](#)
 - qu, [52](#)
- kolejka_tablica, [53](#)
 - kolejka_tablica, [54](#)
 - kolejka_tablica, [54](#)
 - przelicz, [54](#)
 - qu, [54](#)
- korzen
 - drzewo, [23](#)
- lewy
 - drzewo.hh, [102](#)
- lista_incydencji
 - graf, [38](#)
- lsyn
 - wezel, [97](#)
- m
 - algorytm, [17](#)
- m_sort, [55](#)
 - m_sort, [56](#)
 - m_sort, [56](#)
 - przelicz, [56](#)
- main
 - main.cpp, [108](#)
- main.cpp, [107](#)
 - main, [108](#)
- make_heap
 - operacje, [62](#)
- make_node
 - operacje, [63](#)
- merge
 - operacje, [63](#)
- merge_sort
 - operacje, [64](#)
- mnozenie, [56](#)
 - mnozenie, [57](#)
- przelicz, [59](#)
- n
 - algorytm, [17](#)
 - operacje, [67](#)
- odblokuj
 - tablica_asocjacyjna, [89](#)
- odchylenie_standardowe
 - statystyki.cpp, [110](#)
 - statystyki.hh, [112](#)
- odwroc_tablice
 - operacje, [64](#)
- ojciec
 - wezel, [97](#)
- op
 - algorytm, [17](#)
- operacje, [59](#)
 - dodaj_element, [61](#)
 - dodaj_elementy, [62](#)
 - heap_sort, [62](#)
 - make_heap, [62](#)
 - make_node, [63](#)
 - merge, [63](#)
 - merge_sort, [64](#)
 - n, [67](#)
 - odwroc_tablice, [64](#)
 - operacje, [60](#)
 - operator=, [64](#)
 - operator==, [66](#)
 - quick_sort, [66](#)
 - tab, [67](#)
 - zamien_elementy, [66](#)
- operacje.cpp, [108](#)
- operacje.hh, [108](#)
 - ROZMIAR, [109](#)
- operator<
 - str_operacje.cpp, [116](#)
 - str_operacje.hh, [119](#)
- operator<=
 - str_operacje.cpp, [117](#)
 - str_operacje.hh, [119](#)
- operator>
 - str_operacje.cpp, [117](#)
 - str_operacje.hh, [119](#)
- operator>=
 - str_operacje.cpp, [117](#)
 - str_operacje.hh, [119](#)
- operator=
 - operacje, [64](#)
- operator==
 - operacje, [66](#)
 - str_operacje.cpp, [117](#)
 - str_operacje.hh, [119](#)
- plus1
 - stos.hh, [116](#)
- pobierz
 - tablica_asocjacyjna, [89](#)

- pop
 - stack_array, 77
 - stack_list, 79
- porownaj
 - algorytm, 10
- prawy
 - drzewo.hh, 102
- przelicz
 - algorytm, 11
 - bst, 19
 - graf_test, 40
 - h_sort, 43
 - h_table, 46
 - kolejka_lista, 52
 - kolejka_tablica, 54
 - m_sort, 56
 - mnozenie, 59
 - q_sort, 69
 - stos_lista, 81
 - stos_tablica, 83
 - tab_aso, 86
- przeszukaj_wezel
 - graf, 32
- przeszukaj_wezel_1
 - graf, 33
- przeszukaj_wezel_2
 - graf, 34
- psyn
 - wezel, 97
- push
 - stack_array, 77
 - stack_list, 79
- Q
 - graf, 38
- q
 - queue_array, 72
 - queue_list, 75
- Q0
 - graf, 38
- q_sort, 67
 - przelicz, 69
 - q_sort, 69
 - q_sort, 69
- qu
 - kolejka_lista, 52
 - kolejka_tablica, 54
- queue_array
 - clear, 71
 - dequeue, 71
 - enqueue, 71
 - f, 72
 - is_empty, 72
 - q, 72
 - queue_array, 71
 - queue_array, 71
 - s, 72
 - size, 72
 - sp, 72
- queue_array< TYP >, 69
- queue_list
 - clear, 73
 - dequeue, 74
 - enqueue, 74
 - is_empty, 74
 - q, 75
 - size, 74
- queue_list< TYP >, 73
- quick_sort
 - operacje, 66
- ROZMIAR
 - operacje.hh, 109
- s
 - queue_array, 72
 - stack_array, 78
 - tablica_asocjacyjna, 93
- sasiedztwo
 - graf, 35, 36
- set_N
 - algorytm, 11
- size
 - queue_array, 72
 - queue_list, 74
 - stack_array, 77
 - stack_list, 80
- sp
 - queue_array, 72
 - stack_array, 78
 - tablica_asocjacyjna, 93
- srednia
 - statystyki.cpp, 111
 - statystyki.hh, 112
- st
 - stack_array, 78
 - stack_list, 80
- stack_array
 - clear, 76
 - f, 78
 - is_empty, 76
 - pop, 77
 - push, 77
 - s, 78
 - size, 77
 - sp, 78
 - st, 78
 - stack_array, 76
 - stack_array, 76
- stack_array< TYP >, 75
- stack_list
 - clear, 79
 - is_empty, 79
 - pop, 79
 - push, 79
 - size, 80
 - st, 80
- stack_list< TYP >, 78

- statystyki.cpp, 110
 - odchylenie_standardowe, 110
 - srednia, 111
- statystyki.hh, 111
 - odchylenie_standardowe, 112
 - srednia, 112
- stos
 - stos_lista, 82
 - stos_tablica, 84
- stos.hh
 - plus1, 116
 - x2, 116
- stos.hh, 114
 - flag, 116
- stos_lista, 80
 - przelicz, 81
 - stos, 82
 - stos_lista, 81
 - stos_lista, 81
- stos_tablica, 82
 - przelicz, 83
 - stos, 84
 - stos_tablica, 83
 - stos_tablica, 83
- str_operacje.cpp, 116
 - operator<, 116
 - operator<=, 117
 - operator>, 117
 - operator>=, 117
 - operator==, 117
- str_operacje.hh, 118
 - operator<, 119
 - operator<=, 119
 - operator>, 119
 - operator>=, 119
 - operator==, 119
- strona.dox, 120
- syn
 - drzewo.hh, 102
- szukaj
 - drzewo, 22
- tab
 - graf, 38
 - hashtab, 50
 - operacje, 67
- tab_aso, 84
 - d, 87
 - klucze, 87
 - przelicz, 86
 - tab_aso, 86
 - tab_aso, 86
 - wczytaj_klucze, 86
- tablica_asocjacyjna
 - blok, 92
 - czy_blokada, 89
 - czy_pusta, 89
 - dodaj, 89
 - found, 92
 - insert, 89
 - key, 92
 - odblokuj, 89
 - pobierz, 89
 - s, 93
 - sp, 93
 - tablica_asocjacyjna, 88
 - tablica_asocjacyjna, 88
 - ustaw, 90
 - usun, 90
 - value, 93
 - wez, 90
 - wez_id, 91
 - wstaw, 91
 - wypisz, 91
 - zablokuj, 91
 - zlicz_elementy, 91
 - znajdz, 92
- tablica_asocjacyjna< TYP >, 87
- tablica_asocjacyjna.hh, 120
- typ
 - graf_test, 42
- ustaw
 - tablica_asocjacyjna, 90
- ustaw_dlugosc
 - hashtab, 49
- usun
 - drzewo, 22
 - hashtab, 49
 - tablica_asocjacyjna, 90
- usun_krawedz
 - graf, 36
- usun_wierzcholek
 - graf, 37
- value
 - tablica_asocjacyjna, 93
- vec
 - graf.cpp, 103
- waga
 - wierzcholek, 98
- wart
 - el_tab, 25
 - wezel, 97
- wczytaj
 - algorytm, 11
- wczytaj_graf
 - graf_test, 41
- wczytaj_klucze
 - bst, 20
 - h_table, 46
 - tab_aso, 86
- wczytaj_wzor
 - algorytm, 12
- wez
 - tablica_asocjacyjna, 90
- wez_id

tablica_asocjacyjna, 91
wez_klucz
 wezel, 95
wez_wart
 wezel, 96
wezel
 ~wezel, 95
 dodaj_syna, 95
 flag, 96
 klucz, 96
 lsyn, 97
 ojciec, 97
 psyn, 97
 wart, 97
 wez_klucz, 95
 wez_wart, 96
 wezel, 94
 znajdz_nast, 96
wezel< TYP >, 93
wierzcholek, 97
 graf, 98
 id, 98
 waga, 98
 wierzcholek, 98
włącz zegar
 algorytm, 12
wstaw
 tablica_asocjacyjna, 91
wyczyść
 drzewo, 22
 graf, 37
wykonaj
 algorytm, 13
wylłącz zegar
 algorytm, 14
wypisz
 hashtab, 49
 tablica_asocjacyjna, 91
wypisz_liste
 graf, 38

x2
 stos.hh, 116

zablokuj
 tablica_asocjacyjna, 91
zaden
 drzewo.hh, 102
zajety
 el_tab, 25
zamień elementy
 operacje, 66
zapisz_do_csv
 algorytm, 15
zapisz_do_gnuplot
 algorytm, 16
zlicz elementy
 tablica_asocjacyjna, 91
znajdz
 drzewo, 22
 hashtab, 49
 tablica_asocjacyjna, 92
znajdz_nast
 wezel, 96
znaleziony
 drzewo, 23