

PAMSI - pwilkosz  
1.0

Wygenerowano przez Doxygen 1.8.6

N, 11 maj 2014 23:21:19



# Spis treści

<b>1 Indeks hierarchiczny</b>	<b>1</b>
1.1 Hierarchia klas . . . . .	1
<b>2 Indeks klas</b>	<b>3</b>
2.1 Lista klas . . . . .	3
<b>3 Indeks plików</b>	<b>5</b>
3.1 Lista plików . . . . .	5
<b>4 Dokumentacja klas</b>	<b>7</b>
4.1 Dokumentacja klasy algorytm . . . . .	7
4.1.1 Opis szczegółowy . . . . .	10
4.1.2 Dokumentacja konstruktora i destruktor	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 astar	17
4.2.1	Opis szczegółowy	19
4.2.2	Dokumentacja konstruktora i destruktora	19
4.2.2.1	astar	19
4.2.3	Dokumentacja funkcji składowych	19
4.2.3.1	przelicz	19
4.2.3.2	wczytaj_graf	20
4.2.4	Dokumentacja atrybutów składowych	20
4.2.4.1	G1	20
4.2.4.2	G2	21
4.2.4.3	G3	21
4.2.4.4	G4	21
4.2.4.5	G5	21
4.2.4.6	G6	21
4.3	Dokumentacja klasy bst	21
4.3.1	Opis szczegółowy	23
4.3.2	Dokumentacja konstruktora i destruktora	23
4.3.2.1	bst	23
4.3.2.2	~bst	23
4.3.3	Dokumentacja funkcji składowych	23
4.3.3.1	przelicz	23
4.3.3.2	wczytaj_klucze	23
4.3.4	Dokumentacja atrybutów składowych	23
4.3.4.1	d	23
4.3.4.2	klucze	24
4.4	Dokumentacja szablonu klasy drzewo< TYP >	24
4.4.1	Opis szczegółowy	24
4.4.2	Dokumentacja konstruktora i destruktora	25
4.4.2.1	drzewo	25
4.4.2.2	drzewo	25
4.4.3	Dokumentacja funkcji składowych	25
4.4.3.1	czysc	25
4.4.3.2	dodaj	25
4.4.3.3	dodaj_wezel	25
4.4.3.4	szukaj	25
4.4.3.5	usun	26

4.4.3.6	wyczysc	26
4.4.3.7	znajdz	26
4.4.4	Dokumentacja atrybutów składowych	26
4.4.4.1	korzen	26
4.4.4.2	znaleziony	27
4.5	Dokumentacja szablonu klasy el_tab< TYP >	27
4.5.1	Opis szczegółowy	27
4.5.2	Dokumentacja konstruktora i destruktora	28
4.5.2.1	el_tab	28
4.5.2.2	~el_tab	28
4.5.3	Dokumentacja atrybutów składowych	28
4.5.3.1	klucz	28
4.5.3.2	wart	28
4.5.3.3	zajety	28
4.6	Dokumentacja klasy graf	28
4.6.1	Opis szczegółowy	31
4.6.2	Dokumentacja konstruktora i destruktora	31
4.6.2.1	graf	31
4.6.3	Dokumentacja funkcji składowych	31
4.6.3.1	a_star	31
4.6.3.2	best_first	33
4.6.3.3	bfs	34
4.6.3.4	czy_sasiad	35
4.6.3.5	czy_sasiad	36
4.6.3.6	dfs	36
4.6.3.7	dodaj_krawedz	37
4.6.3.8	dodaj_krawedz	38
4.6.3.9	dodaj_krawedz	38
4.6.3.10	dodaj_wierzcholek	39
4.6.3.11	dodaj_wierzcholek	39
4.6.3.12	dodaj_wierzcholek	40
4.6.3.13	przeszukaj_wezel	40
4.6.3.14	przeszukaj_wezel_1	41
4.6.3.15	przeszukaj_wezel_2	42
4.6.3.16	rysuj	43
4.6.3.17	sasiedztwo	44
4.6.3.18	sasiedztwo	44
4.6.3.19	usun_krawedz	44
4.6.3.20	usun_krawedz	45
4.6.3.21	usun_wierzcholek	46

4.6.3.22	usun_wierzcholek	46
4.6.3.23	wyczysc	47
4.6.3.24	wypisz_liste	47
4.6.4	Dokumentacja atrybutów składowych	47
4.6.4.1	dist	47
4.6.4.2	est	47
4.6.4.3	lista_incydencji	47
4.6.4.4	poprzednik	47
4.6.4.5	Q	47
4.6.4.6	tab	48
4.6.4.7	w_x	48
4.6.4.8	w_y	48
4.7	Dokumentacja klasy graf_test	48
4.7.1	Opis szczegółowy	50
4.7.2	Dokumentacja konstruktora i destruktora	50
4.7.2.1	graf_test	50
4.7.3	Dokumentacja funkcji składowych	50
4.7.3.1	przelicz	50
4.7.3.2	wczytaj_graf	50
4.7.4	Dokumentacja atrybutów składowych	51
4.7.4.1	G1	51
4.7.4.2	G2	51
4.7.4.3	G3	51
4.7.4.4	G4	51
4.7.4.5	G5	51
4.7.4.6	G6	51
4.7.4.7	typ	51
4.8	Dokumentacja klasy h_sort	52
4.8.1	Opis szczegółowy	53
4.8.2	Dokumentacja konstruktora i destruktora	53
4.8.2.1	h_sort	53
4.8.3	Dokumentacja funkcji składowych	53
4.8.3.1	przelicz	53
4.9	Dokumentacja klasy h_table	53
4.9.1	Opis szczegółowy	55
4.9.2	Dokumentacja konstruktora i destruktora	55
4.9.2.1	h_table	55
4.9.3	Dokumentacja funkcji składowych	55
4.9.3.1	przelicz	55
4.9.3.2	wczytaj_klucze	55

4.9.4	Dokumentacja atrybutów składowych	55
4.9.4.1	klucze	56
4.10	Dokumentacja szablonu klasy hashtab< TYP >	56
4.10.1	Opis szczegółowy	56
4.10.2	Dokumentacja konstruktora i destruktora	57
4.10.2.1	hashtab	57
4.10.2.2	hashtab	57
4.10.3	Dokumentacja funkcji składowych	57
4.10.3.1	dodaj	57
4.10.3.2	hash	58
4.10.3.3	ustaw_dlugosc	58
4.10.3.4	usun	59
4.10.3.5	wypisz	60
4.10.3.6	znajdz	60
4.10.4	Dokumentacja atrybutów składowych	61
4.10.4.1	dlugosc	61
4.10.4.2	tab	61
4.11	Dokumentacja klasy kolejka_lista	61
4.11.1	Opis szczegółowy	62
4.11.2	Dokumentacja konstruktora i destruktora	62
4.11.2.1	kolejka_lista	62
4.11.3	Dokumentacja funkcji składowych	62
4.11.3.1	przelicz	62
4.11.4	Dokumentacja atrybutów składowych	63
4.11.4.1	qu	63
4.12	Dokumentacja klasy kolejka_tablica	63
4.12.1	Opis szczegółowy	64
4.12.2	Dokumentacja konstruktora i destruktora	64
4.12.2.1	kolejka_tablica	64
4.12.3	Dokumentacja funkcji składowych	64
4.12.3.1	przelicz	65
4.12.4	Dokumentacja atrybutów składowych	65
4.12.4.1	qu	65
4.13	Dokumentacja klasy m_sort	65
4.13.1	Opis szczegółowy	66
4.13.2	Dokumentacja konstruktora i destruktora	67
4.13.2.1	m_sort	67
4.13.3	Dokumentacja funkcji składowych	67
4.13.3.1	przelicz	67
4.14	Dokumentacja klasy mnozenie	67

4.14.1	Opis szczegółowy . . . . .	68
4.14.2	Dokumentacja konstruktora i destruktora . . . . .	68
4.14.2.1	mnozenie . . . . .	69
4.14.3	Dokumentacja funkcji składowych . . . . .	70
4.14.3.1	przelicz . . . . .	70
4.15	Dokumentacja klasy operacje . . . . .	70
4.15.1	Opis szczegółowy . . . . .	71
4.15.2	Dokumentacja konstruktora i destruktora . . . . .	71
4.15.2.1	operacje . . . . .	71
4.15.2.2	operacje . . . . .	71
4.15.3	Dokumentacja funkcji składowych . . . . .	72
4.15.3.1	dodaj_element . . . . .	72
4.15.3.2	dodaj_elementy . . . . .	73
4.15.3.3	heap_sort . . . . .	73
4.15.3.4	make_heap . . . . .	73
4.15.3.5	make_node . . . . .	74
4.15.3.6	merge . . . . .	74
4.15.3.7	merge_sort . . . . .	75
4.15.3.8	odwroc_tablice . . . . .	75
4.15.3.9	operator= . . . . .	76
4.15.3.10	operator== . . . . .	77
4.15.3.11	operator[] . . . . .	77
4.15.3.12	quick_sort . . . . .	77
4.15.3.13	zamien_elementy . . . . .	78
4.15.4	Dokumentacja atrybutów składowych . . . . .	78
4.15.4.1	n . . . . .	78
4.15.4.2	tab . . . . .	78
4.16	Dokumentacja klasy q_sort . . . . .	78
4.16.1	Opis szczegółowy . . . . .	79
4.16.2	Dokumentacja konstruktora i destruktora . . . . .	80
4.16.2.1	q_sort . . . . .	80
4.16.3	Dokumentacja funkcji składowych . . . . .	80
4.16.3.1	przelicz . . . . .	80
4.17	Dokumentacja szablonu klasy queue_array< TYP > . . . . .	80
4.17.1	Opis szczegółowy . . . . .	81
4.17.2	Dokumentacja konstruktora i destruktora . . . . .	82
4.17.2.1	queue_array . . . . .	82
4.17.2.2	queue_array . . . . .	82
4.17.3	Dokumentacja funkcji składowych . . . . .	82
4.17.3.1	clear . . . . .	82



4.17.3.2	dequeue	82
4.17.3.3	enqueue	82
4.17.3.4	is_empty	83
4.17.3.5	size	83
4.17.4	Dokumentacja atrybutów składowych	83
4.17.4.1	f	83
4.17.4.2	q	83
4.17.4.3	s	83
4.17.4.4	sp	84
4.18	Dokumentacja szablonu klasy queue_list< TYP >	84
4.18.1	Opis szczegółowy	84
4.18.2	Dokumentacja funkcji składowych	84
4.18.2.1	clear	84
4.18.2.2	dequeue	85
4.18.2.3	enqueue	85
4.18.2.4	is_empty	85
4.18.2.5	size	85
4.18.3	Dokumentacja atrybutów składowych	86
4.18.3.1	q	86
4.19	Dokumentacja szablonu klasy stack_array< TYP >	86
4.19.1	Opis szczegółowy	87
4.19.2	Dokumentacja konstruktora i destruktoru	87
4.19.2.1	stack_array	87
4.19.2.2	stack_array	87
4.19.3	Dokumentacja funkcji składowych	87
4.19.3.1	clear	87
4.19.3.2	is_empty	87
4.19.3.3	pop	88
4.19.3.4	push	88
4.19.3.5	size	88
4.19.4	Dokumentacja atrybutów składowych	89
4.19.4.1	f	89
4.19.4.2	s	89
4.19.4.3	sp	89
4.19.4.4	st	89
4.20	Dokumentacja szablonu klasy stack_list< TYP >	89
4.20.1	Opis szczegółowy	90
4.20.2	Dokumentacja funkcji składowych	90
4.20.2.1	clear	90
4.20.2.2	is_empty	90

4.20.2.3	pop	90
4.20.2.4	push	90
4.20.2.5	size	91
4.20.3	Dokumentacja atrybutów składowych	91
4.20.3.1	st	91
4.21	Dokumentacja klasy stos_lista	91
4.21.1	Opis szczegółowy	92
4.21.2	Dokumentacja konstruktora i destruktor	92
4.21.2.1	stos_lista	92
4.21.3	Dokumentacja funkcji składowych	92
4.21.3.1	przelicz	92
4.21.4	Dokumentacja atrybutów składowych	93
4.21.4.1	stos	93
4.22	Dokumentacja klasy stos_tablica	93
4.22.1	Opis szczegółowy	94
4.22.2	Dokumentacja konstruktora i destruktor	94
4.22.2.1	stos_tablica	94
4.22.3	Dokumentacja funkcji składowych	94
4.22.3.1	przelicz	95
4.22.4	Dokumentacja atrybutów składowych	95
4.22.4.1	stos	95
4.23	Dokumentacja klasy tab_aso	95
4.23.1	Opis szczegółowy	97
4.23.2	Dokumentacja konstruktora i destruktor	97
4.23.2.1	tab_aso	97
4.23.3	Dokumentacja funkcji składowych	97
4.23.3.1	przelicz	97
4.23.3.2	wczytaj_klucze	97
4.23.4	Dokumentacja atrybutów składowych	98
4.23.4.1	d	98
4.23.4.2	klucze	98
4.24	Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >	98
4.24.1	Opis szczegółowy	99
4.24.2	Dokumentacja konstruktora i destruktor	100
4.24.2.1	tablica_asocjacyjna	100
4.24.3	Dokumentacja funkcji składowych	100
4.24.3.1	czy_blokada	100
4.24.3.2	czy_pusta	100
4.24.3.3	dodaj	100
4.24.3.4	insert	101

4.24.3.5	odblokuj	101
4.24.3.6	pobierz	101
4.24.3.7	ustaw	101
4.24.3.8	usun	102
4.24.3.9	wez	102
4.24.3.10	wez_id	103
4.24.3.11	wstaw	103
4.24.3.12	wyczysc	103
4.24.3.13	wypisz	104
4.24.3.14	zablokuj	104
4.24.3.15	zlicz_elementy	104
4.24.3.16	znajdz	105
4.24.3.17	znajdz	105
4.24.4	Dokumentacja atrybutów składowych	105
4.24.4.1	blok	105
4.24.4.2	found	105
4.24.4.3	key	105
4.24.4.4	s	105
4.24.4.5	sp	106
4.24.4.6	value	106
4.25	Dokumentacja szablonu klasy wezel< TYP >	106
4.25.1	Opis szczegółowy	107
4.25.2	Dokumentacja konstruktora i destruktora	107
4.25.2.1	wezel	107
4.25.2.2	wezel	107
4.25.2.3	~wezel	107
4.25.3	Dokumentacja funkcji składowych	108
4.25.3.1	dodaj_syna	108
4.25.3.2	wez_klucz	108
4.25.3.3	wez_wart	108
4.25.3.4	znajdz_nast	109
4.25.4	Dokumentacja atrybutów składowych	109
4.25.4.1	flag	109
4.25.4.2	klucz	109
4.25.4.3	lsyn	109
4.25.4.4	ojciec	109
4.25.4.5	psyn	109
4.25.4.6	wart	109
4.26	Dokumentacja klasy wierzcholek	110
4.26.1	Opis szczegółowy	110

4.26.2	Dokumentacja konstruktora i destruktora . . . . .	110
4.26.2.1	wierzcholek . . . . .	110
4.26.3	Dokumentacja przyjaciół i funkcji związanych . . . . .	110
4.26.3.1	graf . . . . .	110
4.26.4	Dokumentacja atrybutów składowych . . . . .	111
4.26.4.1	id . . . . .	111
4.26.4.2	waga . . . . .	111
<b>5</b>	<b>Dokumentacja plików</b>	<b>113</b>
5.1	Dokumentacja pliku algorytm.cpp . . . . .	113
5.1.1	Opis szczegółowy . . . . .	113
5.2	Dokumentacja pliku algorytm.hh . . . . .	113
5.2.1	Opis szczegółowy . . . . .	115
5.3	Dokumentacja pliku drzewo.hh . . . . .	115
5.3.1	Opis szczegółowy . . . . .	116
5.3.2	Dokumentacja typów wyliczanych . . . . .	116
5.3.2.1	syn . . . . .	116
5.4	Dokumentacja pliku graf.cpp . . . . .	117
5.4.1	Dokumentacja zmiennych . . . . .	117
5.4.1.1	vec . . . . .	117
5.5	Dokumentacja pliku graf.hh . . . . .	117
5.5.1	Opis szczegółowy . . . . .	118
5.6	Dokumentacja pliku hashtab.hh . . . . .	119
5.6.1	Opis szczegółowy . . . . .	119
5.7	Dokumentacja pliku kolejka.hh . . . . .	120
5.7.1	Opis szczegółowy . . . . .	121
5.8	Dokumentacja pliku main.cpp . . . . .	121
5.8.1	Opis szczegółowy . . . . .	121
5.8.2	Dokumentacja funkcji . . . . .	122
5.8.2.1	main . . . . .	122
5.9	Dokumentacja pliku operacje.cpp . . . . .	122
5.10	Dokumentacja pliku operacje.hh . . . . .	123
5.10.1	Dokumentacja definicji . . . . .	124
5.10.1.1	ROZMIAR . . . . .	124
5.11	Dokumentacja pliku simplex.hh . . . . .	124
5.12	Dokumentacja pliku statystyki.cpp . . . . .	125
5.12.1	Dokumentacja funkcji . . . . .	125
5.12.1.1	odchylenie_standardowe . . . . .	125
5.12.1.2	srednia . . . . .	126
5.13	Dokumentacja pliku statystyki.hh . . . . .	126

5.13.1	Opis szczegółowy . . . . .	127
5.13.2	Dokumentacja funkcji . . . . .	127
5.13.2.1	odchylenie_standardowe . . . . .	127
5.13.2.2	srednia . . . . .	128
5.14	Dokumentacja pliku stos.hh . . . . .	129
5.14.1	Opis szczegółowy . . . . .	130
5.14.2	Dokumentacja typów wyliczanych . . . . .	131
5.14.2.1	flag . . . . .	131
5.15	Dokumentacja pliku str_operacje.cpp . . . . .	131
5.15.1	Dokumentacja funkcji . . . . .	131
5.15.1.1	operator< . . . . .	131
5.15.1.2	operator<= . . . . .	132
5.15.1.3	operator== . . . . .	132
5.15.1.4	operator> . . . . .	132
5.15.1.5	operator>= . . . . .	132
5.16	Dokumentacja pliku str_operacje.hh . . . . .	133
5.16.1	Dokumentacja funkcji . . . . .	134
5.16.1.1	operator< . . . . .	134
5.16.1.2	operator<= . . . . .	134
5.16.1.3	operator== . . . . .	134
5.16.1.4	operator> . . . . .	134
5.16.1.5	operator>= . . . . .	134
5.17	Dokumentacja pliku strona.dox . . . . .	135
5.18	Dokumentacja pliku tablica_asocjacyjna.hh . . . . .	135
5.18.1	Opis szczegółowy . . . . .	136
<b>Indeks</b>		<b>137</b>



# Rozdział 1

## Indeks hierarchiczny

### 1.1 Hierarchia klas

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

algorytm . . . . .	7
astar . . . . .	17
bst . . . . .	21
graf_test . . . . .	48
h_sort . . . . .	52
h_table . . . . .	53
kolejka_lista . . . . .	61
kolejka_tablica . . . . .	63
m_sort . . . . .	65
mnozenie . . . . .	67
q_sort . . . . .	78
stos_lista . . . . .	91
stos_tablica . . . . .	93
tab_aso . . . . .	95
drzewo< TYP > . . . . .	24
drzewo< float > . . . . .	24
el_tab< TYP > . . . . .	27
graf . . . . .	28
hashtab< TYP > . . . . .	56
operacje . . . . .	70
queue_array< TYP > . . . . .	80
queue_array< float > . . . . .	80
queue_list< TYP > . . . . .	84
queue_list< float > . . . . .	84
stack_array< TYP > . . . . .	86
stack_array< float > . . . . .	86
stack_array< int > . . . . .	86
stack_list< TYP > . . . . .	89
stack_list< float > . . . . .	89
tablica_asocjacyjna< TYP > . . . . .	98
tablica_asocjacyjna< bool > . . . . .	98
tablica_asocjacyjna< float > . . . . .	98
wezel< TYP > . . . . .	106
wezel< float > . . . . .	106
wierzcholek . . . . .	110





## Rozdział 2

# Indeks klas

### 2.1 Lista klas

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

<a href="#">algorytm</a>	Definicja klasy algorytm Jest to klasa bazowa, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym . . . . .	7
<a href="#">astar</a>	. . . . .	17
<a href="#">bst</a>	Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania . . . . .	21
<a href="#">drzewo&lt; TYP &gt;</a>	Modeluje binarne drzewo przeszukiwania . . . . .	24
<a href="#">el_tab&lt; TYP &gt;</a>	Pojedynczy element tablicy haszującej . . . . .	27
<a href="#">graf</a>	Klasa modeluje pojęcie grafu w oparciu o listę incydencji, Operacje na grafie możliwe są na dwa sposoby \n . . . . .	28
<a href="#">graf_test</a>	Modeluje strukturę grafów użytych do badań . . . . .	48
<a href="#">h_sort</a>	Klasa reprezentuje dane poddane sortowaniu przez kopcowanie . . . . .	52
<a href="#">h_table</a>	Modeluje tablicę haszującą przeznaczoną do testowania szybkości wyszukiwania . . . . .	53
<a href="#">hashtab&lt; TYP &gt;</a>	Modeluje tablicę haszującą w oparciu o kontener klasy <a href="#">el_tab</a> . . . . .	56
<a href="#">kolejka_lista</a>	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	61
<a href="#">kolejka_tablica</a>	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	63
<a href="#">m_sort</a>	Klasa reprezentuje dane poddane sortowaniu przez scalanie . . . . .	65
<a href="#">mnozenie</a>	Modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2 . . . . .	67
<a href="#">operacje</a>	Klasa modeluje tablicę z danymi i metody służące do operacji na niej . . . . .	70
<a href="#">q_sort</a>	Klasa reprezentuje dane poddane sortowaniu szybkemu . . . . .	78
<a href="#">queue_array&lt; TYP &gt;</a>	Modeluje kolejkę w oparciu o tablicę . . . . .	80
<a href="#">queue_list&lt; TYP &gt;</a>	Modeluje kolejkę opartą na liście STL . . . . .	84

<a href="#">stack_array&lt; TYP &gt;</a>	
Modeluje stos w oparciu o tablice . . . . .	86
<a href="#">stack_list&lt; TYP &gt;</a>	
Modeluje stos oparty na liscie STL . . . . .	89
<a href="#">stos_lista</a>	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	91
<a href="#">stos_tablica</a>	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	93
<a href="#">tab_aso</a>	
Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania . . . . .	95
<a href="#">tablica_asocjacyjna&lt; TYP &gt;</a>	
Klasa modeluje tablice asocjacyjna . . . . .	98
<a href="#">wezel&lt; TYP &gt;</a>	
Modeluje pojedynczy wezel drzewa . . . . .	106
<a href="#">wierzcholek</a>	
Klasa modeluje pojecie wierzchołka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojaki interpretowanie wierzchołka grafu, wedle życzeń użytkownika . . . . .	110

## Rozdział 3

# Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">algorytm.cpp</a>	Plik zawiera definicje metod klas zdefiniowanych w pliku <a href="#">algorytm.hh</a> . . . . .	113
<a href="#">algorytm.hh</a>	Definicja klas wykonujących operacje na zestawie danych wejściowych . . . . .	113
<a href="#">drzewo.hh</a>	. . . . .	115
<a href="#">graf.cpp</a>	. . . . .	117
<a href="#">graf.hh</a>	. . . . .	117
<a href="#">hashtab.hh</a>	. . . . .	119
<a href="#">kolejka.hh</a>	Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby . . . . .	120
<a href="#">main.cpp</a>	Plik glowny . . . . .	121
<a href="#">operacje.cpp</a>	. . . . .	122
<a href="#">operacje.hh</a>	. . . . .	123
<a href="#">simplex.hh</a>	. . . . .	124
<a href="#">statystyki.cpp</a>	. . . . .	125
<a href="#">statystyki.hh</a>	Plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk . . . . .	126
<a href="#">stos.hh</a>	Plik zawiera definicje klasy Stos Zaimplementowana na 2 sposoby . . . . .	129
<a href="#">str_operacje.cpp</a>	. . . . .	131
<a href="#">str_operacje.hh</a>	. . . . .	133
<a href="#">tablica_asocjacyjna.hh</a>	. . . . .	135



## 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

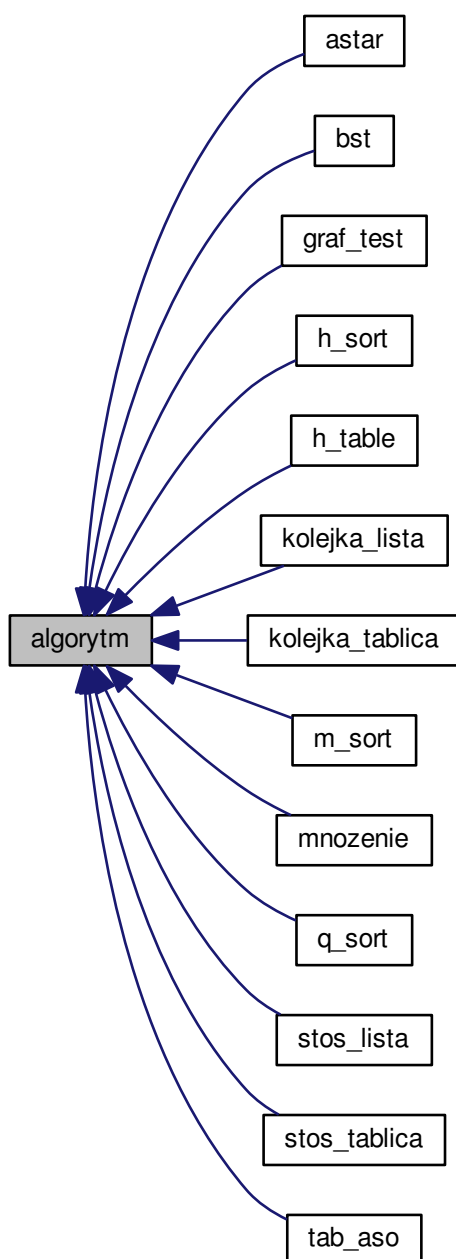
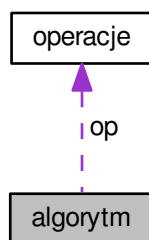


Diagram współpracy dla algorytm:



## 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.

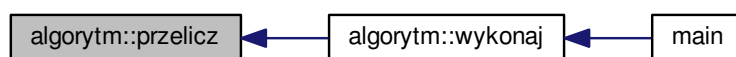
#### 4.1.3.4 float algorytm::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadany algorytmem.

Reimplementowana w [astar](#), [graf\\_test](#), [tab\\_aso](#), [h\\_table](#), [bst](#), [m\\_sort](#), [h\\_sort](#), [q\\_sort](#), [kolejka\\_lista](#), [kolejka\\_tablica](#), [stos\\_lista](#), [stos\\_tablica](#) i [mnozenie](#).

Definicja w linii 9 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

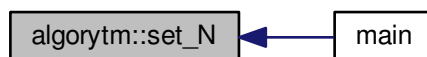


#### 4.1.3.5 void algorytm::set\_N ( int wart ) [inline]

metoda ustawia wartosc n

Definicja w linii 94 pliku algorytm.hh.

Oto graf wywoływań tej funkcji:



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

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

**Parametry**

<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 ( )**

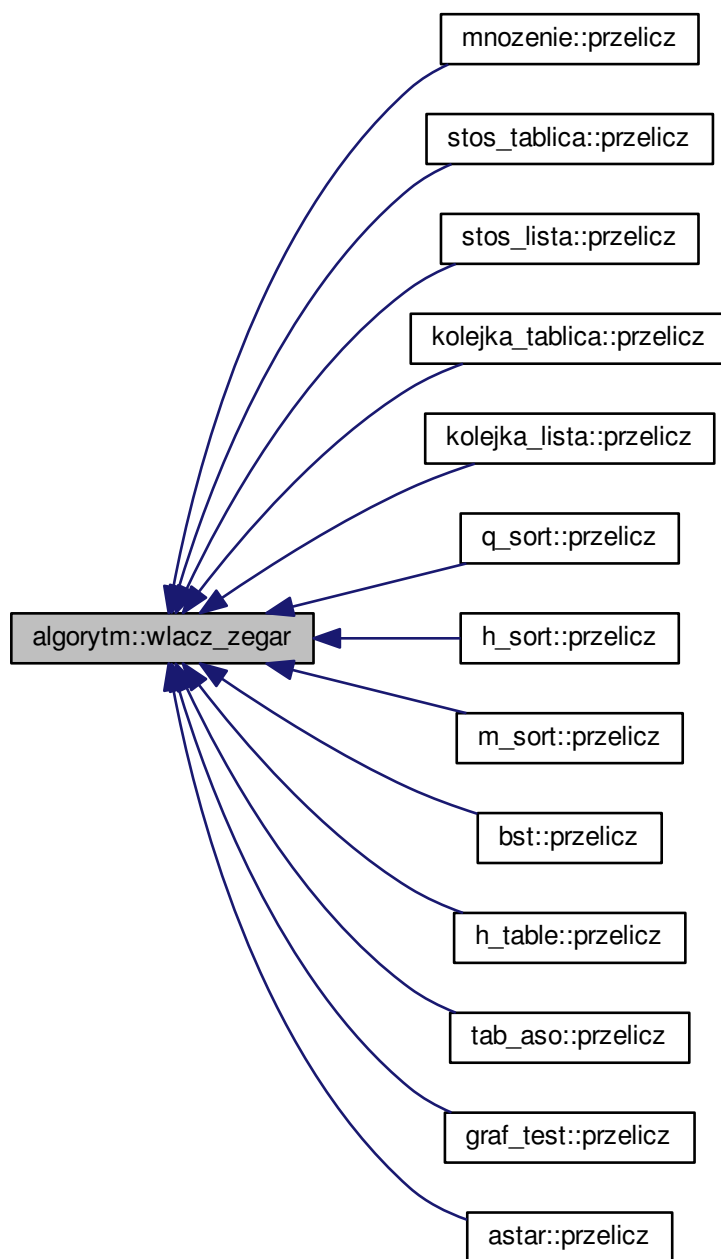
Metoda włącza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `start`.

**Zwraca**

`start` - zmienna pamiętająca czas poprzedzający wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywołań tej funkcji:

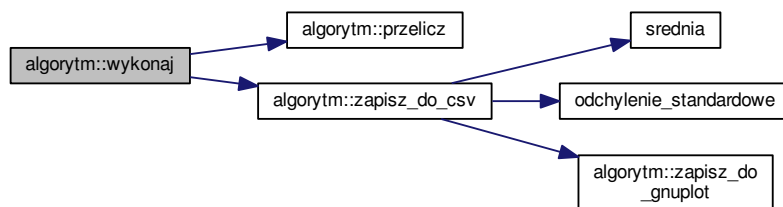


#### 4.1.3.9 void algorytm::wykonaj ( ofstream & out )

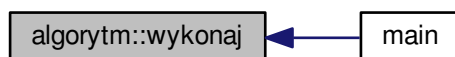
funkcja dokonuje operacji na pliku wejściowym, wywołuje metody odpowiedzialne za pomiar czasu oraz za porównanie wyniku operacji z plikiem wzorcowym

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 ( )

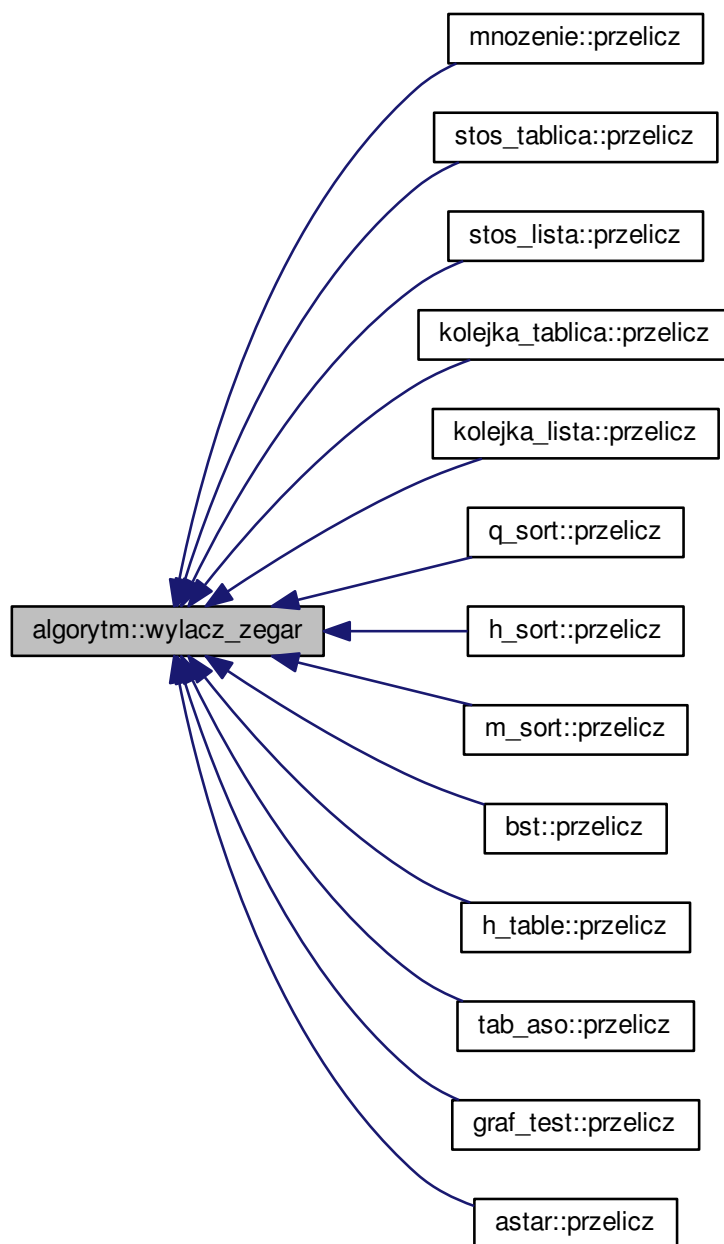
Metoda wyacza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `end`.

**Zwraca**

`end` - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 49 pliku `algorytm.cpp`.

Oto graf wywołań tej funkcji:

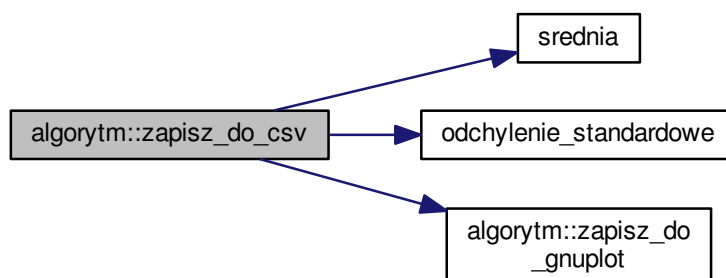


#### 4.1.3.11 void algorytm::zapisz\_do\_csv ( ofstream & out )

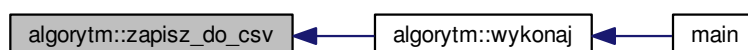
Metoda zapisuje tablice `czas` do pliku `wyjście.csv`.

Definicja w linii 62 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

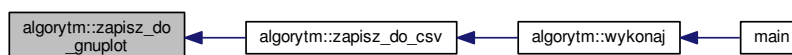


#### 4.1.3.12 void algorytm::zapisz\_do\_gnuplot ( ofstream & out, float sr, float od )

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 astar

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

Diagram dziedziczenia dla astar

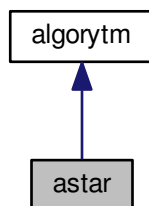
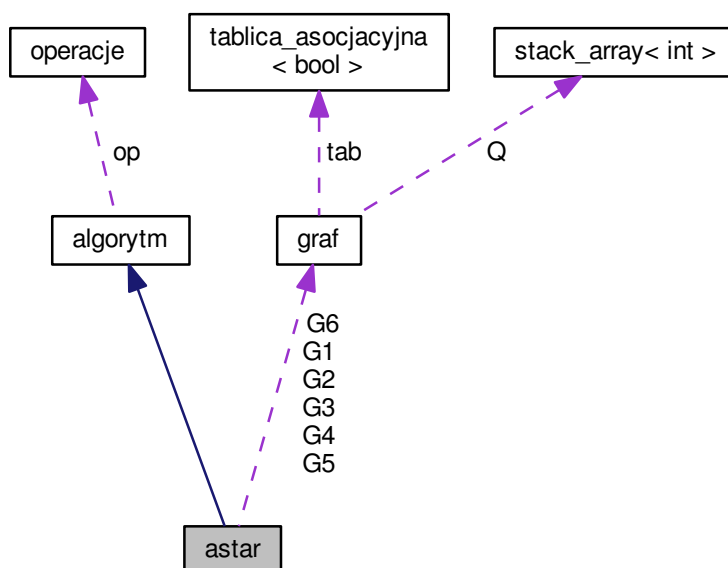


Diagram współpracy dla astar:



## Metody publiczne

- void `wczytaj_graf()`  
*na podstawie danych z pliku, tworzone sa grafy*
- `astar` (ifstream &plik1, ifstream &plik2, int N, int M)  
*konstruktor*
- float `przelicz()`  
*Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadany algorytm.*



### Atrybuty prywatne

- [graf G1](#)
- [graf G2](#)
- [graf G3](#)
- [graf G4](#)
- [graf G5](#)
- [graf G6](#)

## Dodatkowe Dziedziczone Składowe

### 4.2.1 Opis szczegółowy

Definicja w linii 303 pliku algorytm.hh.

### 4.2.2 Dokumentacja konstruktora i destruktor

4.2.2.1 `astar::astar ( ifstream & plik1, ifstream & plik2, int N, int M )` `[inline]`

konstruktor

Definicja w linii 309 pliku algorytm.hh.

### 4.2.3 Dokumentacja funkcji składowych

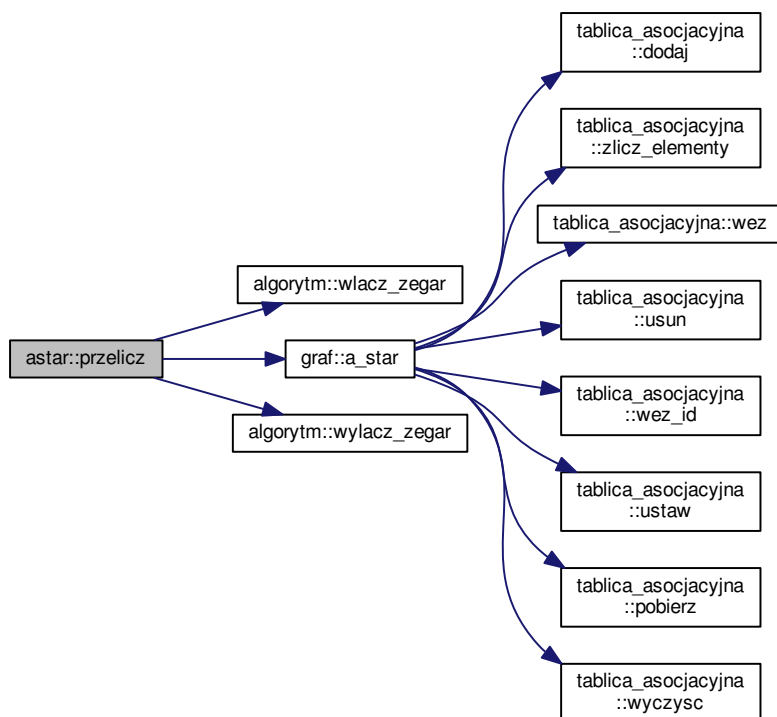
4.2.3.1 `float astar::przelicz ( )` `[virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 439 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:

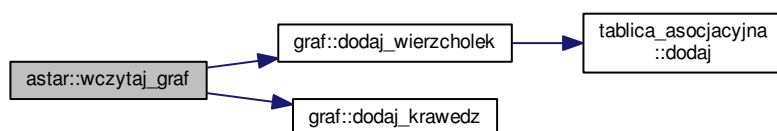


#### 4.2.3.2 void astar::wczytaj\_graf ( )

na podstawie danych z pliku, tworzone sa grafy

Definicja w linii 371 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



## 4.2.4 Dokumentacja atrybutów składowych

### 4.2.4.1 graf astar::G1 [private]

Definicja w linii 304 pliku algorytm.hh.

#### 4.2.4.2 `graf astar::G2` `[private]`

Definicja w linii 304 pliku `algorytm.hh`.

#### 4.2.4.3 `graf astar::G3` `[private]`

Definicja w linii 304 pliku `algorytm.hh`.

#### 4.2.4.4 `graf astar::G4` `[private]`

Definicja w linii 304 pliku `algorytm.hh`.

#### 4.2.4.5 `graf astar::G5` `[private]`

Definicja w linii 304 pliku `algorytm.hh`.

#### 4.2.4.6 `graf astar::G6` `[private]`

Definicja w linii 304 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.3 Dokumentacja klasy bst

Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla bst

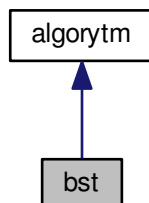
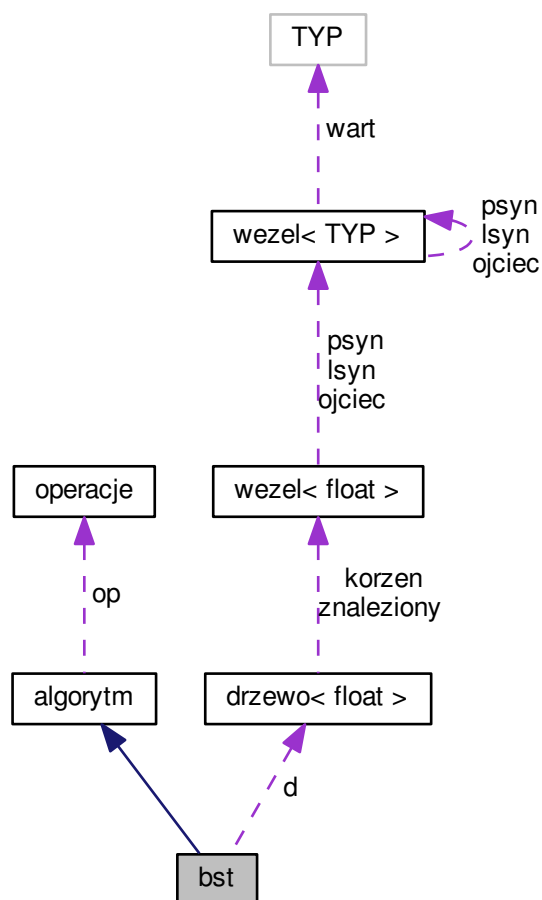


Diagram współpracy dla bst:



## 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 wejściowych zgodnie z zadany algorytmem.*

## Atrybuty prywatne

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

## Dodatkowe Dziedziczone Składowe

### 4.3.1 Opis szczegółowy

Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.

Definicja w linii 227 pliku algorytm.hh.

### 4.3.2 Dokumentacja konstruktora i destruktor

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

Definicja w linii 232 pliku algorytm.hh.

4.3.2.2 `bst::~~bst ( )` `[inline]`

Definicja w linii 240 pliku algorytm.hh.

### 4.3.3 Dokumentacja funkcji składowych

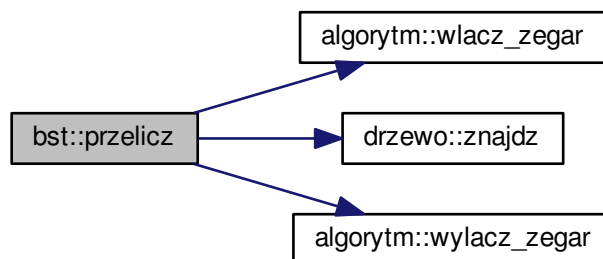
4.3.3.1 `float bst::przelicz ( )` `[virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadaniem algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 204 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.3.3.2 `void bst::wczytaj_klucze ( ifstream & plik )`

Definicja w linii 199 pliku algorytm.cpp.

### 4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 `drzewo<float> bst::d` `[private]`

Definicja w linii 228 pliku algorytm.hh.

#### 4.3.4.2 `string* bst::klucze` `[private]`

Definicja w linii 229 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.4 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 `wyczyszc()`  
*czysci cale drzewo*

### Atrybuty publiczne

- `wezel< TYP > * korzen`  
*korzen drzewa*
- `wezel< TYP > * znaleziony`  
*znaleziony wezel w drzewie*

#### 4.4.1 Opis szczegółowy

```
template<typename TYP>class drzewo< TYP >
```

modeluje binarne drzewo przeszukiwan

Definicja w linii 70 pliku `drzewo.hh`.

### 4.4.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 77 pliku drzewo.hh.

#### 4.4.2.2 `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.

### 4.4.3 Dokumentacja funkcji składowych

#### 4.4.3.1 `template<typename TYP> void drzewo< TYP >::czyszc ( wezel< TYP > * w ) [inline]`

rekursywne czyszczenie wezla

Parametry

in	w	- czyszczony wezel
----	---	--------------------

Definicja w linii 180 pliku drzewo.hh.

#### 4.4.3.2 `template<typename TYP> void drzewo< TYP >::dodaj ( string k, TYP v ) [inline]`

dodaje wezel do drzewa

Parametry

in	k	- klucz wezla
in	v	= wartosc wezla

Definicja w linii 91 pliku drzewo.hh.

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

dodaje wezel do drzewa

Parametry

in	W	- utworzony uprzednio wezel
----	---	-----------------------------

Definicja w linii 83 pliku drzewo.hh.

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

sprawdza, czy w danym wezle znajduje sie szukany klucz

Parametry

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

**Zwraca**

true, gdy znaleziono, false w przeciwnym przypadku

Definicja w linii 109 pliku drzewo.hh.

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

usuwa element o kluczu k, jezeli zostanie on znaleizony

**Parametry**

in	k	- klucz wezla, który należy usunac
----	---	------------------------------------

Definicja w linii 124 pliku drzewo.hh.

**4.4.3.6** `template<typename TYP> void drzewo< TYP >::wyczysc ( ) [inline]`

czysci cale drzewo

Definicja w linii 188 pliku drzewo.hh.

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

szuka wezla o zadanym kluczu

**Parametry**

in	k	- klucz
----	---	---------

**Zwraca**

true, gdy znaleziono, w przeciwnym wypadku zwraca false

Definicja w linii 100 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:

**4.4.4 Dokumentacja atrybutów składowych**

**4.4.4.1** `template<typename TYP> wezel<TYP>* drzewo< TYP >::korzen`

korzen drzewa

Definicja w linii 73 pliku drzewo.hh.



4.4.4.2 `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:

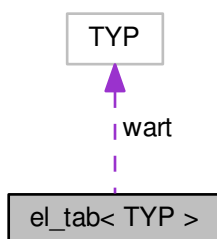
- [drzewo.hh](#)

4.5 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*

## 4.5.1 Opis szczegółowy

```
template<typename TYP>class el_tab< TYP >
```

pojedynczy element tablicy haszującej

Definicja w linii 11 pliku `hashtab.hh`.

## 4.5.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 26 pliku hashtable.hh.

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

Definicja w linii 27 pliku hashtable.hh.

## 4.5.3 Dokumentacja atrybutów składowych

4.5.3.1 `template<typename TYP> string el_tab< TYP>::klucz`

identyfikator

Definicja w linii 16 pliku hashtable.hh.

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

wartosc pola

Definicja w linii 21 pliku hashtable.hh.

4.5.3.3 `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:

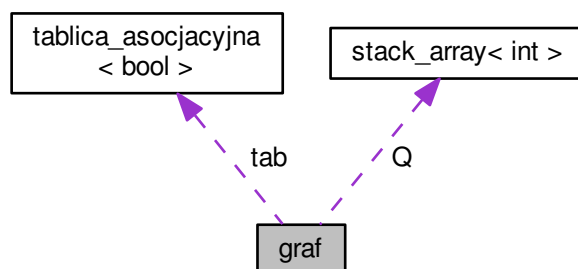
- [hashtab.hh](#)

## 4.6 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 wierzcholki sa polaczone krawedzia - odwołanie poprzez identyfikatory.*
- bool **czy\_sasiad** (wierzcholek w1, wierzcholek w2)  
*Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwołanie poprzez obiekt klasy wierzcholek.*
- void **sasiedztwo** (wierzcholek w)  
*wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzchołkiem - odwołanie poprzez obiekt typu wierzcholek*
- void **dodaj\_wierzcholek** (int id)  
*Dodaje wierzcholek do grafu, metoda dedykowana do algorytmu A\* - algorytm ustawia wezly grafu na siatce, nadajac im jednoczesnie wspolrzedne kartezjanskie.*
- void **dodaj\_wierzcholek** (wierzcholek w)  
*Dodaje wierzcholek do wezla, wierzchołkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.*
- void **dodaj\_krawedz** (int id1, int id2)  
*dodaje krawedz pomiedzy zadane wierzcholki. Waga krawedzi jest wyznaczana w sposob losowy. Najkrotsza odlegloscia miedzy wezlami, jest odleglosc wezlow w sensie metryki typu Manhattan*
- void **dodaj\_krawedz** (unsigned int id1, unsigned int id2, unsigned int waga)  
*Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwołanie poprzez identyfikatory wierzchołkow.*
- void **dodaj\_krawedz** (wierzcholek w1, wierzcholek w2, unsigned int waga)  
*Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwołanie poprzez obiekty typu wierzcholek.*
- void **sasiedztwo** (unsigned int id)  
*wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzchołkiem - odwołanie poprzez identyfikator wierzcholka*
- void **usun\_krawedz** (unsigned int id1, unsigned int id2)  
*usuwa krawedz spomiedzy 2 wierzchołkow - odwołanie poprzez identyfikatory wierzchołkow*
- void **usun\_krawedz** (wierzcholek w1, wierzcholek w2)  
*usuwa krawedz spomiedzy 2 wierzchołkow - odwołanie 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 `wyczyszc` ()  
*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.*
- void `a_star` (int id, int wzor)  
*Metoda przeszukuje wyszukuje zadany element grafu w mysl algorytmu A\*.*
- void `rysuj` ()

## Atrybuty prywatne

- `stack_array`< int > `Q`  
*przechowuje sciezke w algorytmach dfs, bfs oraz best - first*
- `vector`< int > `w_x`  
*wspolrzedna x-owa wierzcholka grafu*
- `vector`< int > `w_y`  
*wspolrzedna y-owa wierzcholka grafu*
- `vector`< int > `poprzednik`  
*wektor, ktory zawiera sciezke w algorytmie A*
- `vector`< int > `dist`  
*dystans wierzcholka od zadanego wezla zrodlowego*
- `vector`< int > `est`  
*estymacja odleglosci do celu w oparciu o metryke typu Manhattan*
- `tablica_asocjacyjna`< bool > `tab`  
*struktura sluzaca do przechowywania grafu, zawiera informacje, czy wierzcholek byl odwiedzony*
- `vector`< `tablica_asocjacyjna`  
< int > > `lista_incydencji`  
*lista incydencji grafu*

### 4.6.1 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.

### 4.6.2 Dokumentacja konstruktora i destruktor

#### 4.6.2.1 `graf::graf ( ) [inline]`

Konstruktor nieparametryczny - ustala sposob zarzadzania pamiecia na stosie.

Definicja w linii 61 pliku graf.hh.

### 4.6.3 Dokumentacja funkcji składowych

#### 4.6.3.1 `void graf::a_star ( int id, int wzor )`

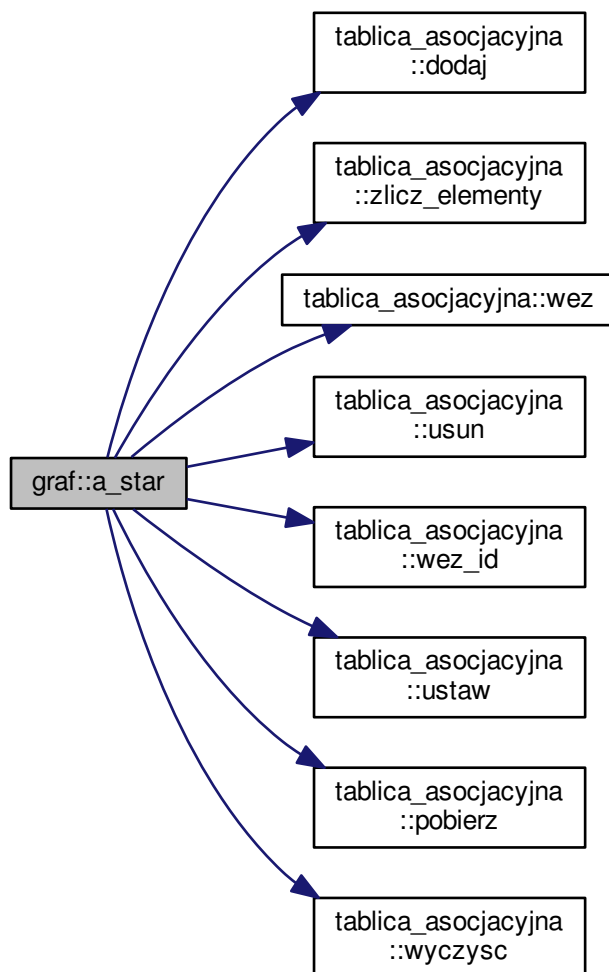
Metoda przeszukuje wyszukuje zadany element grafu w mysl algorytmu A\*.

Parametry

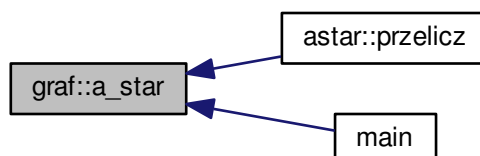
<code>in</code>	<code>id</code>	- id wezla zrodlowego
<code>in</code>	<code>wzor</code>	- wezel docelowy

Definicja w linii 280 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.2 void graf::best\_first ( int *id* )

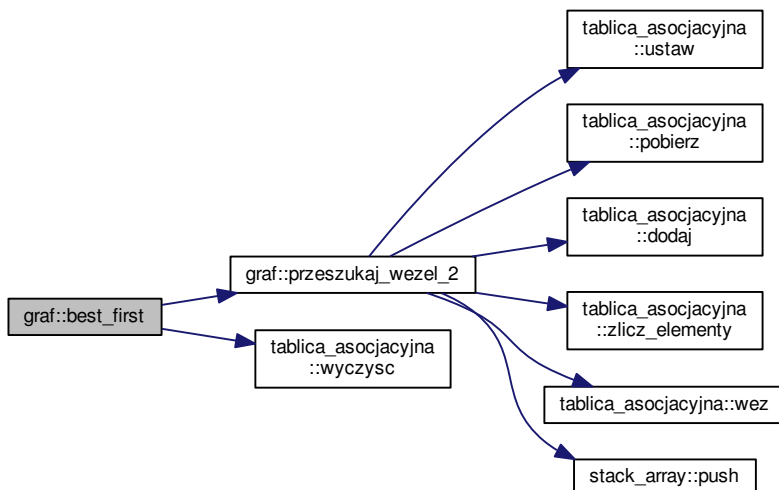
Metoda przeszukuje graf, poczynając od najkrótszej ścieżki.

## Parametry

<code>in</code>	<code>id</code>	- wezel, ktorego szukamy
-----------------	-----------------	--------------------------

Definicja w linii 256 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.3 void graf::bfs ( int id )

Metoda przeszukuje wszcz cały graf.

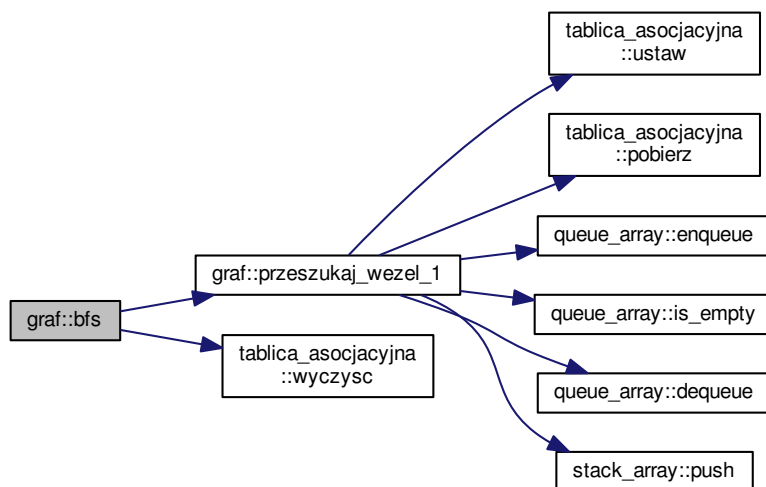
## Parametry

<code>in</code>	<code>id</code>	- wezel, ktorego szukamy
-----------------	-----------------	--------------------------

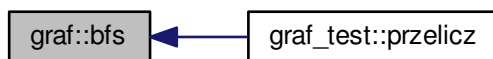
Definicja w linii 205 pliku graf.cpp.



Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.4 bool graf::czy\_sasiad ( unsigned int *id1*, unsigned int *id2* )

Sprawdza czy podane wierzchołki są połączone krawędzią - odwołanie poprzez identyfikatory.

Parametry

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

**Zwraca**

true - gdy sa sasiadami, false - gdy nie sa

Definicja w linii 92 pliku graf.cpp.

Oto graf wywoływań tej funkcji:

**4.6.3.5 bool graf::czy\_sasiad ( wierzcholek w1, wierzcholek w2 )**

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

**Parametry**

in	w1	- pierwszy wierzcholek
in	w2	- drugi wierzcholek

**Zwraca**

true - gdy sa sasiadami, false - gdy nie sa

Definicja w linii 99 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

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

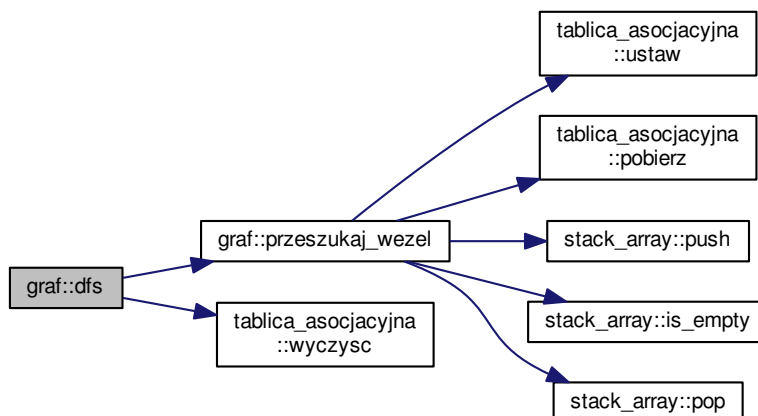
Metoda przeszukuje wglab caly graf.

**Parametry**

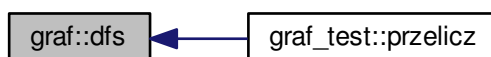
in	id	- wezel, ktorego szukamy
----	----	--------------------------

Definicja w linii 160 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.7 void graf::dodaj\_krawedz ( int id1, int id2 )

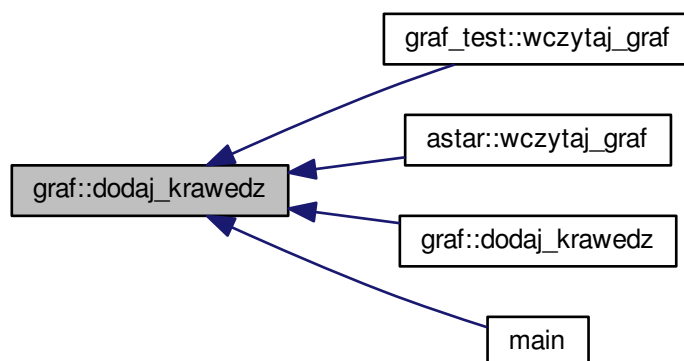
dodaje krawedz pomiędzy zadane wierzchołki. Waga krawedzi jest wyznaczana w sposób losowy. Najkrotsza odlegloscia miedzy wezlami, jest odleglosc wezlow w sensie metryki typu Manhattan

##### Parametry

in	id1	- id pierwszego wezla
in	id2	- id drugiego wezla

Definicja w linii 41 pliku graf.cpp.

Oto graf wywołań tej funkcji:



#### 4.6.3.8 void graf::dodaj\_krawedz ( unsigned int *id1*, unsigned int *id2*, unsigned int *waga* )

Dodaje krawedz o wadze *waga* pomiędzy 2 węzły - odwołanie poprzez identyfikatory wierzchołków.

##### Parametry

in	<i>id1</i>	- id 1. wierzchołka
in	<i>id2</i>	- id 2. wierzchołka
in	<i>waga</i>	- waga krawedzi

Definicja w linii 59 pliku graf.cpp.

#### 4.6.3.9 void graf::dodaj\_krawedz ( wierzcholek *w1*, wierzcholek *w2*, unsigned int *waga* )

Dodaje krawedz o wadze *waga* pomiędzy 2 węzły - odwołanie poprzez obiekty typu wierzcholek.

##### Parametry

in	<i>w1</i>	- pierwszy wierzcholek
in	<i>w2</i>	- drugi wierzcholek
in	<i>waga</i>	- waga krawedzi

Definicja w linii 55 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

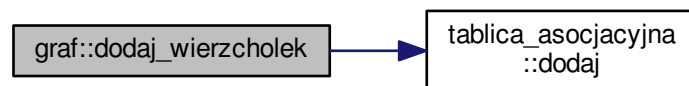


## 4.6.3.10 void graf::dodaj\_wierzcholek ( )

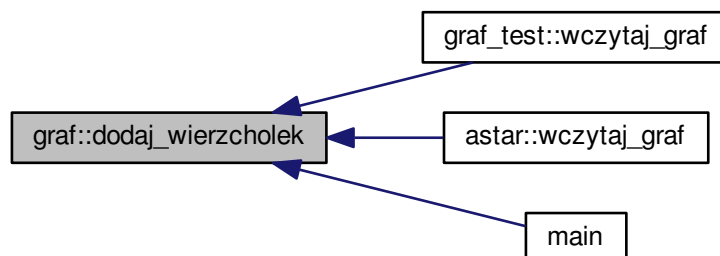
Dodaje wierzcholek do wezla, wierzchołkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.

Definicja w linii 9 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



## 4.6.3.11 void graf::dodaj\_wierzcholek ( int id )

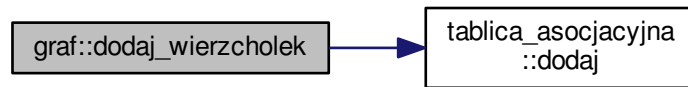
Dodaje wierzcholek do grafu, metoda dedykowana do algorytmu A\* - algorytm ustawia wezly grafu na siatce, nadajac im jednocześnie wspolrzedne kartezjanskie.

Parametry

in	id	- id wierzcholka, wg ktorego wyliczamy jego wspolrzedne
----	----	---

Definicja w linii 27 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

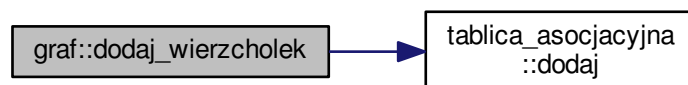


#### 4.6.3.12 void graf::dodaj\_wierzcholek ( wierzcholek w )

Dodaje wierzcholek do węża, wierzchołkom przypisuje się identyfikatory będące kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawędzi incydentnych.

Definicja w linii 17 pliku `graf.cpp`.

Oto graf wywołań dla tej funkcji:



#### 4.6.3.13 bool graf::przeszukaj\_wzegl ( int id, int wzor )

Jeżeli węzeł nie był odwiedzony, odkłada na stos wszystkie jego nieodwiedzone następniki i rekurencyjnie je przeszukuje.

##### Parametry

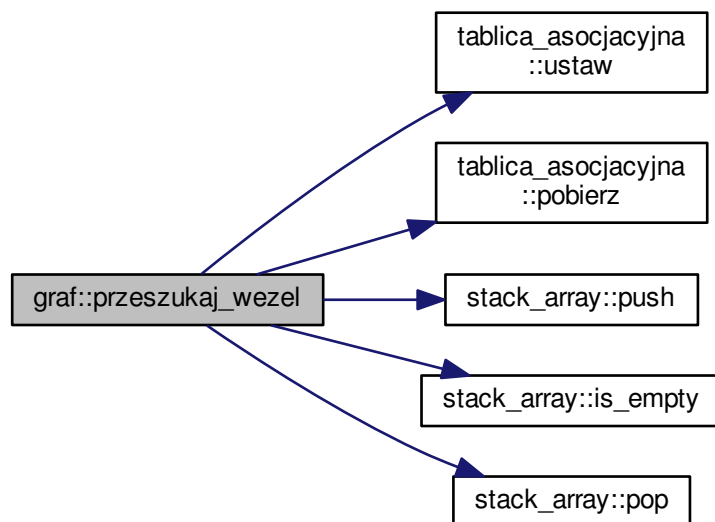
in	id	- id węzła, który ma być przeszukany
in	wzor	- id węzła, którego szukamy

**Zwraca**

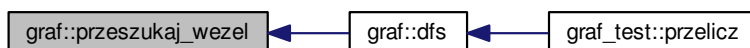
informacja, czy wezel zostal znaleziony

Definicja w linii 126 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.14 bool graf::przeszukaj\_wazel\_1 ( int id, int wzor )

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

**Parametry**

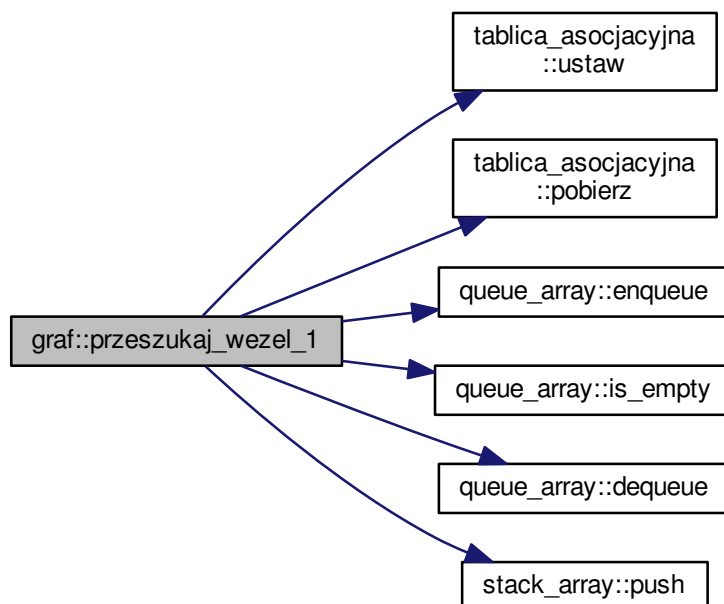
in	id	- id wezła, który ma być przeszukany
in	wzor	- id wezła, którego szukamy

**Zwraca**

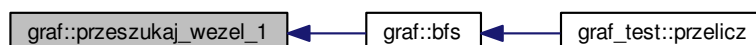
informacja, czy wezel zostal znaleziony

Definicja w linii 171 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.6.3.15 bool graf::przeszukaj\_wazel\_2 ( int id, int wzor )

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

**Parametry**

<code>in</code>	<code>id</code>	- id węzła, który ma być przeszukany
-----------------	-----------------	--------------------------------------



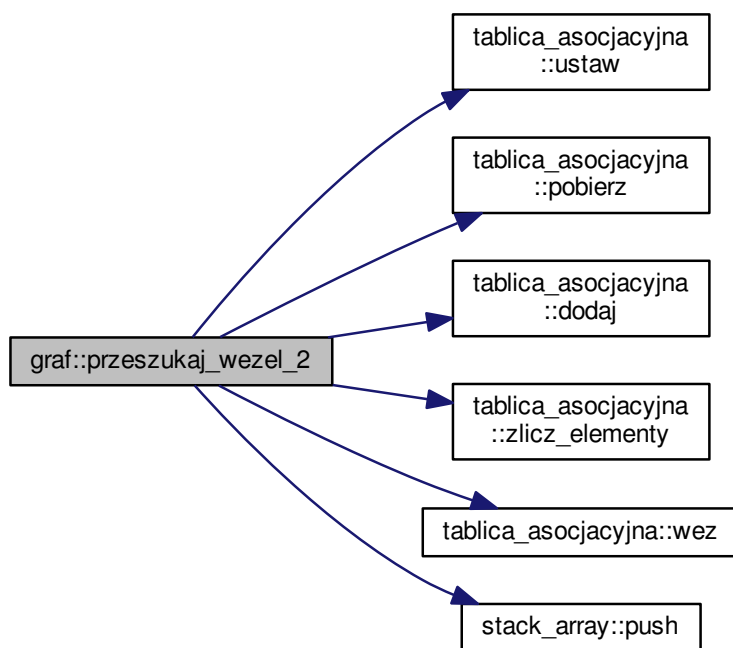
<code>in</code>	<code>wzor</code>	- id wezla, ktorego szukamy
-----------------	-------------------	-----------------------------

**Zwraca**

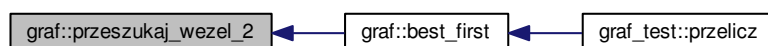
informacja, czy wezel zostal znaleziony

Definicja w linii 218 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

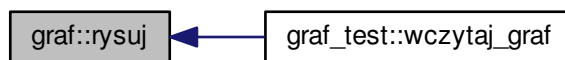


Oto graf wywoływań tej funkcji:

**4.6.3.16 void graf::rysuj ( )**

Definicja w linii 268 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



#### 4.6.3.17 void graf::sasiedztwo ( wierzcholek w )

wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzchołkiem - odwołanie poprzez obiekt typu wierzcholek

##### Parametry

in	w	- zadany wierzcholek
----	---	----------------------

Definicja w linii 76 pliku graf.cpp.

#### 4.6.3.18 void graf::sasiedztwo ( unsigned int id )

wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzchołkiem - odwołanie poprzez identyfikator wierzcholka

##### Parametry

in	id	- id wierzcholka
----	----	------------------

Definicja w linii 67 pliku graf.cpp.

#### 4.6.3.19 void graf::usun\_krawedz ( unsigned int id1, unsigned int id2 )

usuwa krawedz spomiedzy 2 wierzchołkow - odwołanie poprzez identyfikatory wierzchołkow

##### Parametry

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

Definicja w linii 104 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



4.6.3.20 void graf::usun\_krawedz ( wierzcholek w1, wierzcholek w2 )

usuwa krawedz spomiedzy 2 wierzchołkow - odwołanie poprzez obiekt typu wierzcholek

**Parametry**

in	w1	- pierwszy wierzcholek
in	w2	- drugi wierzcholek

Definicja w linii 113 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

**4.6.3.21 void graf::usun\_wierzcholek ( unsigned int id )**

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

**Parametry**

in	id	- id wierzcholka
----	----	------------------

Definicja w linii 116 pliku graf.cpp.

Oto graf wywoływań tej funkcji:

**4.6.3.22 void graf::usun\_wierzcholek ( wierzcholek w )**

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

**Parametry**

in	w	- wierzcholek ktory trzeba usunac
----	---	-----------------------------------

Definicja w linii 122 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.6.3.23 void graf::wyczysc ( ) [inline]

usuwa wszystkie obiekty z listy incydencji grafu

Definicja w linii 136 pliku graf.hh.

#### 4.6.3.24 void graf::wypisz\_liste ( )

wypisuje pełną listę incydencji grafu

Definicja w linii 80 pliku graf.cpp.

### 4.6.4 Dokumentacja atrybutów składowych

#### 4.6.4.1 vector<int> graf::dist [private]

dystans wierzchołka od zadanego węzła źródłowego

Definicja w linii 50 pliku graf.hh.

#### 4.6.4.2 vector<int> graf::est [private]

estymacja odległości do celu w oparciu o metrykę typu Manhattan

Definicja w linii 52 pliku graf.hh.

#### 4.6.4.3 vector<tablica\_asocjacyjna<int> > graf::lista\_incydencji [private]

lista incydencji grafu

Definicja w linii 58 pliku graf.hh.

#### 4.6.4.4 vector<int> graf::poprzednik [private]

wektor, który zawiera ścieżkę w algorytmie A

Definicja w linii 48 pliku graf.hh.

#### 4.6.4.5 stack\_array<int> graf::Q [private]

przechowuje ścieżkę w algorytmach dfs, bfs oraz best - first

Definicja w linii 42 pliku graf.hh.

#### 4.6.4.6 `tablica_asocjacyjna<bool> graf::tab` [private]

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

Definicja w linii 56 pliku graf.hh.

#### 4.6.4.7 `vector<int> graf::w_x` [private]

wspolrzedna x-owa wierzcholka grafu

Definicja w linii 44 pliku graf.hh.

#### 4.6.4.8 `vector<int> graf::w_y` [private]

wspolrzedna y-owa wierzcholka grafu

Definicja w linii 46 pliku graf.hh.

Dokumentacja dla tej klasy zostala wygenerowana z plikow:

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

## 4.7 Dokumentacja klasy graf\_test

modeluje strukture grafow uzytych do badan

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

Diagram dziedziczenia dla graf\_test

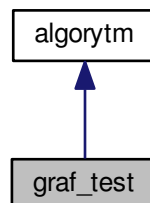
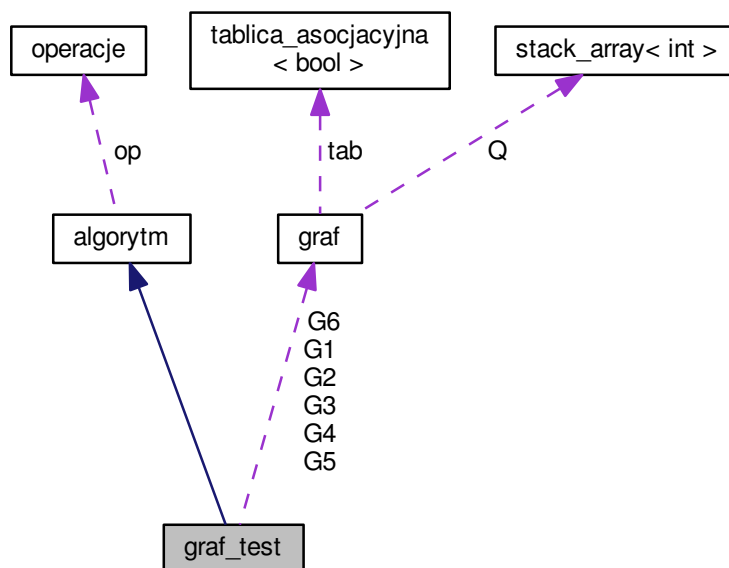


Diagram współpracy dla graf\_test:



### 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 algorytm.*

### 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

### 4.7.1 Opis szczegółowy

modeluje strukture grafów użytych do badań

Definicja w linii 291 pliku algorytm.hh.

### 4.7.2 Dokumentacja konstruktora i destruktor

#### 4.7.2.1 `graf_test::graf_test ( ifstream & plik1, ifstream & plik2, int N, int M, int t ) [inline]`

konstruktor

Definicja w linii 300 pliku algorytm.hh.

### 4.7.3 Dokumentacja funkcji składowych

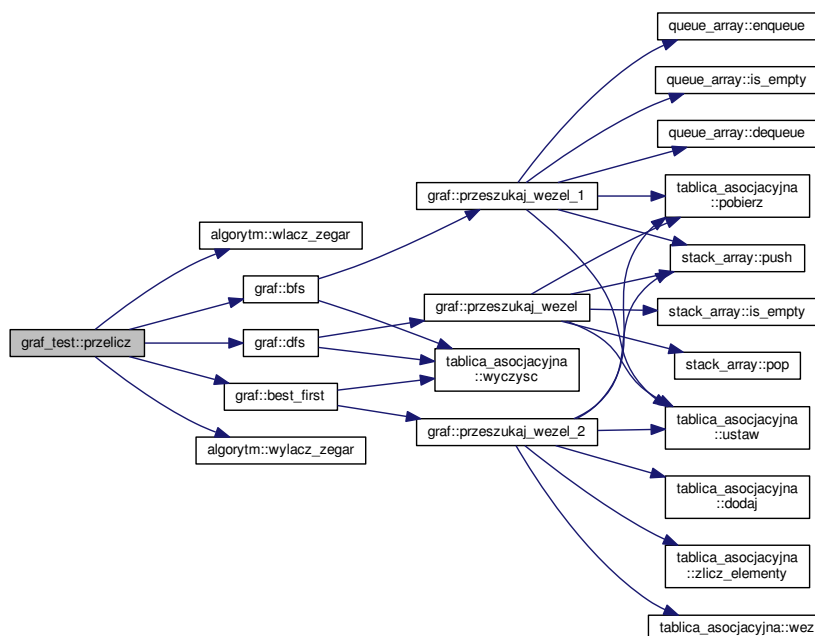
#### 4.7.3.1 `float graf_test::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadaniem algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 329 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



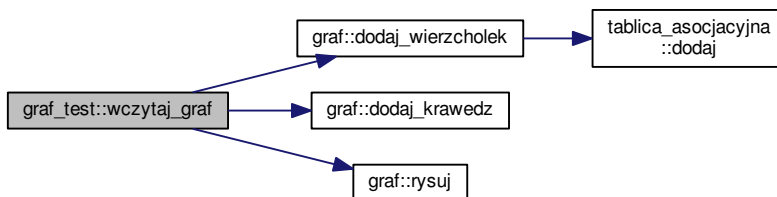
#### 4.7.3.2 `void graf_test::wczytaj_graf ( )`

na podstawie danych z pliku, tworzone są grafy



Definicja w linii 262 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.7.4 Dokumentacja atrybutów składowych

##### 4.7.4.1 `graf graf_test::G1` [private]

\ grafy, na których testuje się algorytm przeszukiwania

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.2 `graf graf_test::G2` [private]

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.3 `graf graf_test::G3` [private]

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.4 `graf graf_test::G4` [private]

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.5 `graf graf_test::G5` [private]

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.6 `graf graf_test::G6` [private]

Definicja w linii 293 pliku algorytm.hh.

##### 4.7.4.7 `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](#)

## 4.8 Dokumentacja klasy h\_sort

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

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

Diagram dziedziczenia dla h\_sort

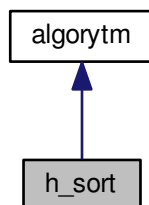
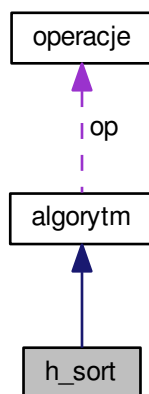


Diagram współpracy dla h\_sort:



### Metody publiczne

- `h_sort` (`ifstream &plik1`, `ifstream &plik2`, `int N`, `int M`)  
*konstruktor klasy*
- `float przelicz ()`  
*metoda dokonujaca sortowania danych*

### Dodatkowe Dziedziczone Składowe

### 4.8.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

Definicja w linii 208 pliku `algorytm.hh`.

### 4.8.2 Dokumentacja konstruktora i destruktor

4.8.2.1 `h_sort::h_sort ( ifstream &plik1, ifstream &plik2, int N, int M ) [inline]`

konstruktor klasy

Definicja w linii 211 pliku `algorytm.hh`.

### 4.8.3 Dokumentacja funkcji składowych

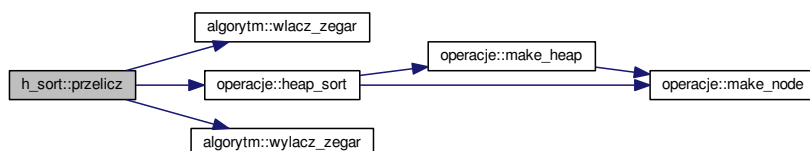
4.8.3.1 `float h_sort::przelicz ( ) [virtual]`

metoda dokonująca 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](#)

## 4.9 Dokumentacja klasy `h_table`

Modeluje tablice haszująca przeznaczona do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla h\_table

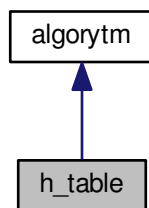
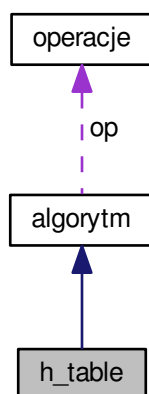


Diagram współpracy dla h\_table:



## 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 wejsciowych zgodnie z zadany algorytmem.*

## Atrybuty prywatne

- string \* `klucze`  
*tablica kluczy*

## Dodatkowe Dziedziczone Składowe

### 4.9.1 Opis szczegółowy

Modeluje tablice haszująca przeznaczona do testowania szybkości wyszukiwania.

Definicja w linii 247 pliku `algorytm.hh`.

### 4.9.2 Dokumentacja konstruktora i destruktor

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

konstruktor

Definicja w linii 256 pliku `algorytm.hh`.

### 4.9.3 Dokumentacja funkcji składowych

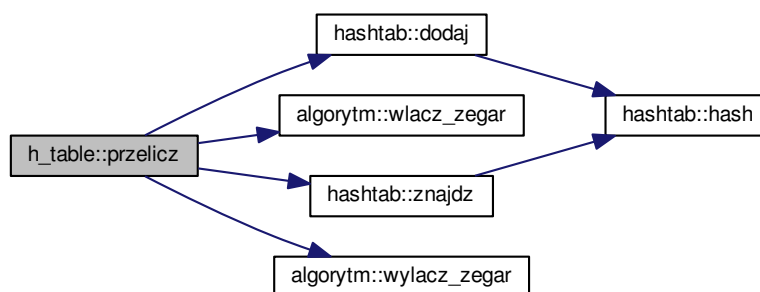
4.9.3.1 `float h_table::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 219 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:



4.9.3.2 `void h_table::wczytaj_klucze ( ifstream & plik )`

wczytywanie kluczy

Parametry

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

Definicja w linii 213 pliku `algorytm.cpp`.

### 4.9.4 Dokumentacja atrybutów składowych

#### 4.9.4.1 `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](#)

## 4.10 Dokumentacja szablonu klasy `hashtab< TYP >`

modeluje tablice haszujca w oparciu o kontener klasy [el\\_tab](#)

```
#include <hashtab.hh>
```

### Metody publiczne

- void [ustaw\\_dlugosc](#) (int d)  
*ustawia dlugosc tablicy*
- [hashtab](#) ()  
*konstruktor bezparametryczny*
- [hashtab](#) (int N)  
*konsruktor parametryczny*
- unsigned long [hash](#) (string k)  
*funkcja haszujaca*
- void [dodaj](#) (string k, TYP v)  
*metoda dodaje element do tablicy 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*

#### 4.10.1 Opis szczegółowy

```
template<typename TYP>class hashtab< TYP >
```

modeluje tablice haszujca w oparciu o kontener klasy [el\\_tab](#)

Definicja w linii 34 pliku `hashtab.hh`.

### 4.10.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 43 pliku hashtable.hh.

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

konstruktor parametryczny

Parametry

<code>in</code>	<code>N</code>	- rozmiar tablicy
-----------------	----------------	-------------------

Definicja w linii 47 pliku hashtable.hh.

Oto graf wywołań dla tej funkcji:



### 4.10.3 Dokumentacja funkcji składowych

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

metoda dodaje element do tablicy haszującej

Definicja w linii 59 pliku hashtable.hh.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań tej funkcji:



**4.10.3.2** `template<typename TYP> unsigned long hashtable< TYP >::hash ( string k ) [inline]`

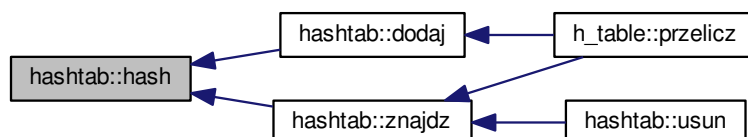
funkcja haszująca

**Zwraca**

`h` - liczba, która po kompresji będzie indeksem danego elementu

Definicja w linii 51 pliku `hashtab.hh`.

Oto graf wywołań tej funkcji:



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

ustawia dlugosc tablicy

Definicja w linii 41 pliku `hashtab.hh`.

Oto graf wywołań tej funkcji:





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

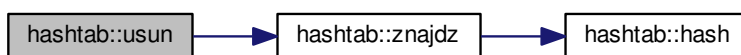
usuwa element jesli znajduje sie w tablicy

## Parametry

<i>in</i>	<i>k</i>	- klucz
-----------	----------	---------

Definicja w linii 89 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



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

Definicja w linii 93 pliku hashtab.hh.

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

metoda szuka zadanego elementu w oparciu o klucz

## Parametry

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

## Zwraca

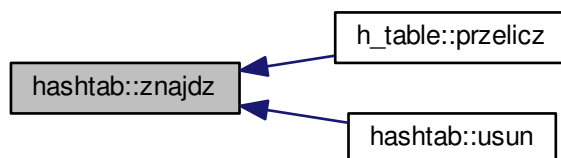
znaleziony element

Definicja w linii 74 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.10.4 Dokumentacja atrybutów składowych

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

dlugosc tablicy

Definicja w linii 36 pliku `hashtab.hh`.

4.10.4.2 `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](#)

### 4.11 Dokumentacja klasy kolejka\_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla `kolejka_lista`

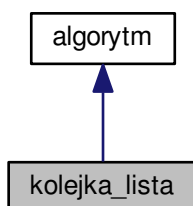
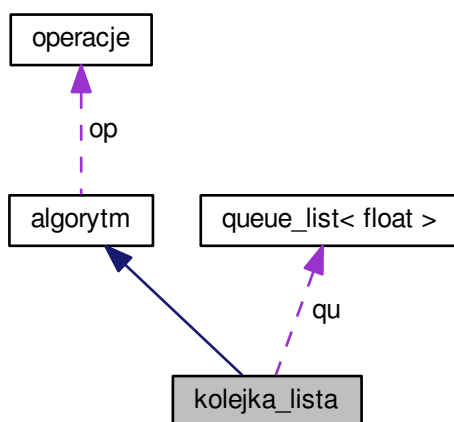


Diagram współpracy dla kolejka\_lista:



## 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

### 4.11.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 191 pliku algorytm.hh.

### 4.11.2 Dokumentacja konstruktora i destruktor

4.11.2.1 `kolejka_lista::kolejka_lista ( ifstream &plik1, ifstream &plik2, int N, int M ) [inline]`

Definicja w linii 194 pliku algorytm.hh.

### 4.11.3 Dokumentacja funkcji składowych

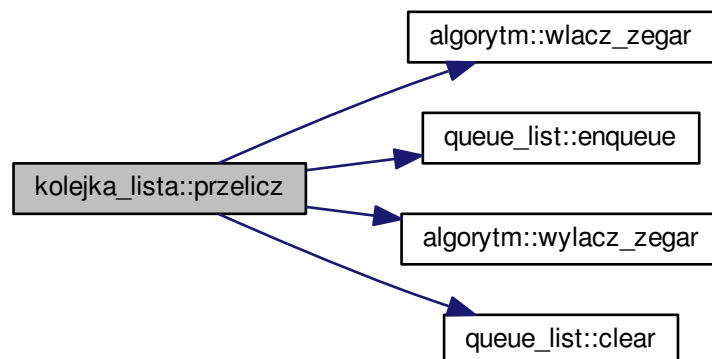
4.11.3.1 `float kolejka_lista::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 160 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.11.4 Dokumentacja atrybutów składowych

##### 4.11.4.1 `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](#)

## 4.12 Dokumentacja klasy kolejka\_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka\_tablica

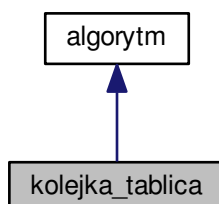
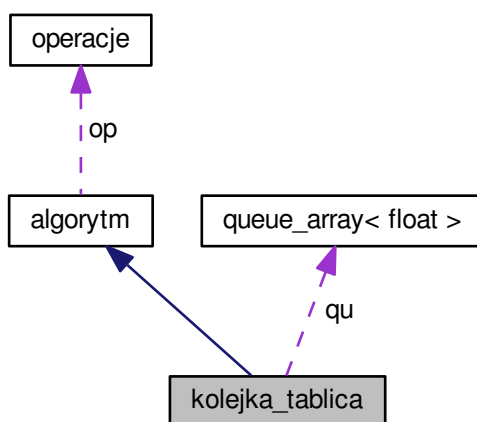


Diagram współpracy dla kolejka\_tablica:



## 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

### 4.12.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 179 pliku `algorytm.hh`.

### 4.12.2 Dokumentacja konstruktora i destruktor

4.12.2.1 `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`.

### 4.12.3 Dokumentacja funkcji składowych

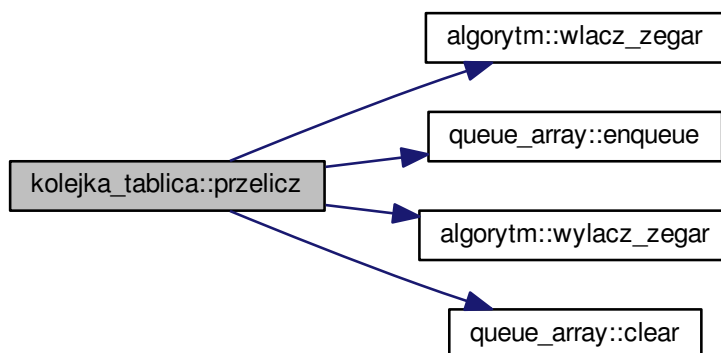
#### 4.12.3.1 float kolejka\_tablica::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 148 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



### 4.12.4 Dokumentacja atrybutów składowych

#### 4.12.4.1 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](#)

## 4.13 Dokumentacja klasy m\_sort

klasa reprezentuje dane poddane sortowaniu przez scalanie

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

Diagram dziedziczenia dla m\_sort

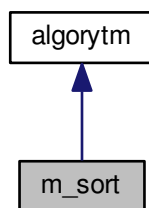
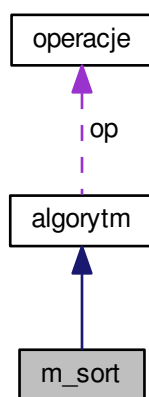


Diagram współpracy dla m\_sort:



## Metody publiczne

- `m_sort` (ifstream &plik1, ifstream &plik2, int N, int M)  
*konstruktor*
- float `przelicz` ()  
*metoda dokonujaca sortowania danych*

## Dodatkowe Dziedziczone Składowe

### 4.13.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez scalanie

Definicja w linii 217 pliku algorytm.hh.



### 4.13.2 Dokumentacja konstruktora i destruktor

4.13.2.1 `m_sort::m_sort ( ifstream & plik1, ifstream & plik2, int N, int M ) [inline]`

konstruktor

Definicja w linii 220 pliku algorytm.hh.

### 4.13.3 Dokumentacja funkcji składowych

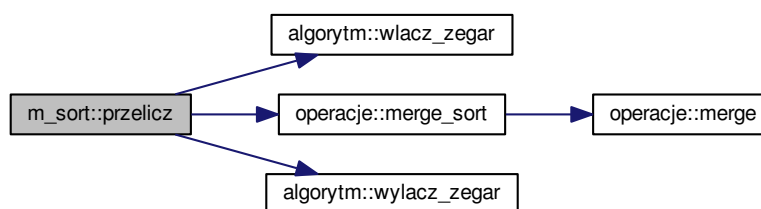
4.13.3.1 `float m_sort::przelicz ( ) [virtual]`

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](#)

## 4.14 Dokumentacja klasy mnozenie

modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2

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

Diagram dziedziczenia dla mnozenie

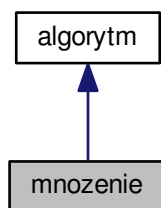
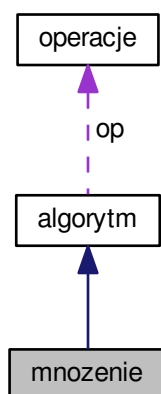


Diagram współpracy dla mnozenie:



## Metody publiczne

- `mnozenie` (ifstream &plik1, ifstream &plik2, int N, int M)
- float `przelicz` ()  
*wykonuje zalozony algorytm mnozenia elementow tablicy przez 2*

## Dodatkowe Dziedziczone Składowe

### 4.14.1 Opis szczegółowy

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2

Definicja w linii 139 pliku algorytm.hh.

### 4.14.2 Dokumentacja konstruktora i destruktor

4.14.2.1 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.

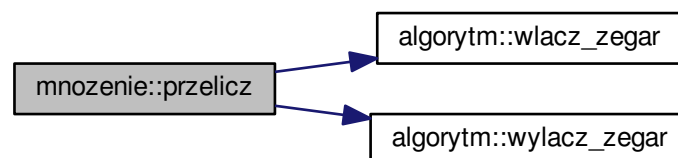
**4.14.3 Dokumentacja funkcji składowych****4.14.3.1 float mnozenie::przelicz ( ) [virtual]**

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](#)

**4.15 Dokumentacja klasy operacje**

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

```
#include <operacje.hh>
```

**Metody publiczne**

- [operacje](#) ()  
*konstruktor bezparametryczny*
- [operacje](#) (int N)  
*konstruktor parametryczny - alokuje pamiec w dynamicznej tablicy tab*
- bool [zamien\\_elementy](#) (int i, int j)  
*Metoda zamienia 2 elementy tablicy.*
- void [quick\\_sort](#) (int l, int p)  
*Metoda Dokonuje sortownaia szybkiego.*

- void `make_node` (int rozmiar, int i)  
*Metoda tworzy wezel drzewa, przypisujac mu 2 synow, ustawiajac ich w odpowiedniej kolejnosci (ojciec ma najwieksza wartosc)*
- void `make_heap` ()  
*Metoda tworzy kopiec binarny.*
- void `heap_sort` ()  
*Metoda dokonuje sortowania po uprzednim utworzeniu kopca.*
- void `merge` (int poczatek, int srodek, int koniec)  
*Metoda scala dwie czesci tablicy, jednoczesnie je porzadkujac.*
- void `merge_sort` (int poczatek, int koniec)
- void `odwroc_tablice` ()  
*metoda odwraca wszystkie elementy tablicy*
- void `dodaj_element` (float e)  
*metoda dodaje element do tablicy, alokujac dodatkowa pamiec*
- void `dodaj_elementy` (float \*tab2, int rozm)  
*metoda dodaje elementy do tablicy*
- void `operator=` (float \*tab1)  
*Przeciazenie operatora przypisania; przypisuje elementy tablicy `tab1` do tablicy bedacej polem klasy.*
- bool `operator==` (float \*tab1)  
*Przeciazenie operatora porownania; metoda porownuje zawartosci dwoch tablic.*
- float & `operator[]` (int ind)

### Atrybuty publiczne

- int `n`  
*ilosc elementow w tablicy*
- float \* `tab`  
*tablica z liczbami*

### 4.15.1 Opis szczegółowy

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

Definicja w linii 11 pliku operacje.hh.

### 4.15.2 Dokumentacja konstruktora i destruktora

#### 4.15.2.1 `operacje::operacje ( )`

konstruktor bezparametryczny

#### 4.15.2.2 `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.

### 4.15.3 Dokumentacja funkcji składowych

#### 4.15.3.1 void operacje::dodaj\_element ( float e )

metoda dodaje element do tablicy, alokując dodatkową pamięć

## Parametry

in	e	- element, który należy dołączyć do tablicy
----	---	---

Definicja w linii 27 pliku operacje.cpp.

## 4.15.3.2 void operacje::dodaj\_elementy ( float \* tab2, int rozm )

metoda dodaje elementy do tablicy

## Parametry

in	tab2	- tablica, która należy dołączyć
in	rozm	- rozmiar tablicy tab2

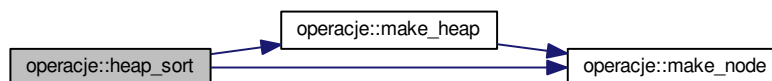
Definicja w linii 46 pliku operacje.cpp.

## 4.15.3.3 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:



## 4.15.3.4 void operacje::make\_heap ( )

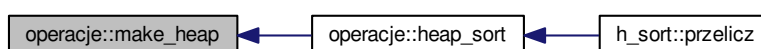
Metoda tworzy kopiec binarny.

Definicja w linii 110 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.15.3.5 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:



#### 4.15.3.6 void operacje::merge ( int poczatek, int srodek, int koniec )

Metoda scala dwie czesci tablicy, jednocześnie je porzadkujac.

##### Parametry

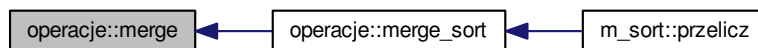
in	<i>poczatek</i>	- pierwszy indeks tablicy
in	<i>srodek</i>	- srodkowy indeks tablicy



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

Definicja w linii 130 pliku operacje.cpp.

Oto graf wywołań tej funkcji:



#### 4.15.3.7 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:



#### 4.15.3.8 void operacje::odwroc\_tablice ( )

metoda odwraca wszystkie elementy tablicy

Definicja w linii 12 pliku operacje.cpp.

#### 4.15.3.9 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.

4.15.3.10 `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.

4.15.3.11 `float& operacje::operator[] ( int ind ) [inline]`

Definicja w linii 88 pliku operacje.hh.

4.15.3.12 `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:



#### 4.15.3.13 bool operacje::zamien\_elementy ( int *i*, int *j* )

Metoda zamienia 2 elementy tablicy.

Parametry

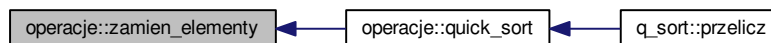
in	<i>i</i>	- element tablicy
in	<i>j</i>	- element tablicy

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:



### 4.15.4 Dokumentacja atrybutów składowych

#### 4.15.4.1 int operacje::n

ilosc elementow w tablicy

Definicja w linii 16 pliku operacje.hh.

#### 4.15.4.2 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](#)

## 4.16 Dokumentacja klasy q\_sort

klasa reprezentuje dane poddane sortowaniu szybkemu

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

Diagram dziedziczenia dla q\_sort

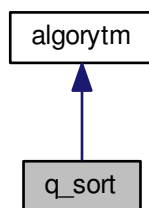
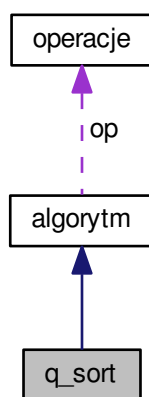


Diagram współpracy dla q\_sort:



### Metody publiczne

- `q_sort` (ifstream &plik1, ifstream &plik2, int N, int M)  
*konstruktor klasy*
- float `przelicz` ()  
*metoda dokonujaca sortowania danych*

### Dodatkowe Dziedziczone Składowe

#### 4.16.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkemu

Definicja w linii 200 pliku algorytm.hh.

## 4.16.2 Dokumentacja konstruktora i destruktor

4.16.2.1 `q_sort::q_sort ( ifstream & plik1, ifstream & plik2, int N, int M )` `[inline]`

konstruktor klasy

Definicja w linii 203 pliku `algorytm.hh`.

## 4.16.3 Dokumentacja funkcji składowych

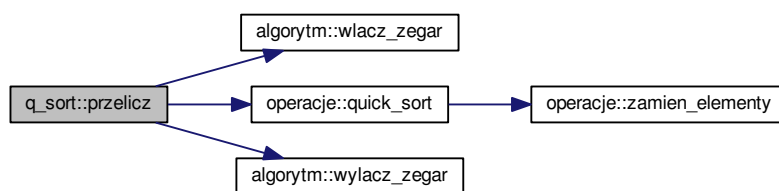
4.16.3.1 `float q_sort::przelicz ( )` `[virtual]`

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:

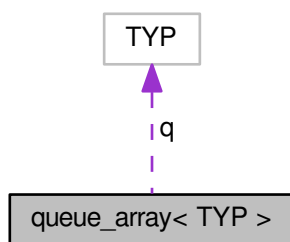
- [algorytm.hh](#)
- [algorytm.cpp](#)

## 4.17 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`

#### 4.17.1 Opis szczegółowy

```
template<typename TYP>class queue_array< TYP >
```

Modeluje kolejke w oparciu o tablice.

Definicja w linii 50 pliku kolejka.hh.

## 4.17.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 63 pliku kolejka.hh.

4.17.2.2 `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.

## 4.17.3 Dokumentacja funkcji składowych

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

czysci kolejke

Definicja w linii 173 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:

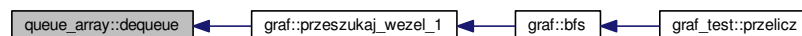


4.17.3.2 `template<typename TYP> TYP queue_array< TYP >::dequeue ( ) [inline]`

usuwa element z konca kolejki

Definicja w linii 129 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



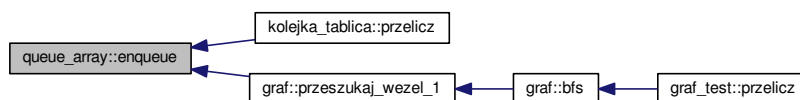
4.17.3.3 `template<typename TYP> void queue_array< TYP >::enqueue ( TYP element ) [inline]`

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

Definicja w linii 82 pliku kolejka.hh.



Oto graf wywoływań tej funkcji:



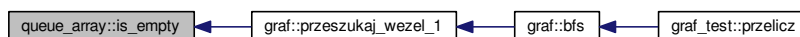
#### 4.17.3.4 `template<typename TYP> bool queue_array< TYP >::is_empty ( ) [inline]`

Zwraca

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

Definicja w linii 75 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



#### 4.17.3.5 `template<typename TYP> int queue_array< TYP >::size ( ) [inline]`

Zwraca

rozmiar kolejki

Definicja w linii 70 pliku kolejka.hh.

### 4.17.4 Dokumentacja atrybutów składowych

#### 4.17.4.1 `template<typename TYP> flag queue_array< TYP >::f`

flaga trybu zwiększania pamięci , przyjmuje wartosc : plus1 - dla trybu kazdorazowego powiększania pamięci x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 59 pliku kolejka.hh.

#### 4.17.4.2 `template<typename TYP> TYP* queue_array< TYP >::q [private]`

Definicja w linii 51 pliku kolejka.hh.

#### 4.17.4.3 `template<typename TYP> int queue_array< TYP >::s [private]`

Definicja w linii 52 pliku kolejka.hh.

#### 4.17.4.4 `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](#)

## 4.18 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`

### 4.18.1 Opis szczegółowy

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

Modeluje kolejkę opartą na liście STL.

Definicja w linii 19 pliku kolejka.hh.

### 4.18.2 Dokumentacja funkcji składowych

#### 4.18.2.1 `template<typename TYP> void queue_list< TYP >::clear ( )` `[inline]`

*czyszczy stos*

Definicja w linii 41 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



**4.18.2.2** `template<typename TYP> TYP queue_list< TYP >::dequeue ( ) [inline]`

usuwa element

Definicja w linii 35 pliku `kolejka.hh`.

**4.18.2.3** `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:



**4.18.2.4** `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`.

**4.18.2.5** `template<typename TYP> int queue_list< TYP >::size ( ) [inline]`

Zwraca

rozmiar kolejki

Definicja w linii 31 pliku `kolejka.hh`.

### 4.18.3 Dokumentacja atrybutów składowych

4.18.3.1 `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:

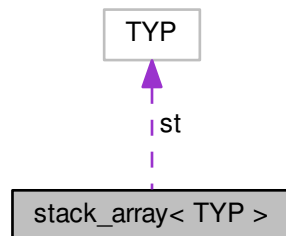
- [kolejka.hh](#)

## 4.19 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`

### 4.19.1 Opis szczegółowy

```
template<typename TYP>class stack_array< TYP >
```

Modeluje stos w oparciu o tablice.

Definicja w linii 59 pliku `stos.hh`.

### 4.19.2 Dokumentacja konstruktora i destruktor

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

konstruktor bezparametryczny

Definicja w linii 72 pliku `stos.hh`.

4.19.2.2 `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`.

### 4.19.3 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 183 pliku `stos.hh`.

Oto graf wywoływań tej funkcji:



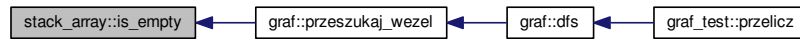
4.19.3.2 `template<typename TYP> bool stack_array< TYP >::is_empty ( ) [inline]`

**Zwraca**

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

Definicja w linii 79 pliku stos.hh.

Oto graf wywołań tej funkcji:

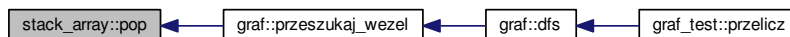


#### 4.19.3.3 `template<typename TYP> TYP stack_array< TYP >::pop ( ) [inline]`

zdejmuje element ze stosu

Definicja w linii 140 pliku stos.hh.

Oto graf wywołań tej funkcji:

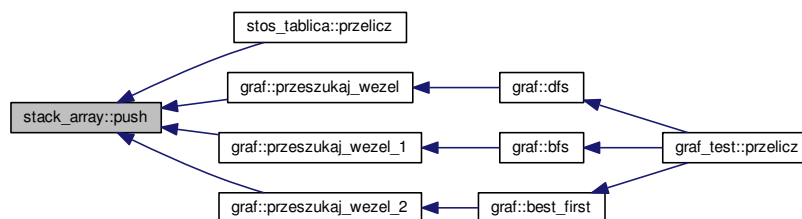


#### 4.19.3.4 `template<typename TYP> void stack_array< TYP >::push ( TYP element ) [inline]`

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

Definicja w linii 91 pliku stos.hh.

Oto graf wywołań tej funkcji:



#### 4.19.3.5 `template<typename TYP> int stack_array< TYP >::size ( ) [inline]`

**Zwraca**

rozmiar ztosu

Definicja w linii 87 pliku stos.hh.

#### 4.19.4 Dokumentacja atrybutów składowych

##### 4.19.4.1 `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`.

##### 4.19.4.2 `template<typename TYP> int stack_array< TYP >::s [private]`

Definicja w linii 61 pliku `stos.hh`.

##### 4.19.4.3 `template<typename TYP> int stack_array< TYP >::sp [private]`

Definicja w linii 61 pliku `stos.hh`.

##### 4.19.4.4 `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](#)

## 4.20 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`

### 4.20.1 Opis szczegółowy

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

Modeluje stos oparty na liście STL.

Definicja w linii 22 pliku stos.hh.

### 4.20.2 Dokumentacja funkcji składowych

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

czyszczy stos

Definicja w linii 50 pliku stos.hh.

Oto graf wywoływań tej funkcji:



4.20.2.2 `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.

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

zdejmuje element z wierzchu stosu

Definicja w linii 42 pliku stos.hh.

4.20.2.4 `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:



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

Zwraca

rozmiar ztosu

Definicja w linii 34 pliku stos.hh.

### 4.20.3 Dokumentacja atrybutów składowych

4.20.3.1 `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](#)

## 4.21 Dokumentacja klasy stos\_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla stos\_lista

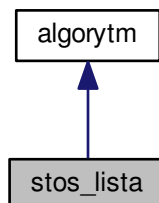
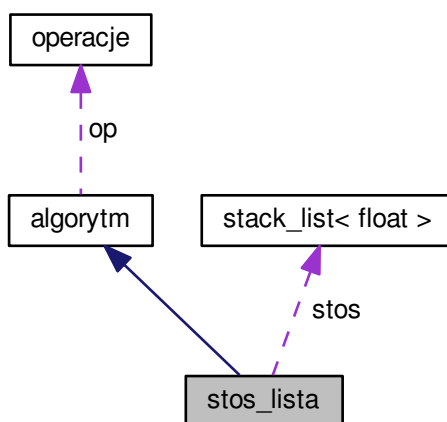


Diagram współpracy dla `stos_lista`:



## 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

### 4.21.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 170 pliku `algorytm.hh`.

### 4.21.2 Dokumentacja konstruktora i destruktor

4.21.2.1 `stos_lista::stos_lista ( ifstream &plik1, ifstream &plik2, int N, int M ) [inline]`

Definicja w linii 173 pliku `algorytm.hh`.

### 4.21.3 Dokumentacja funkcji składowych

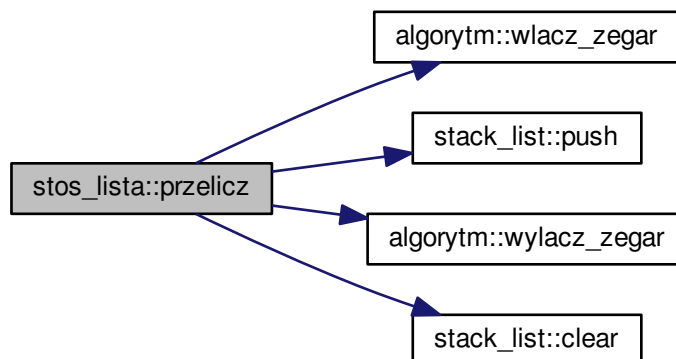
4.21.3.1 `float stos_lista::przelicz ( ) [virtual]`

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:



#### 4.21.4 Dokumentacja atrybutów składowych

##### 4.21.4.1 `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](#)

## 4.22 Dokumentacja klasy `stos_tablica`

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla `stos_tablica`

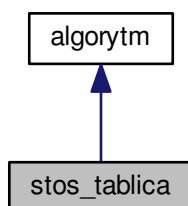
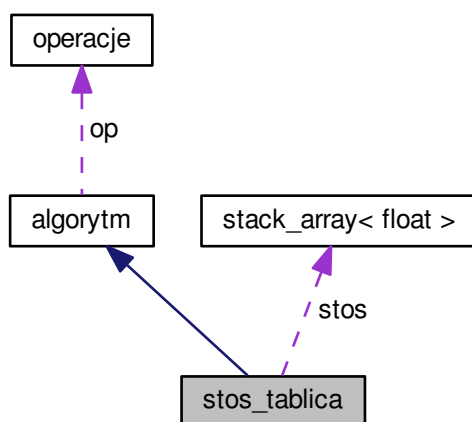


Diagram współpracy dla stos\_tablica:



## 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

### 4.22.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 158 pliku `algorytm.hh`.

### 4.22.2 Dokumentacja konstruktora i destruktor

#### 4.22.2.1 `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`.

### 4.22.3 Dokumentacja funkcji składowych

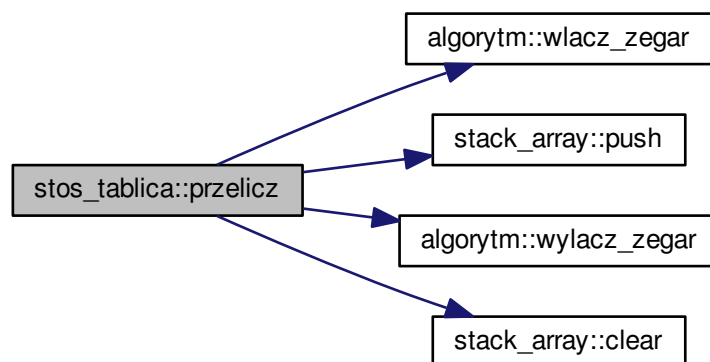
#### 4.22.3.1 float stos\_tablica::przelicz ( ) [virtual]

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:



## 4.22.4 Dokumentacja atrybutów składowych

#### 4.22.4.1 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](#)

## 4.23 Dokumentacja klasy tab\_aso

Modeluje tablice asocjacyjną przeznaczoną do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla tab\_aso

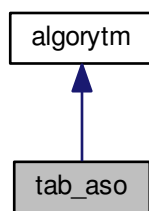
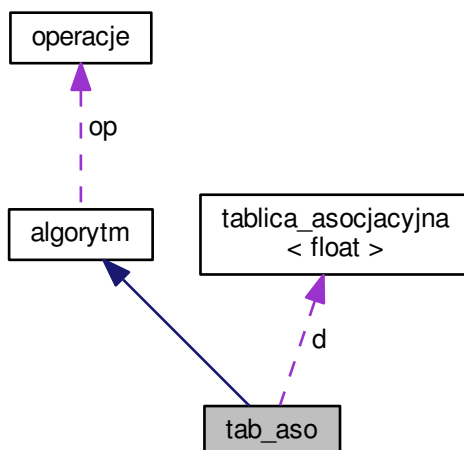


Diagram współpracy dla tab\_aso:



### 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

### 4.23.1 Opis szczegółowy

Modeluje tablice asocjacyjną przeznaczoną do testowania szybkości wyszukiwania.

Definicja w linii 265 pliku `algorytm.hh`.

### 4.23.2 Dokumentacja konstruktora i destruktor

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

konstruktor

Definicja w linii 278 pliku `algorytm.hh`.

### 4.23.3 Dokumentacja funkcji składowych

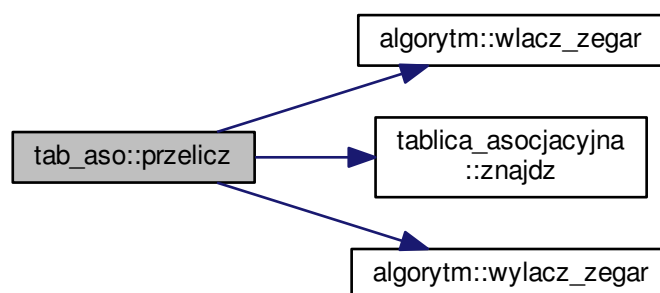
4.23.3.1 `float tab_aso::przelicz ( )` `[virtual]`

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:



4.23.3.2 `void tab_aso::wczytaj_klucze ( ifstream & plik )`

wczytywanie kluczy

## Parametry

in	plik	- strumień z kluczami użytymi podczas testów
----	------	--

Definicja w linii 241 pliku algorytm.cpp.

#### 4.23.4 Dokumentacja atrybutów składowych

##### 4.23.4.1 `tablica_asocjacyjna<float> tab_aso::d` [private]

tablica asocjacyjna

Definicja w linii 267 pliku algorytm.hh.

##### 4.23.4.2 `string* tab_aso::klucze` [private]

wczytywanie kluczy

## Parametry

in	plik	- 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](#)

#### 4.24 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)  
*Metoda sprawdza, czy element o kluczu k znajduje się w strukturze.*
- bool `czy_pusta` ()
- int `zlicz_elementy` ()
- void `wypisz` ()  
*wypisuje wszystkie elementy tablicy*
- TYP `wez` (int ind)  
*Metoda wprost odnosi się do konkretnego elementu tablicy - metoda używana przy grafie.*



- string `wez_id` (int ind)  
*metoda wprost odnosi sie do klucza, konkretnego leemnetu tablicy - metoda uzywana przy grafie*
- bool `czy_blokada` ()  
*metoda sprawdza, czy dostep do tablicy jest mozliwy*
- void `zablokuj` ()  
*metoda blokuje dostep do tablicy*
- void `odblokuj` ()  
*metoda zezwala na dostep do tablicy*
- void `ustaw` (string k, TYP v)  
*metoda zmienia wartosc w miejscu, ktore wskazuje klucz k*
- void `wyczyszc` ()  
*resetuje wartosci wszystkich elementow tablicy - ustawia wartosci an wartosc 0*

### Metody prywatne

- void `insert` (int ind, string k, TYP v)  
*Metoda ktora umieszcza wartosc oraz jej klucz w zadanym miejscu. Gdy wartosc z kluczem jest dodawana w srodek struktury, dane na prawo od niej przesuwane sa o jeden w prawo. Gdy istnieje potrzeba powiekszenia tablicy, stosuje sie znany juz radzaj gospodarowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.*
- void `wstaw` (string k, TYP v, int ind\_l, int ind\_r)  
*Metoda szuka pozycji, w ktora nalezy dodac element, aby tablica byla posortowana alfabetycznie.*
- int `znajdz` (string k, int ind\_l, int ind\_r)  
*Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, tzn jest zawarty w strukturze, flaga found ustawiana jest na wartosc true.*

### Atrybuty prywatne

- string \* `key`  
*Tablica zawierajaca klucze poszukiwan.*
- TYP \* `value`  
*Tablica zawierajaca wartosci.*
- int `s`  
*rozmiar tablicy*
- int `sp`  
*rozmiar danych zapelniajacych tablice*
- bool `found`  
*flaga informujaca o tym, czy dany klucz znaleziono w zbiorze*
- bool `blok`

#### 4.24.1 Opis szczegółowy

```
template<typename TYP>class tablica_asocjacyjna< TYP >
```

Klasa modeluje tablice asocjacyjna.

Definicja w linii 19 pliku tablica\_asocjacyjna.hh.

## 4.24.2 Dokumentacja konstruktora i destruktora

### 4.24.2.1 `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 123 pliku `tablica_asocjacyjna.hh`.

## 4.24.3 Dokumentacja funkcji składowych

### 4.24.3.1 `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 222 pliku `tablica_asocjacyjna.hh`.

### 4.24.3.2 `template<typename TYP> bool tablica_asocjacyjna< TYP >::czy_pusta ( ) [inline]`

Zwraca

true, gdy stos jest pusty, false w przeciwnym wypadku

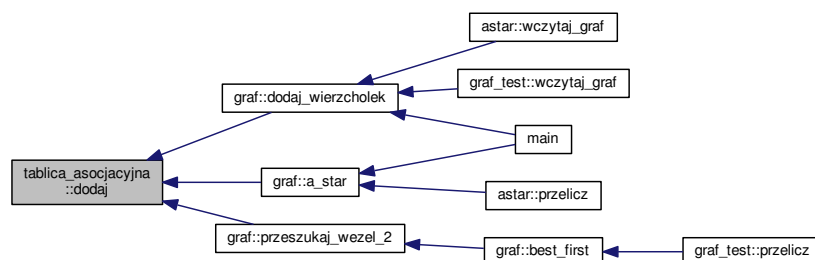
Definicja w linii 190 pliku `tablica_asocjacyjna.hh`.

### 4.24.3.3 `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 129 pliku `tablica_asocjacyjna.hh`.

Oto graf wywołań tej funkcji:



4.24.3.4 `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 36 pliku `tablica_asocjacyjna.hh`.

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

metoda zezwala na dostęp do tablicy

Definicja w linii 230 pliku `tablica_asocjacyjna.hh`.

4.24.3.6 `template<typename TYP> TYP tablica_asocjacyjna< TYP >::pobierz ( string k ) [inline]`

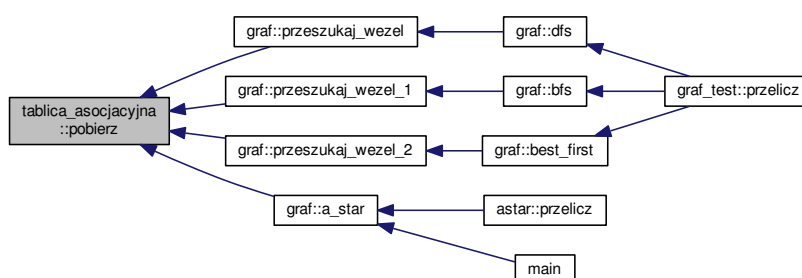
Metoda zwraca użytkownikowi szukany element, pod warunkiem, że jest on w zbiorze.

#### Zwraca

szukany element Gdy słownik jest pusty lub szukany element nie istnieje, użytkownik zostaje o tym poinformowany

Definicja w linii 170 pliku `tablica_asocjacyjna.hh`.

Oto graf wywołań tej funkcji:



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

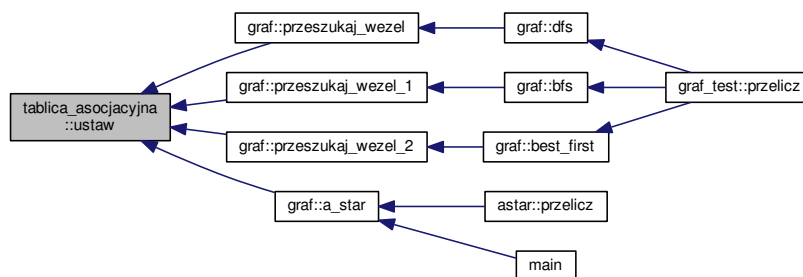
metoda zmienia wartość w miejscu, które wskazuje klucz *k*

#### Parametry

<i>in</i>	<i>k</i>	- klucz
<i>in</i>	<i>v</i>	wartość, która ma zastąpić dotychczasową wartość w tablicy

Definicja w linii 237 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływań tej funkcji:

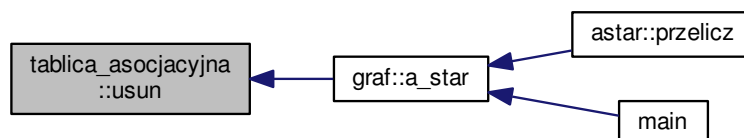


4.24.3.8 `template<typename TYP> void tablica_asocjacyjna< TYP >::usun ( string k ) [inline]`

Metoda usuwa zadany element, korzystając z funkcji znajdz.

Definicja w linii 144 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływań tej funkcji:



4.24.3.9 `template<typename TYP> TYP tablica_asocjacyjna< TYP >::wez ( int ind ) [inline]`

Metoda wprost odnosi się do konkretnego elementu tablicy - metoda używana przy grafie.

Parametry

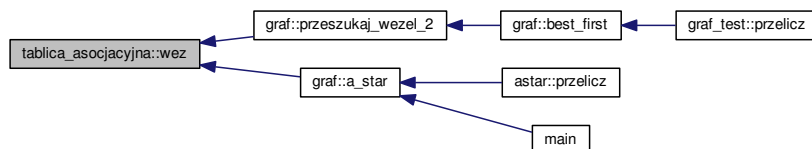
<code>in</code>	<code>ind</code>	- indeks tablicy
-----------------	------------------	------------------

**Zwraca**

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

Definicja w linii 206 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływań tej funkcji:



#### 4.24.3.10 `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**

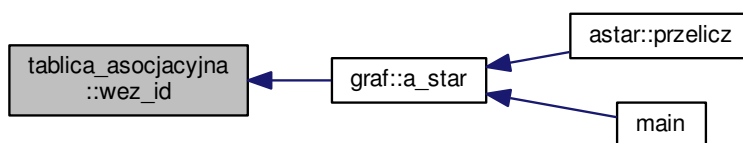
<code>in</code>	<code>ind</code>	- indeks tablicy
-----------------	------------------	------------------

**Zwraca**

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

Definicja w linii 214 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływań tej funkcji:



#### 4.24.3.11 `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.

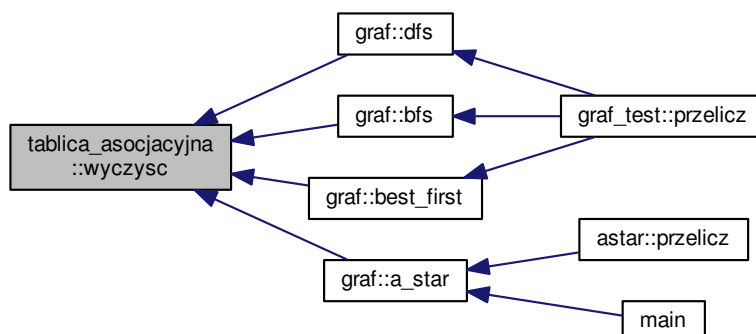
Definicja w linii 84 pliku `tablica_asocjacyjna.hh`.

#### 4.24.3.12 `template<typename TYP> void tablica_asocjacyjna< TYP >::wyczysc ( ) [inline]`

resetuje wartosci wszystkich elementow tablicy - ustawia wartosci an wartosc 0

Definicja w linii 242 pliku `tablica_asocjacyjna.hh`.

Oto graf wywołań tej funkcji:



**4.24.3.13** `template<typename TYP> void tablica_asocjacyjna< TYP >::wypisz ( ) [inline]`

wypisuje wszystkie elementy tablicy

Definicja w linii 197 pliku `tablica_asocjacyjna.hh`.

**4.24.3.14** `template<typename TYP> void tablica_asocjacyjna< TYP >::zablokuj ( ) [inline]`

metoda blokuje dostęp do tablicy

Definicja w linii 226 pliku `tablica_asocjacyjna.hh`.

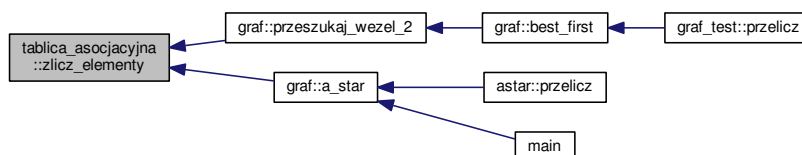
**4.24.3.15** `template<typename TYP> int tablica_asocjacyjna< TYP >::zlicz_elementy ( ) [inline]`

**Zwraca**

ilosc elementow w strukturze

Definicja w linii 195 pliku `tablica_asocjacyjna.hh`.

Oto graf wywołań tej funkcji:



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

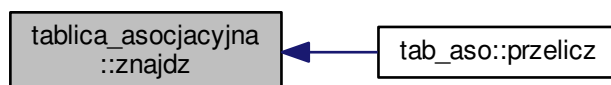
Metoda szuka w zbiorze zadanego klucza (przeszukiwanie binarne), gdy element zostanie odnaleziony, tzn jest zawarty w strukturze, flaga found ustawiana jest na wartosc true.

Zwraca

indeks szukanego elementu

Definicja w linii 101 pliku `tablica_asocjacyjna.hh`.

Oto graf wywoływań tej funkcji:



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

Metoda sprawdza, czy element o kluczu k znajduje sie w strukturze.

Parametry

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

Definicja w linii 182 pliku `tablica_asocjacyjna.hh`.

## 4.24.4 Dokumentacja atrybutów składowych

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

Definicja w linii 30 pliku `tablica_asocjacyjna.hh`.

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

flaga informujaca o tym, czy dany klucz znaleziono w zbiorze

Definicja w linii 29 pliku `tablica_asocjacyjna.hh`.

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

Tablica zawierajaca klucze poszukiwan.

Definicja w linii 21 pliku `tablica_asocjacyjna.hh`.

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

rozmiar tablicy

Definicja w linii 25 pliku `tablica_asocjacyjna.hh`.

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

rozmiar danych zapelniajacych tablice

Definicja w linii 27 pliku `tablica_asocjacyjna.hh`.

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

Tablica zawierajaca wartosci.

Definicja w linii 23 pliku `tablica_asocjacyjna.hh`.

Dokumentacja dla tej klasy została wygenerowana z pliku:

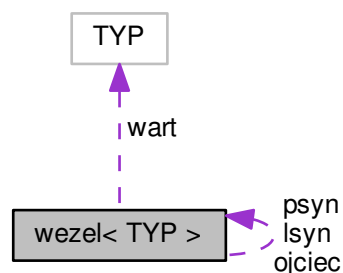
- [tablica\\_asocjacyjna.hh](#)

## 4.25 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*

#### 4.25.1 Opis szczegółowy

`template<typename TYP>class wezel< TYP >`

modeluje pojedynczy wezel drzewa

Definicja w linii 14 pliku drzewo.hh.

#### 4.25.2 Dokumentacja konstruktora i destruktor

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

konstruktor bezparametryczny

Definicja w linii 29 pliku drzewo.hh.

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

konstruktor parametryczny

Parametry

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

Definicja w linii 35 pliku drzewo.hh.

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

destruktor

Definicja w linii 37 pliku drzewo.hh.

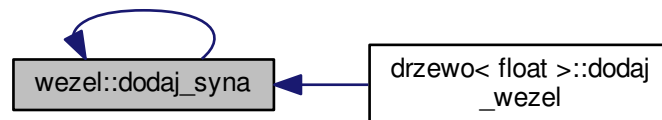
### 4.25.3 Dokumentacja funkcji składowych

4.25.3.1 `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:



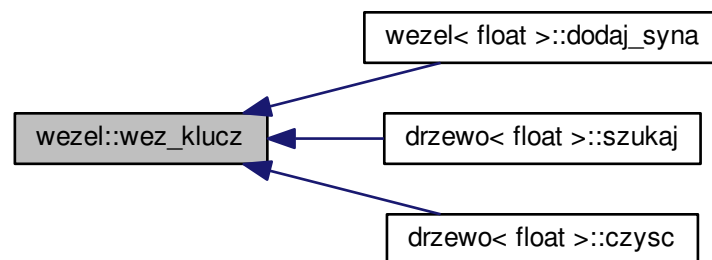
4.25.3.2 `template<typename TYP> string wezel< TYP >::wez_klucz ( ) [inline]`

Zwraca

klucz wezla

Definicja w linii 41 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



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

Zwraca

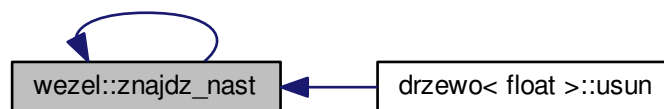
wartosc wezla

Definicja w linii 45 pliku drzewo.hh.

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

Definicja w linii 61 pliku `drzewo.hh`.

Oto graf wywoływań tej funkcji:



## 4.25.4 Dokumentacja atrybutów składowych

4.25.4.1 `template<typename TYP> syn wezel< TYP >::flag`

okresla, czyim synem jest wezel

Definicja w linii 21 pliku `drzewo.hh`.

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

klucz sluzacy do wyszukiwania

Definicja w linii 16 pliku `drzewo.hh`.

4.25.4.3 `template<typename TYP> wezel* wezel< TYP >::lsyn`

wskaznik na lewego syna wezla

Definicja w linii 25 pliku `drzewo.hh`.

4.25.4.4 `template<typename TYP> wezel* wezel< TYP >::ojciec`

wskaznik na ojca danego wezla

Definicja w linii 23 pliku `drzewo.hh`.

4.25.4.5 `template<typename TYP> wezel* wezel< TYP >::psyn`

wskaznik na prawego syna wezla

Definicja w linii 27 pliku `drzewo.hh`.

4.25.4.6 `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](#)

## 4.26 Dokumentacja klasy wierzcholek

Klasa modeluje pojecie wierzchołka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakię interpretowanie wierzchołka grafu, wedle zyczeń uzytkownika.

```
#include <graf.hh>
```

### Metody publiczne

- [wierzcholek](#) (int wz, int wg)

*konstruktor*

### Atrybuty prywatne

- int [id](#)

*numer indentyfikacyjny wierzchołka*

- int [waga](#)

*waga wezła, uzywana w stosunku do wierzchołka, z ktorym ow wierzcholek jest incydentny*

### Przyjaciele

- class [graf](#)

*klasa zparzyjawniona*

#### 4.26.1 Opis szczegółowy

Klasa modeluje pojecie wierzchołka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakię interpretowanie wierzchołka grafu, wedle zyczeń uzytkownika.

Definicja w linii 17 pliku graf.hh.

#### 4.26.2 Dokumentacja konstruktora i destruktora

##### 4.26.2.1 wierzcholek::wierzcholek ( int wz, int wg ) [inline]

konstruktor

Parametry

in	wz	- id wierzchołka
in	wg	- waga

Definicja w linii 30 pliku graf.hh.

#### 4.26.3 Dokumentacja przyjaciół i funkcji związanych

##### 4.26.3.1 friend class graf [friend]

klasa zparzyjawniona

Definicja w linii 19 pliku graf.hh.

#### 4.26.4 Dokumentacja atrybutów składowych

##### 4.26.4.1 `int wierzcholek::id` `[private]`

numer identyfikacyjny wierzchołka

Definicja w linii 22 pliku `graf.hh`.

##### 4.26.4.2 `int wierzcholek::waga` `[private]`

waga węzła, używana w stosunku do wierzchołka, z którym ow wierzcholek jest incydentny

Definicja w linii 24 pliku `graf.hh`.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [graf.hh](#)



## Rozdział 5

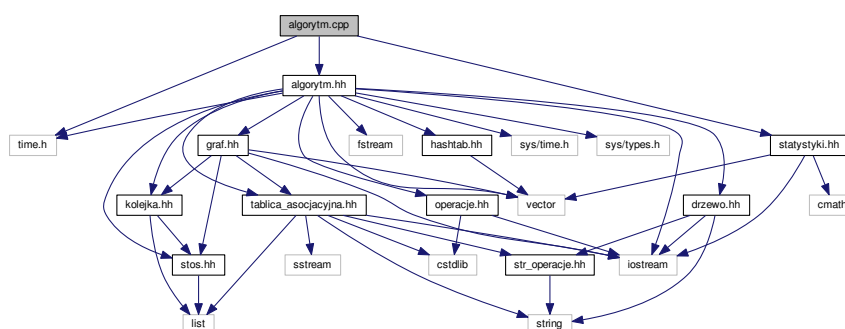
# Dokumentacja plików

### 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](#)

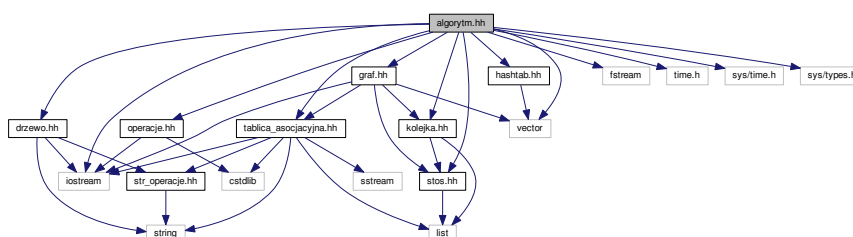
Definicja w pliku [algorytm.cpp](#).

### 5.2 Dokumentacja pliku algorytm.hh

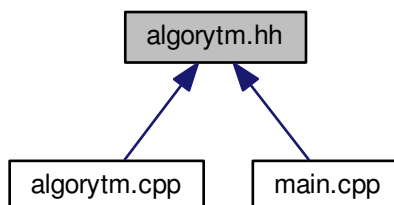
Definicja klas wykonujących operacje na zestawie danych wejściowych.

```
#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*
- class [astar](#)

### 5.2.1 Opis szczegółowy

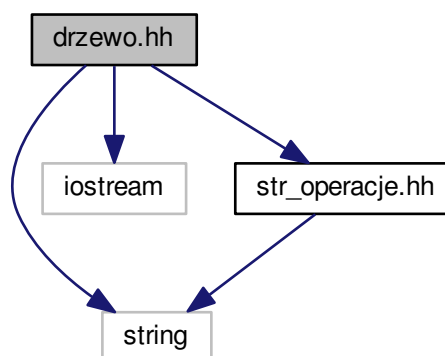
Definicja klas wykonujących operacje na zestawie danych wejściowych.

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

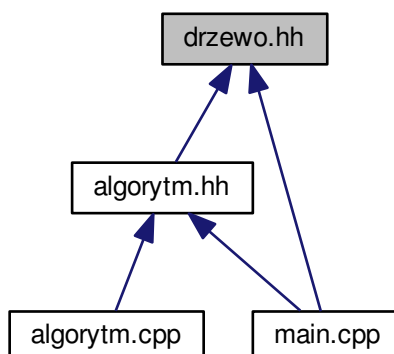
## 5.3 Dokumentacja pliku drzewo.hh

```
#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 wezel 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

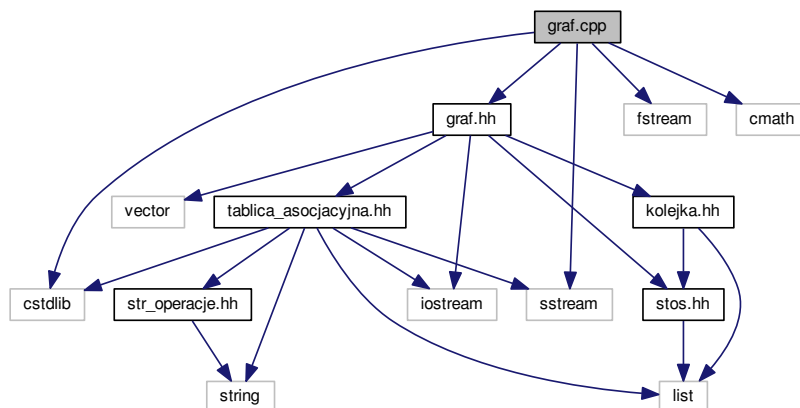
***lewy***  
***zaden***  
***prawy***

Definicja w linii 11 pliku `drzewo.hh`.

## 5.4 Dokumentacja pliku graf.cpp

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

Wykres zależności załączania dla graf.cpp:



### Zmienne

- `tablica_asocjacyjna< int > vec`

### 5.4.1 Dokumentacja zmiennych

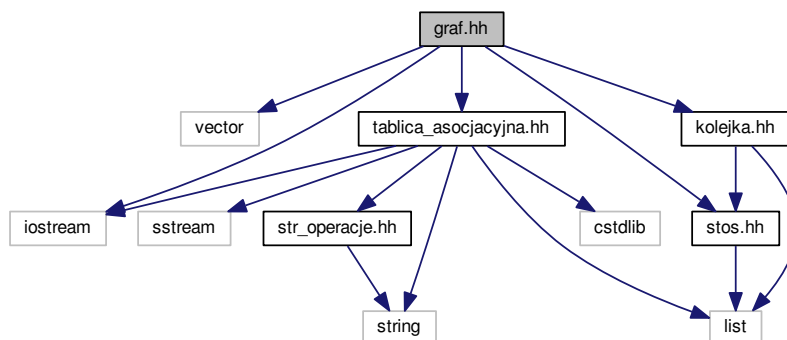
#### 5.4.1.1 `tablica_asocjacyjna<int> vec`

Definicja w linii 7 pliku graf.cpp.

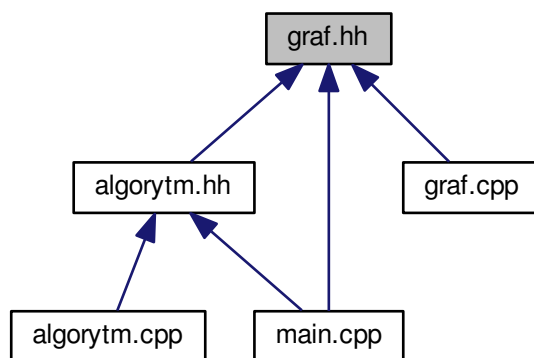
## 5.5 Dokumentacja pliku graf.hh

```
#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 życzenia 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

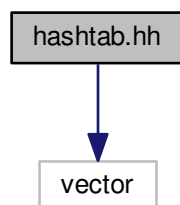
Plik zawiera definicje klasy wierzcholek i klasy graf

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

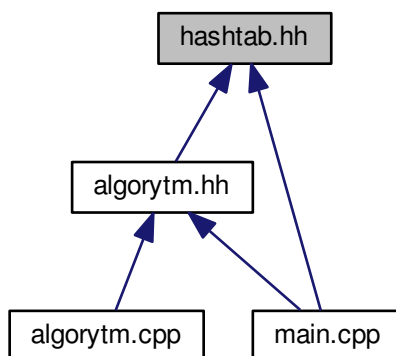
## 5.6 Dokumentacja pliku hashtab.hh

```
#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

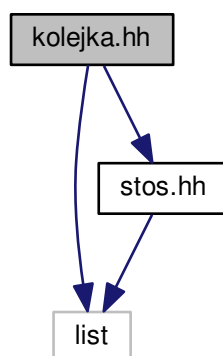
Definicja w pliku [hashtab.hh](#).

## 5.7 Dokumentacja pliku kolejka.hh

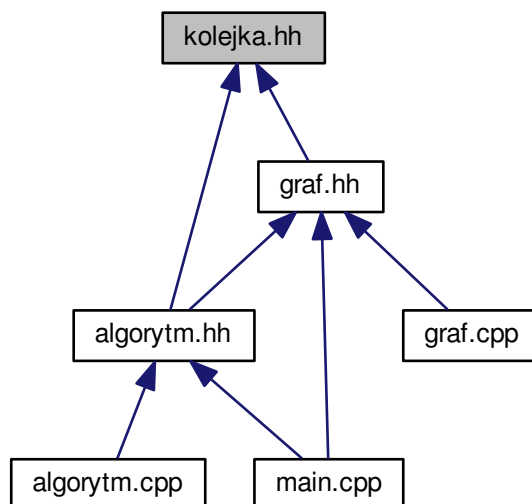
Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby.

```
#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 oparta 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

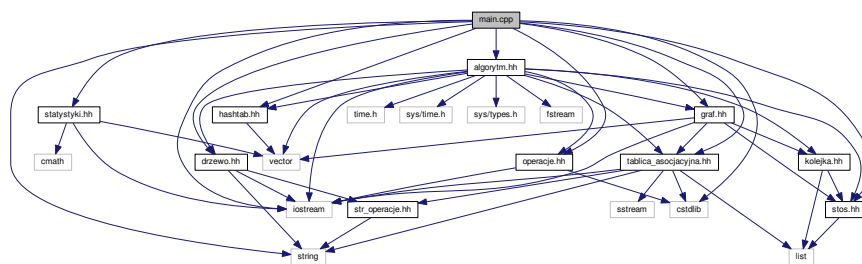
Definicja w pliku `kolejka.hh`.

## 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

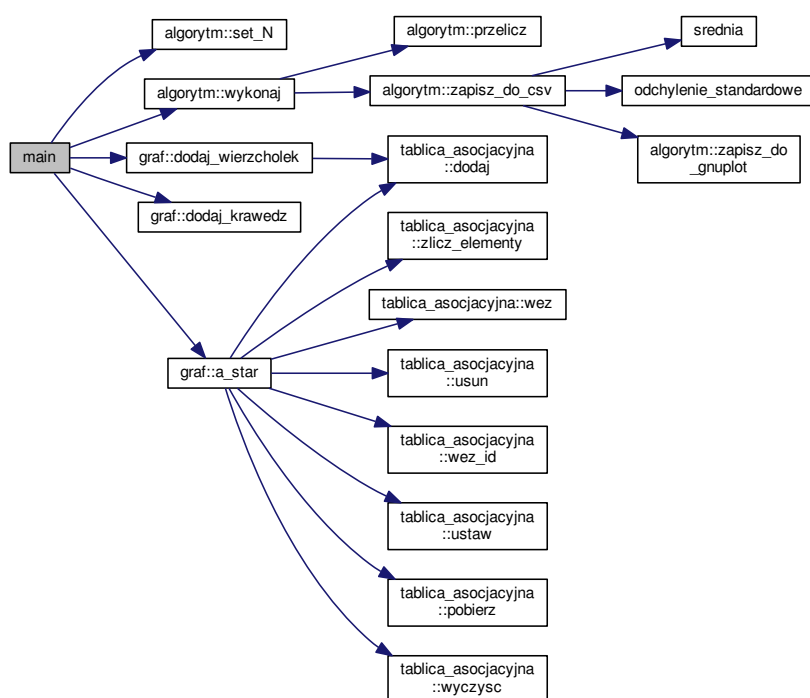
Definicja w pliku `main.cpp`.

## 5.8.2 Dokumentacja funkcji

### 5.8.2.1 `int main ( )`

Definicja w linii 20 pliku `main.cpp`.

Oto graf wywołań dla tej funkcji:

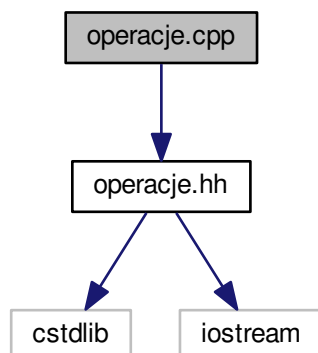


## 5.9 Dokumentacja pliku `operacje.cpp`

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



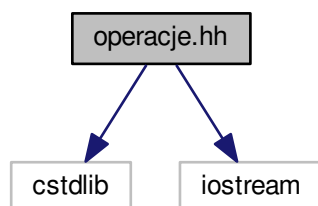
Wykres zależności załączania dla operacje.cpp:



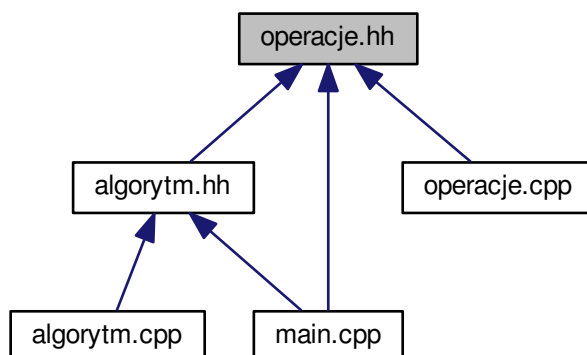
## 5.10 Dokumentacja pliku operacje.hh

```
#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`

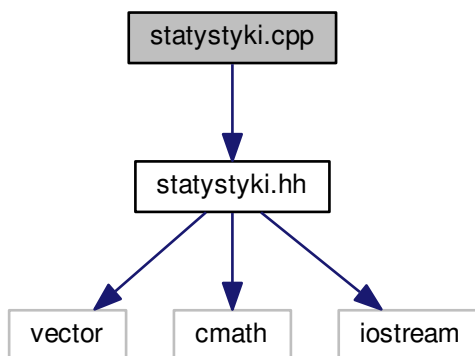
Definicja w linii 3 pliku `operacje.hh`.

## 5.11 Dokumentacja pliku `simplex.hh`

## 5.12 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*

### 5.12.1 Dokumentacja funkcji

#### 5.12.1.1 float odchylenie\_standardowe ( float *srednia*, float \* *tab*, int *rozmiar* )

funckja oblicza odchylenie standardowe

#### Parametry

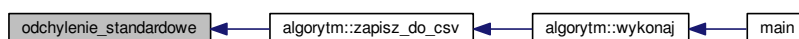
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

#### Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



### 5.12.1.2 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:

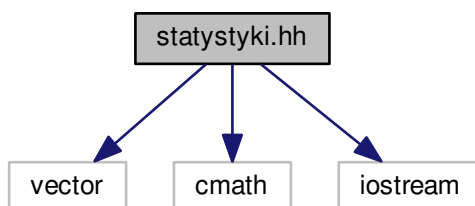


## 5.13 Dokumentacja pliku statystyki.hh

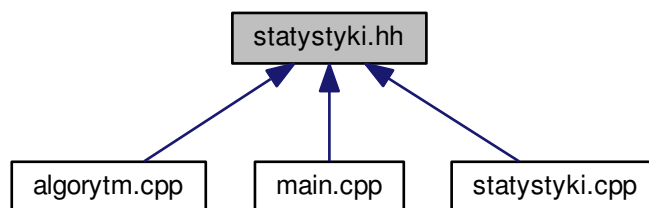
plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk

```
#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*

### 5.13.1 Opis szczegółowy

plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk

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

### 5.13.2 Dokumentacja funkcji

#### 5.13.2.1 float odchylenie\_standardowe ( float *srednia*, float \* *tab*, int *rozmiar* )

funckja oblicza odchylenie standardowe

#### Parametry

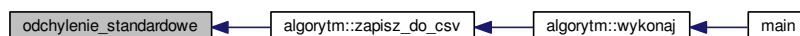
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

#### Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.13.2.2 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:

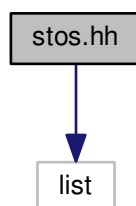


## 5.14 Dokumentacja pliku stos.hh

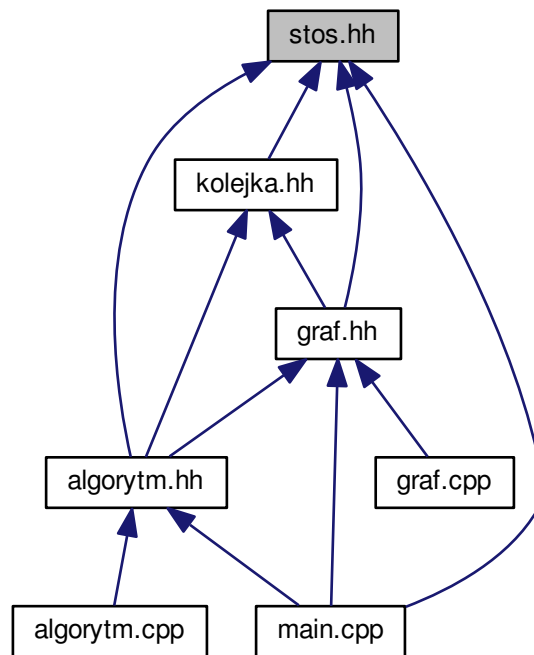
Plik zawiera definicje klasy `Stos` Zaimplementowana na 2 sposoby.

```
#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*

### 5.14.1 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`.



## 5.14.2 Dokumentacja typów wyliczanych

### 5.14.2.1 enum flag

typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

Wartości wyliczeń

***plus1***

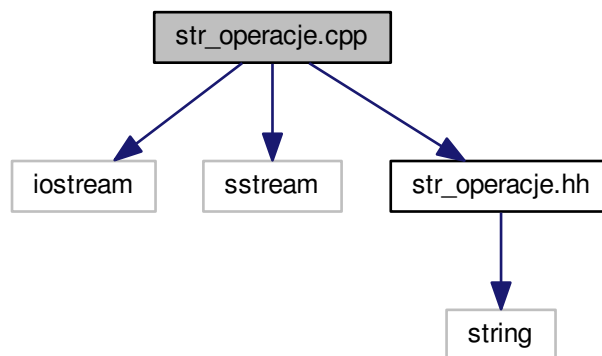
***x2***

Definicja w linii 17 pliku stos.hh.

## 5.15 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)

## 5.15.1 Dokumentacja funkcji

### 5.15.1.1 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.

**5.15.1.2 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.

**5.15.1.3 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 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.

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

Zwraca

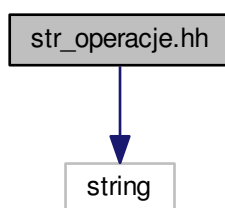
true, gdy s1 jest niżej w porządku alfabetycznym niż s2 lub gdy oba stringi są sobie równe, false, gdy s1 jest wyżej w porządku alfabetycznym niż s2

Definicja w linii 38 pliku str\_operacje.cpp.

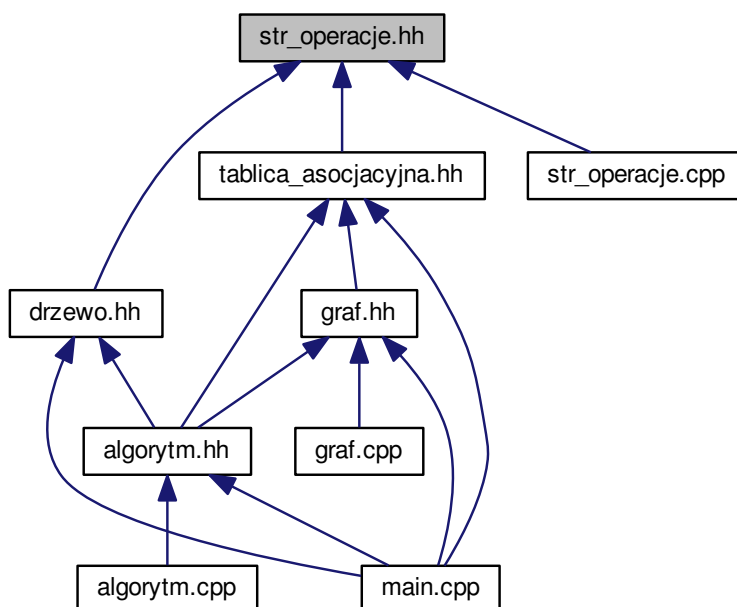
## 5.16 Dokumentacja pliku str\_operacje.hh

```
#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)

### 5.16.1 Dokumentacja funkcji

#### 5.16.1.1 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.

#### 5.16.1.2 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.

#### 5.16.1.3 bool operator== ( string s1, string s2 )

##### Zwraca

true, gdy łańcuchy są identyczne

Definicja w linii 42 pliku str\_operacje.cpp.

#### 5.16.1.4 bool operator> ( string s1, string s2 )

funkcja sluzaca do alfabetycznego porzadkowania napisow

##### Zwraca

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

Definicja w linii 21 pliku str\_operacje.cpp.

#### 5.16.1.5 bool operator>= ( string s1, string s2 )

##### Zwraca

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

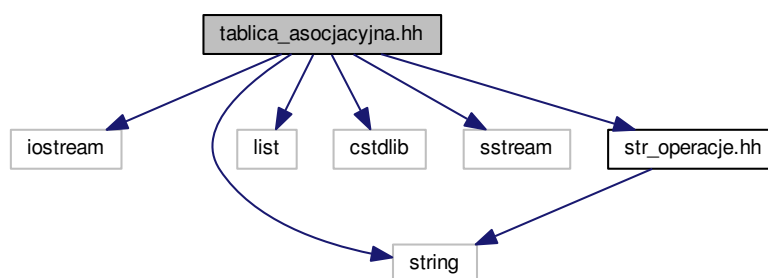
Definicja w linii 38 pliku str\_operacje.cpp.

## 5.17 Dokumentacja pliku strona.dox

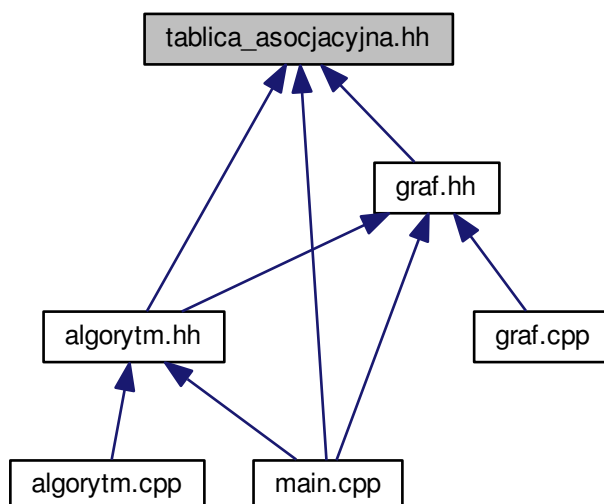
## 5.18 Dokumentacja pliku tablica\_asocjacyjna.hh

```
#include <iostream>
#include <string>
#include <list>
#include <cstdlib>
#include <sstream>
#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.*

### 5.18.1 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, [23](#)
- ~el\_tab
  - el\_tab, [28](#)
- ~wezel
  - wezel, [107](#)
- a\_star
  - graf, [31](#)
- 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, [113](#)
- algorytm.hh, [113](#)
- astar, [17](#)
  - astar, [19](#)
  - G1, [20](#)
  - G2, [20](#)
  - G3, [21](#)
  - G4, [21](#)
  - G5, [21](#)
  - G6, [21](#)
  - przelicz, [19](#)
  - wczytaj\_graf, [20](#)
- best\_first
  - graf, [32](#)
- bfs
  - graf, [34](#)
- blok
  - tablica\_asocjacyjna, [105](#)
- bst, [21](#)
  - ~bst, [23](#)
  - bst, [23](#)
  - d, [23](#)
  - klucze, [23](#)
  - przelicz, [23](#)
  - wczytaj\_klucze, [23](#)
- clear
  - queue\_array, [82](#)
  - queue\_list, [84](#)
  - stack\_array, [87](#)
  - stack\_list, [90](#)
- czas
  - algorytm, [16](#)
- czas1
  - algorytm, [16](#)
- czas2
  - algorytm, [16](#)
- czy\_blokada
  - tablica\_asocjacyjna, [100](#)
- czy\_pusta
  - tablica\_asocjacyjna, [100](#)
- czy\_sasiad
  - graf, [35](#), [36](#)
- czysc
  - drzewo, [25](#)
- d
  - bst, [23](#)
  - tab\_aso, [98](#)
- dane
  - algorytm, [17](#)
- dane\_wz
  - algorytm, [17](#)
- dequeue
  - queue\_array, [82](#)
  - queue\_list, [85](#)
- dfs
  - graf, [36](#)
- dist
  - graf, [47](#)
- dlugosc
  - hashtab, [61](#)
- dodaj
  - drzewo, [25](#)
  - hashtab, [57](#)
  - tablica\_asocjacyjna, [100](#)
- dodaj\_element
  - operacje, [72](#)

dodaj\_elementy  
     operacje, 73  
 dodaj\_krawedz  
     graf, 37, 38  
 dodaj\_syna  
     wezel, 108  
 dodaj\_wezel  
     drzewo, 25  
 dodaj\_wierzcholek  
     graf, 38–40  
 drzewo  
     czysc, 25  
     dodaj, 25  
     dodaj\_wezel, 25  
     drzewo, 25  
     korzen, 26  
     szukaj, 25  
     usun, 26  
     wyczysc, 26  
     znajdz, 26  
     znaleziony, 26  
 drzewo< TYP >, 24  
 drzewo.hh  
     lewy, 116  
     prawy, 116  
     zaden, 116  
 drzewo.hh, 115  
     syn, 116  
  
 el\_tab  
     ~el\_tab, 28  
     el\_tab, 28  
     el\_tab, 28  
     klucz, 28  
     wart, 28  
     zajety, 28  
 el\_tab< TYP >, 27  
 enqueue  
     queue\_array, 82  
     queue\_list, 85  
 est  
     graf, 47  
  
 f  
     queue\_array, 83  
     stack\_array, 89  
 flag  
     stos.hh, 131  
     wezel, 109  
 found  
     tablica\_asocjacyjna, 105  
  
 G1  
     astar, 20  
     graf\_test, 51  
 G2  
     astar, 20  
     graf\_test, 51  
 G3  
     astar, 21  
     graf\_test, 51  
 G4  
     astar, 21  
     graf\_test, 51  
 G5  
     astar, 21  
     graf\_test, 51  
 G6  
     astar, 21  
     graf\_test, 51  
 graf, 28  
     a\_star, 31  
     best\_first, 32  
     bfs, 34  
     czy\_sasiad, 35, 36  
     dfs, 36  
     dist, 47  
     dodaj\_krawedz, 37, 38  
     dodaj\_wierzcholek, 38–40  
     est, 47  
     graf, 31  
     lista\_incydencji, 47  
     poprzednik, 47  
     przeszukaj\_wezel, 40  
     przeszukaj\_wezel\_1, 41  
     przeszukaj\_wezel\_2, 42  
     Q, 47  
     rysuj, 43  
     sasiedztwo, 44  
     tab, 47  
     usun\_krawedz, 44  
     usun\_wierzcholek, 46  
     w\_x, 48  
     w\_y, 48  
     wierzcholek, 110  
     wyczysc, 47  
     wypisz\_liste, 47  
 graf.cpp, 117  
     vec, 117  
 graf.hh, 117  
 graf\_test, 48  
     G1, 51  
     G2, 51  
     G3, 51  
     G4, 51  
     G5, 51  
     G6, 51  
     graf\_test, 50  
     graf\_test, 50  
     przelicz, 50  
     typ, 51  
     wczytaj\_graf, 50  
  
 h\_sort, 52  
     h\_sort, 53  
     h\_sort, 53  
     przelicz, 53  
 h\_table, 53



- h\_table, 55
- h\_table, 55
- klucze, 55
- przelicz, 55
- wczytaj\_klucze, 55
- hash
  - hashtab, 58
- hashtab
  - dlugosc, 61
  - dodaj, 57
  - hash, 58
  - hashtab, 57
  - tab, 61
  - ustaw\_dlugosc, 58
  - usun, 58
  - wypisz, 60
  - znajdz, 60
- hashtab< TYP >, 56
- hashtab.hh, 119
- heap\_sort
  - operacje, 73
- id
  - wierzcholek, 111
- ile\_danych
  - algorytm, 10
- insert
  - tablica\_asocjacyjna, 100
- is\_empty
  - queue\_array, 83
  - queue\_list, 85
  - stack\_array, 87
  - stack\_list, 90
- jaki\_czas
  - algorytm, 10
- key
  - tablica\_asocjacyjna, 105
- klucz
  - el\_tab, 28
  - wezel, 109
- klucze
  - bst, 23
  - h\_table, 55
  - tab\_aso, 98
- kolejka.hh, 120
- kolejka\_lista, 61
  - kolejka\_lista, 62
  - kolejka\_lista, 62
  - przelicz, 62
  - qu, 63
- kolejka\_tablica, 63
  - kolejka\_tablica, 64
  - kolejka\_tablica, 64
  - przelicz, 64
  - qu, 65
- korzen
  - drzewo, 26
- lewy
  - drzewo.hh, 116
- lista\_incydencji
  - graf, 47
- lsyn
  - wezel, 109
- m
  - algorytm, 17
- m\_sort, 65
  - m\_sort, 67
  - m\_sort, 67
  - przelicz, 67
- main
  - main.cpp, 122
- main.cpp, 121
- main, 122
- make\_heap
  - operacje, 73
- make\_node
  - operacje, 74
- merge
  - operacje, 74
- merge\_sort
  - operacje, 75
- mnozenie, 67
  - mnozenie, 68
  - przelicz, 70
- n
  - algorytm, 17
  - operacje, 78
- odblokuj
  - tablica\_asocjacyjna, 101
- odchylenie\_standardowe
  - statystyki.cpp, 125
  - statystyki.hh, 127
- odwroc\_tablice
  - operacje, 75
- ojciec
  - wezel, 109
- op
  - algorytm, 17
- operacje, 70
  - dodaj\_element, 72
  - dodaj\_elementy, 73
  - heap\_sort, 73
  - make\_heap, 73
  - make\_node, 74
  - merge, 74
  - merge\_sort, 75
  - n, 78
  - odwroc\_tablice, 75
  - operacje, 71
  - operator=, 75
  - operator==, 77
  - quick\_sort, 77
  - tab, 78

- zamien\_elementy, 77
- operacje.cpp, 122
- operacje.hh, 123
- ROZMIAR, 124
- operator<
  - str\_operacje.cpp, 131
  - str\_operacje.hh, 134
- operator<=
  - str\_operacje.cpp, 132
  - str\_operacje.hh, 134
- operator>
  - str\_operacje.cpp, 132
  - str\_operacje.hh, 134
- operator>=
  - str\_operacje.cpp, 132
  - str\_operacje.hh, 134
- operator=
  - operacje, 75
- operator==
  - operacje, 77
  - str\_operacje.cpp, 132
  - str\_operacje.hh, 134
- plus1
  - stos.hh, 131
- pobierz
  - tablica\_asocjacyjna, 101
- pop
  - stack\_array, 88
  - stack\_list, 90
- poprzednik
  - graf, 47
- porownaj
  - algorytm, 10
- prawy
  - drzewo.hh, 116
- przelicz
  - algorytm, 11
  - astar, 19
  - bst, 23
  - graf\_test, 50
  - h\_sort, 53
  - h\_table, 55
  - kolejka\_lista, 62
  - kolejka\_tablica, 64
  - m\_sort, 67
  - mnozenie, 70
  - q\_sort, 80
  - stos\_lista, 92
  - stos\_tablica, 94
  - tab\_aso, 97
- przeszukaj\_wezel
  - graf, 40
- przeszukaj\_wezel\_1
  - graf, 41
- przeszukaj\_wezel\_2
  - graf, 42
- psyn
  - wezel, 109
- push
  - stack\_array, 88
  - stack\_list, 90
- Q
  - graf, 47
- q
  - queue\_array, 83
  - queue\_list, 86
- q\_sort, 78
  - przelicz, 80
  - q\_sort, 80
  - q\_sort, 80
- qu
  - kolejka\_lista, 63
  - kolejka\_tablica, 65
- queue\_array
  - clear, 82
  - dequeue, 82
  - enqueue, 82
  - f, 83
  - is\_empty, 83
  - q, 83
  - queue\_array, 82
  - queue\_array, 82
  - s, 83
  - size, 83
  - sp, 83
- queue\_array< TYP >, 80
- queue\_list
  - clear, 84
  - dequeue, 85
  - enqueue, 85
  - is\_empty, 85
  - q, 86
  - size, 85
- queue\_list< TYP >, 84
- quick\_sort
  - operacje, 77
- ROZMIAR
  - operacje.hh, 124
- rysuj
  - graf, 43
- s
  - queue\_array, 83
  - stack\_array, 89
  - tablica\_asocjacyjna, 105
- sasiedztwo
  - graf, 44
- set\_N
  - algorytm, 11
- simplex.hh, 124
- size
  - queue\_array, 83
  - queue\_list, 85
  - stack\_array, 88
  - stack\_list, 91

sp  
    queue\_array, 83  
    stack\_array, 89  
    tablica\_asocjacyjna, 105  
srednia  
    statystyki.cpp, 126  
    statystyki.hh, 127  
st  
    stack\_array, 89  
    stack\_list, 91  
stack\_array  
    clear, 87  
    f, 89  
    is\_empty, 87  
    pop, 88  
    push, 88  
    s, 89  
    size, 88  
    sp, 89  
    st, 89  
    stack\_array, 87  
    stack\_array, 87  
stack\_array< TYP >, 86  
stack\_list  
    clear, 90  
    is\_empty, 90  
    pop, 90  
    push, 90  
    size, 91  
    st, 91  
stack\_list< TYP >, 89  
statystyki.cpp, 125  
    odchylenie\_standardowe, 125  
    srednia, 126  
statystyki.hh, 126  
    odchylenie\_standardowe, 127  
    srednia, 127  
stos  
    stos\_lista, 93  
    stos\_tablica, 95  
stos.hh  
    plus1, 131  
    x2, 131  
stos.hh, 129  
    flag, 131  
stos\_lista, 91  
    przelicz, 92  
    stos, 93  
    stos\_lista, 92  
    stos\_lista, 92  
stos\_tablica, 93  
    przelicz, 94  
    stos, 95  
    stos\_tablica, 94  
    stos\_tablica, 94  
str\_operacje.cpp, 131  
    operator<, 131  
    operator<=, 132  
    operator>, 132  
    operator>=, 132  
    operator==, 132  
str\_operacje.hh, 133  
    operator<, 134  
    operator<=, 134  
    operator>, 134  
    operator>=, 134  
    operator==, 134  
strona.dox, 135  
syn  
    drzewo.hh, 116  
szukaj  
    drzewo, 25  
  
tab  
    graf, 47  
    hashtab, 61  
    operacje, 78  
tab\_aso, 95  
    d, 98  
    klucze, 98  
    przelicz, 97  
    tab\_aso, 97  
    tab\_aso, 97  
    wczytaj\_klucze, 97  
tablica\_asocjacyjna  
    blok, 105  
    czy\_blokada, 100  
    czy\_pusta, 100  
    dodaj, 100  
    found, 105  
    insert, 100  
    key, 105  
    odblokuj, 101  
    pobierz, 101  
    s, 105  
    sp, 105  
    tablica\_asocjacyjna, 100  
    tablica\_asocjacyjna, 100  
    ustaw, 101  
    usun, 102  
    value, 106  
    wez, 102  
    wez\_id, 103  
    wstaw, 103  
    wyczysc, 103  
    wypisz, 104  
    zablokuj, 104  
    zlicz\_elementy, 104  
    znajdz, 104, 105  
tablica\_asocjacyjna< TYP >, 98  
tablica\_asocjacyjna.hh, 135  
typ  
    graf\_test, 51  
  
ustaw  
    tablica\_asocjacyjna, 101  
ustaw\_dlugosc

- hashtab, 58
- usun
  - drzewo, 26
  - hashtab, 58
  - tablica\_asocjacyjna, 102
- usun\_krawedz
  - graf, 44
- usun\_wierzcholek
  - graf, 46
- value
  - tablica\_asocjacyjna, 106
- vec
  - graf.cpp, 117
- w\_x
  - graf, 48
- w\_y
  - graf, 48
- waga
  - wierzcholek, 111
- wart
  - el\_tab, 28
  - wezel, 109
- wczytaj
  - algorytm, 11
- wczytaj\_graf
  - astar, 20
  - graf\_test, 50
- wczytaj\_klucze
  - bst, 23
  - h\_table, 55
  - tab\_aso, 97
- wczytaj\_wzor
  - algorytm, 12
- wez
  - tablica\_asocjacyjna, 102
- wez\_id
  - tablica\_asocjacyjna, 103
- wez\_klucz
  - wezel, 108
- wez\_wart
  - wezel, 108
- wezel
  - ~wezel, 107
  - dodaj\_syna, 108
  - flag, 109
  - klucz, 109
  - lsyn, 109
  - ojciec, 109
  - psyn, 109
  - wart, 109
  - wez\_klucz, 108
  - wez\_wart, 108
  - wezel, 107
  - znajdz\_nast, 108
- wezel< TYP >, 106
- wierzcholek, 110
  - graf, 110
- id, 111
- waga, 111
- wierzcholek, 110
- wlacz\_zegar
  - algorytm, 12
- wstaw
  - tablica\_asocjacyjna, 103
- wyczysc
  - drzewo, 26
  - graf, 47
  - tablica\_asocjacyjna, 103
- wykonaj
  - algorytm, 13
- wylacz\_zegar
  - algorytm, 14
- wypisz
  - hashtab, 60
  - tablica\_asocjacyjna, 104
- wypisz\_liste
  - graf, 47
- x2
  - stos.hh, 131
- zablokuj
  - tablica\_asocjacyjna, 104
- zaden
  - drzewo.hh, 116
- zajety
  - el\_tab, 28
- zamien\_elementy
  - operacje, 77
- zapisz\_do\_csv
  - algorytm, 15
- zapisz\_do\_gnuplot
  - algorytm, 16
- zlicz\_elementy
  - tablica\_asocjacyjna, 104
- znajdz
  - drzewo, 26
  - hashtab, 60
  - tablica\_asocjacyjna, 104, 105
- znajdz\_nast
  - wezel, 108
- znaleziony
  - drzewo, 26