

PAMSI - pwilkosz

1.0

Wygenerowano przez Doxygen 1.7.6.1

Sun Apr 27 2014 22:27:42



# Spis treści

<b>1</b>	<b>Indeks klas</b>	<b>1</b>
1.1	Hierarchia klas . . . . .	1
<b>2</b>	<b>Indeks klas</b>	<b>3</b>
2.1	Lista klas . . . . .	3
<b>3</b>	<b>Indeks plików</b>	<b>5</b>
3.1	Lista plików . . . . .	5
<b>4</b>	<b>Dokumentacja klas</b>	<b>7</b>
4.1	Dokumentacja klasy algorytm . . . . .	7
4.1.1	Opis szczegółowy . . . . .	10
4.1.2	Dokumentacja konstruktora i destruktora . . . . .	10
4.1.2.1	algorytm . . . . .	10
4.1.3	Dokumentacja funkcji składowych . . . . .	11
4.1.3.1	ile_danych . . . . .	11
4.1.3.2	jaki_czas . . . . .	11
4.1.3.3	porownaj . . . . .	11
4.1.3.4	przelicz . . . . .	11
4.1.3.5	set_N . . . . .	12
4.1.3.6	wczytaj . . . . .	12
4.1.3.7	wczytaj_wzor . . . . .	12
4.1.3.8	wlacz_zegar . . . . .	12
4.1.3.9	wykonaj . . . . .	14
4.1.3.10	wylacz_zegar . . . . .	14
4.1.3.11	zapisz_do_csv . . . . .	15

4.1.3.12	<code>zapisz_do_gnuplot</code>	16
4.1.4	Dokumentacja atrybutów składowych	17
4.1.4.1	<code>czas</code>	17
4.1.4.2	<code>czas1</code>	17
4.1.4.3	<code>czas2</code>	17
4.1.4.4	<code>dane</code>	17
4.1.4.5	<code>dane_wz</code>	17
4.1.4.6	<code>m</code>	17
4.1.4.7	<code>n</code>	17
4.1.4.8	<code>op</code>	17
4.2	Dokumentacja klasy <code>bst</code>	18
4.2.1	Opis szczegółowy	20
4.2.2	Dokumentacja konstruktora i destruktora	20
4.2.2.1	<code>bst</code>	20
4.2.2.2	<code>~bst</code>	20
4.2.3	Dokumentacja funkcji składowych	20
4.2.3.1	<code>przelicz</code>	20
4.2.3.2	<code>wczytaj_klucze</code>	21
4.2.4	Dokumentacja atrybutów składowych	21
4.2.4.1	<code>d</code>	21
4.2.4.2	<code>klucze</code>	21
4.3	Dokumentacja szablonu klasy <code>drzewo&lt; TYP &gt;</code>	21
4.3.1	Opis szczegółowy	22
4.3.2	Dokumentacja konstruktora i destruktora	22
4.3.2.1	<code>drzewo</code>	22
4.3.2.2	<code>drzewo</code>	23
4.3.3	Dokumentacja funkcji składowych	23
4.3.3.1	<code>czysc</code>	23
4.3.3.2	<code>dodaj</code>	23
4.3.3.3	<code>dodaj_wezel</code>	23
4.3.3.4	<code>szukaj</code>	23
4.3.3.5	<code>usun</code>	24
4.3.3.6	<code>wyczysc</code>	24
4.3.3.7	<code>znajdz</code>	24

4.3.4	Dokumentacja atrybutów składowych . . . . .	25
4.3.4.1	korzen . . . . .	25
4.3.4.2	znaleziony . . . . .	25
4.4	Dokumentacja szablonu klasy el_tab< TYP > . . . . .	25
4.4.1	Opis szczegółowy . . . . .	26
4.4.2	Dokumentacja konstruktora i destruktora . . . . .	26
4.4.2.1	el_tab . . . . .	26
4.4.2.2	~el_tab . . . . .	27
4.4.3	Dokumentacja atrybutów składowych . . . . .	27
4.4.3.1	klucz . . . . .	27
4.4.3.2	wart . . . . .	27
4.4.3.3	zajety . . . . .	27
4.5	Dokumentacja klasy graf . . . . .	27
4.5.1	Opis szczegółowy . . . . .	29
4.5.2	Dokumentacja konstruktora i destruktora . . . . .	29
4.5.2.1	graf . . . . .	29
4.5.3	Dokumentacja funkcji składowych . . . . .	30
4.5.3.1	czy_sasiad . . . . .	30
4.5.3.2	czy_sasiad . . . . .	30
4.5.3.3	dfs . . . . .	31
4.5.3.4	dodaj_krawedz . . . . .	32
4.5.3.5	dodaj_krawedz . . . . .	32
4.5.3.6	dodaj_wierzcholek . . . . .	33
4.5.3.7	dodaj_wierzcholek . . . . .	34
4.5.3.8	przeszukaj_wezel . . . . .	34
4.5.3.9	sasiedztwo . . . . .	35
4.5.3.10	sasiedztwo . . . . .	36
4.5.3.11	usun_krawedz . . . . .	36
4.5.3.12	usun_krawedz . . . . .	36
4.5.3.13	usun_wierzcholek . . . . .	37
4.5.3.14	usun_wierzcholek . . . . .	37
4.5.3.15	wyczysc . . . . .	38
4.5.3.16	wypisz_liste . . . . .	38
4.5.4	Dokumentacja atrybutów składowych . . . . .	38

4.5.4.1	list <sub>a</sub> _incydencji . . . . .	38
4.5.4.2	S . . . . .	38
4.5.4.3	tab . . . . .	39
4.6	Dokumentacja klasy h <sub>sort</sub> . . . . .	39
4.6.1	Opis szczegółowy . . . . .	40
4.6.2	Dokumentacja konstruktora i destruktora . . . . .	40
4.6.2.1	h <sub>sort</sub> . . . . .	40
4.6.3	Dokumentacja funkcji składowych . . . . .	40
4.6.3.1	przelicz . . . . .	41
4.7	Dokumentacja klasy h <sub>table</sub> . . . . .	41
4.7.1	Opis szczegółowy . . . . .	42
4.7.2	Dokumentacja konstruktora i destruktora . . . . .	43
4.7.2.1	h <sub>table</sub> . . . . .	43
4.7.3	Dokumentacja funkcji składowych . . . . .	43
4.7.3.1	przelicz . . . . .	43
4.7.3.2	wczytaj_klucze . . . . .	43
4.7.4	Dokumentacja atrybutów składowych . . . . .	44
4.7.4.1	klucze . . . . .	44
4.8	Dokumentacja szablonu klasy hashtable< TYP > . . . . .	44
4.8.1	Opis szczegółowy . . . . .	45
4.8.2	Dokumentacja konstruktora i destruktora . . . . .	45
4.8.2.1	hashtable . . . . .	45
4.8.2.2	hashtable . . . . .	45
4.8.3	Dokumentacja funkcji składowych . . . . .	45
4.8.3.1	dodaj . . . . .	45
4.8.3.2	hash . . . . .	46
4.8.3.3	ustaw_dlugosc . . . . .	47
4.8.3.4	usun . . . . .	47
4.8.3.5	wypisz . . . . .	47
4.8.3.6	znajdz . . . . .	48
4.8.4	Dokumentacja atrybutów składowych . . . . .	48
4.8.4.1	dlugosc . . . . .	48
4.8.4.2	tab . . . . .	49
4.9	Dokumentacja klasy kolejka_list <sub>a</sub> . . . . .	49

4.9.1	Opis szczegółowy . . . . .	50
4.9.2	Dokumentacja konstruktora i destruktora . . . . .	50
4.9.2.1	kolejka_list . . . . .	51
4.9.3	Dokumentacja funkcji składowych . . . . .	51
4.9.3.1	przelicz . . . . .	51
4.9.4	Dokumentacja atrybutów składowych . . . . .	51
4.9.4.1	qu . . . . .	51
4.10	Dokumentacja klasy kolejka_tablica . . . . .	52
4.10.1	Opis szczegółowy . . . . .	53
4.10.2	Dokumentacja konstruktora i destruktora . . . . .	53
4.10.2.1	kolejka_tablica . . . . .	53
4.10.3	Dokumentacja funkcji składowych . . . . .	53
4.10.3.1	przelicz . . . . .	53
4.10.4	Dokumentacja atrybutów składowych . . . . .	54
4.10.4.1	qu . . . . .	54
4.11	Dokumentacja klasy m_sort . . . . .	54
4.11.1	Opis szczegółowy . . . . .	56
4.11.2	Dokumentacja konstruktora i destruktora . . . . .	56
4.11.2.1	m_sort . . . . .	56
4.11.3	Dokumentacja funkcji składowych . . . . .	56
4.11.3.1	przelicz . . . . .	56
4.12	Dokumentacja klasy mnozenie . . . . .	56
4.12.1	Opis szczegółowy . . . . .	58
4.12.2	Dokumentacja konstruktora i destruktora . . . . .	58
4.12.2.1	mnozenie . . . . .	58
4.12.3	Dokumentacja funkcji składowych . . . . .	58
4.12.3.1	przelicz . . . . .	58
4.13	Dokumentacja klasy operacje . . . . .	59
4.13.1	Opis szczegółowy . . . . .	60
4.13.2	Dokumentacja konstruktora i destruktora . . . . .	60
4.13.2.1	operacje . . . . .	60
4.13.2.2	operacje . . . . .	60
4.13.3	Dokumentacja funkcji składowych . . . . .	60
4.13.3.1	dodaj_element . . . . .	60

4.13.3.2	dodaj_elementy . . . . .	60
4.13.3.3	heap_sort . . . . .	61
4.13.3.4	make_heap . . . . .	61
4.13.3.5	make_node . . . . .	62
4.13.3.6	merge . . . . .	63
4.13.3.7	merge_sort . . . . .	63
4.13.3.8	odwroc_tablice . . . . .	64
4.13.3.9	operator= . . . . .	64
4.13.3.10	operator== . . . . .	64
4.13.3.11	operator[] . . . . .	64
4.13.3.12	quick_sort . . . . .	65
4.13.3.13	zamien_elementy . . . . .	65
4.13.4	Dokumentacja atrybutów składowych . . . . .	66
4.13.4.1	n . . . . .	66
4.13.4.2	tab . . . . .	66
4.14	Dokumentacja klasy q_sort . . . . .	66
4.14.1	Opis szczegółowy . . . . .	68
4.14.2	Dokumentacja konstruktora i destruktora . . . . .	68
4.14.2.1	q_sort . . . . .	68
4.14.3	Dokumentacja funkcji składowych . . . . .	68
4.14.3.1	przelicz . . . . .	68
4.15	Dokumentacja szablonu klasy queue_array< TYP > . . . . .	68
4.15.1	Opis szczegółowy . . . . .	70
4.15.2	Dokumentacja konstruktora i destruktora . . . . .	70
4.15.2.1	queue_array . . . . .	70
4.15.2.2	queue_array . . . . .	70
4.15.3	Dokumentacja funkcji składowych . . . . .	70
4.15.3.1	clear . . . . .	70
4.15.3.2	dequeue . . . . .	71
4.15.3.3	enqueue . . . . .	71
4.15.3.4	is_empty . . . . .	71
4.15.3.5	size . . . . .	71
4.15.4	Dokumentacja atrybutów składowych . . . . .	71
4.15.4.1	f . . . . .	72

4.15.4.2 <code>q</code>	72
4.15.4.3 <code>s</code>	72
4.15.4.4 <code>sp</code>	72
4.16 Dokumentacja szablonu klasy <code>queue_list&lt;TYP&gt;</code>	72
4.16.1 Opis szczegółowy	73
4.16.2 Dokumentacja funkcji składowych	73
4.16.2.1 <code>clear</code>	73
4.16.2.2 <code>dequeue</code>	73
4.16.2.3 <code>enqueue</code>	73
4.16.2.4 <code>is_empty</code>	74
4.16.2.5 <code>size</code>	74
4.16.3 Dokumentacja atrybutów składowych	74
4.16.3.1 <code>q</code>	74
4.17 Dokumentacja szablonu klasy <code>stack_array&lt;TYP&gt;</code>	74
4.17.1 Opis szczegółowy	76
4.17.2 Dokumentacja konstruktora i destruktora	76
4.17.2.1 <code>stack_array</code>	76
4.17.2.2 <code>stack_array</code>	76
4.17.3 Dokumentacja funkcji składowych	76
4.17.3.1 <code>clear</code>	76
4.17.3.2 <code>is_empty</code>	76
4.17.3.3 <code>pop</code>	77
4.17.3.4 <code>push</code>	77
4.17.3.5 <code>size</code>	78
4.17.4 Dokumentacja atrybutów składowych	78
4.17.4.1 <code>f</code>	78
4.17.4.2 <code>s</code>	78
4.17.4.3 <code>sp</code>	78
4.17.4.4 <code>st</code>	78
4.18 Dokumentacja szablonu klasy <code>stack_list&lt;TYP&gt;</code>	79
4.18.1 Opis szczegółowy	79
4.18.2 Dokumentacja funkcji składowych	79
4.18.2.1 <code>clear</code>	79
4.18.2.2 <code>is_empty</code>	80

4.18.2.3	pop	80
4.18.2.4	push	80
4.18.2.5	size	81
4.18.3	Dokumentacja atrybutów składowych	81
4.18.3.1	st	81
4.19	Dokumentacja klasy stos_list	81
4.19.1	Opis szczegółowy	82
4.19.2	Dokumentacja konstruktora i destruktora	82
4.19.2.1	stos_list	83
4.19.3	Dokumentacja funkcji składowych	83
4.19.3.1	przelicz	83
4.19.4	Dokumentacja atrybutów składowych	83
4.19.4.1	stos	83
4.20	Dokumentacja klasy stos_tablica	84
4.20.1	Opis szczegółowy	85
4.20.2	Dokumentacja konstruktora i destruktora	85
4.20.2.1	stos_tablica	85
4.20.3	Dokumentacja funkcji składowych	85
4.20.3.1	przelicz	85
4.20.4	Dokumentacja atrybutów składowych	86
4.20.4.1	stos	86
4.21	Dokumentacja klasy tab_aso	86
4.21.1	Opis szczegółowy	88
4.21.2	Dokumentacja konstruktora i destruktora	88
4.21.2.1	tab_aso	88
4.21.3	Dokumentacja funkcji składowych	88
4.21.3.1	przelicz	88
4.21.3.2	wczytaj_klucze	89
4.21.4	Dokumentacja atrybutów składowych	89
4.21.4.1	d	89
4.21.4.2	klucze	89
4.22	Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >	90
4.22.1	Opis szczegółowy	91
4.22.2	Dokumentacja konstruktora i destruktora	91

4.22.2.1	tablica_asocjacyjna	91
4.22.3	Dokumentacja funkcji składowych	92
4.22.3.1	czy_blokada	92
4.22.3.2	czy_pusta	92
4.22.3.3	dodaj	92
4.22.3.4	insert	93
4.22.3.5	odblokuj	93
4.22.3.6	pobierz	93
4.22.3.7	ustaw	93
4.22.3.8	usun	94
4.22.3.9	wez	94
4.22.3.10	wez_id	94
4.22.3.11	wstaw	95
4.22.3.12	wypisz	95
4.22.3.13	zablokuj	95
4.22.3.14	zlicz_elementy	95
4.22.3.15	znajdz	95
4.22.3.16	znajdz	96
4.22.4	Dokumentacja atrybutów składowych	96
4.22.4.1	blok	96
4.22.4.2	found	96
4.22.4.3	key	96
4.22.4.4	s	96
4.22.4.5	sp	97
4.22.4.6	value	97
4.23	Dokumentacja szablonu klasy wezel< TYP >	97
4.23.1	Opis szczegółowy	98
4.23.2	Dokumentacja konstruktora i destruktora	98
4.23.2.1	wezel	98
4.23.2.2	wezel	99
4.23.2.3	~wezel	99
4.23.3	Dokumentacja funkcji składowych	99
4.23.3.1	dodaj_syna	99
4.23.3.2	wez_klucz	99

4.23.3.3	wez_wart . . . . .	100
4.23.3.4	znajdz_nast . . . . .	100
4.23.4	Dokumentacja atrybutów składowych . . . . .	101
4.23.4.1	flag . . . . .	101
4.23.4.2	klucz . . . . .	101
4.23.4.3	lsyn . . . . .	101
4.23.4.4	ojciec . . . . .	101
4.23.4.5	psyn . . . . .	101
4.23.4.6	wart . . . . .	102
4.24	Dokumentacja klasy wierzcholek . . . . .	102
4.24.1	Opis szczegółowy . . . . .	102
4.24.2	Dokumentacja konstruktora i destruktora . . . . .	103
4.24.2.1	wierzcholek . . . . .	103
4.24.3	Dokumentacja przyjaciół i funkcji związkowych . . . . .	103
4.24.3.1	graf . . . . .	103
4.24.4	Dokumentacja atrybutów składowych . . . . .	103
4.24.4.1	id . . . . .	103
4.24.4.2	waga . . . . .	103
5	<b>Dokumentacja plików</b>	<b>105</b>
5.1	Dokumentacja pliku algorytm.cpp . . . . .	105
5.1.1	Opis szczegółowy . . . . .	105
5.2	Dokumentacja pliku algorytm.hh . . . . .	105
5.2.1	Opis szczegółowy . . . . .	107
5.3	Dokumentacja pliku drzewo.hh . . . . .	107
5.3.1	Opis szczegółowy . . . . .	108
5.3.2	Dokumentacja typów wyliczanych . . . . .	108
5.3.2.1	syn . . . . .	108
5.4	Dokumentacja pliku graf.cpp . . . . .	109
5.4.1	Dokumentacja zmiennych . . . . .	109
5.4.1.1	vec . . . . .	109
5.5	Dokumentacja pliku graf.hh . . . . .	110
5.5.1	Opis szczegółowy . . . . .	111
5.6	Dokumentacja pliku hashtab.hh . . . . .	111

---

5.6.1	Opis szczegółowy . . . . .	112
5.7	Dokumentacja pliku kolejka.hh . . . . .	112
5.7.1	Opis szczegółowy . . . . .	114
5.8	Dokumentacja pliku main.cpp . . . . .	114
5.8.1	Opis szczegółowy . . . . .	114
5.8.2	Dokumentacja funkcji . . . . .	114
5.8.2.1	main . . . . .	115
5.9	Dokumentacja pliku operacje.cpp . . . . .	115
5.10	Dokumentacja pliku operacje.hh . . . . .	116
5.10.1	Dokumentacja definicji . . . . .	117
5.10.1.1	ROZMIAR . . . . .	117
5.11	Dokumentacja pliku statystyki.cpp . . . . .	117
5.11.1	Dokumentacja funkcji . . . . .	117
5.11.1.1	odchylenie_standardowe . . . . .	118
5.11.1.2	srednia . . . . .	118
5.12	Dokumentacja pliku statystyki.hh . . . . .	119
5.12.1	Opis szczegółowy . . . . .	120
5.12.2	Dokumentacja funkcji . . . . .	120
5.12.2.1	odchylenie_standardowe . . . . .	120
5.12.2.2	srednia . . . . .	121
5.13	Dokumentacja pliku stos.hh . . . . .	121
5.13.1	Opis szczegółowy . . . . .	123
5.13.2	Dokumentacja typów wyliczanych . . . . .	123
5.13.2.1	flag . . . . .	123
5.14	Dokumentacja pliku str_operacje.cpp . . . . .	124
5.14.1	Dokumentacja funkcji . . . . .	124
5.14.1.1	operator< . . . . .	124
5.14.1.2	operator<= . . . . .	125
5.14.1.3	operator== . . . . .	125
5.14.1.4	operator> . . . . .	125
5.14.1.5	operator>= . . . . .	125
5.15	Dokumentacja pliku str_operacje.hh . . . . .	126
5.15.1	Dokumentacja funkcji . . . . .	127
5.15.1.1	operator< . . . . .	127

---

5.15.1.2 operator<= . . . . .	128
5.15.1.3 operator== . . . . .	128
5.15.1.4 operator> . . . . .	128
5.15.1.5 operator>= . . . . .	128
5.16 Dokumentacja pliku strona.dox . . . . .	128
5.17 Dokumentacja pliku tablica_asocjacyjna.hh . . . . .	128
5.17.1 Opis szczegółowy . . . . .	130

# Rozdział 1

## Indeks klas

### 1.1 Hierarchia klas

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

alorytm . . . . .	7
bst . . . . .	18
h_sort . . . . .	39
h_table . . . . .	41
kolejka_list . . . . .	49
kolejka_tablica . . . . .	52
m_sort . . . . .	54
mnozenie . . . . .	56
q_sort . . . . .	66
stos_list . . . . .	81
stos_tablica . . . . .	84
tab_aso . . . . .	86
drzewo< TYP > . . . . .	21
el_tab< TYP > . . . . .	25
graf . . . . .	27
hashtab< TYP > . . . . .	44
operacje . . . . .	59
queue_array< TYP > . . . . .	68
queue_list< TYP > . . . . .	72
stack_array< TYP > . . . . .	74
stack_list< TYP > . . . . .	79
tablica_asocacyjna< TYP > . . . . .	90
wezel< TYP > . . . . .	97
wierzcholek . . . . .	102



## 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, ktora ma za zadanie wczytac, przetworzyc i porownac dane z plikiem wzorcowym . . . . .	<a href="#">7</a>
<a href="#">bst</a>	Modeluje drzewo binarne przeznaczone do testowania szybkosci wyszukiwania . . . . .	<a href="#">18</a>
<a href="#">drzewo&lt; TYP &gt;</a>	Modeluje binarne drzewo przeszukiwan . . . . .	<a href="#">21</a>
<a href="#">el_tab&lt; TYP &gt;</a>	Pojedynczy element tablicy haszujacej . . . . .	<a href="#">25</a>
<a href="#">graf</a>	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 . . . . .	<a href="#">27</a>
<a href="#">h_sort</a>	Klasa reprezentuje dane poddane sortowaniu przez kopcowanie . . . . .	<a href="#">39</a>
<a href="#">h_table</a>	Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwania . . . . .	<a href="#">41</a>
<a href="#">hashtab&lt; TYP &gt;</a>	Modeluje tablice haszujaca w oparciu o kontener klasy <a href="#">el_tab</a> . . . . .	<a href="#">44</a>
<a href="#">kolejka_lista</a>	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	<a href="#">49</a>
<a href="#">kolejka_tablica</a>	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . . . . .	<a href="#">52</a>
<a href="#">m_sort</a>	Klasa reprezentuje dane poddane sortowaniu przez scalanie . . . . .	<a href="#">54</a>

<b>mnozenie</b>	
Modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2 . . . . .	56
<b>operacje</b>	
Klasa modeluje tablice z danymi i metody sluzace do operacji na niej	59
<b>q_sort</b>	
Klasa reprezentuje dane poddane sortowaniu szybkiemu . . . . .	66
<b>queue_array&lt; TYP &gt;</b>	
Modeluje kolejke w oparciu o tablice . . . . .	68
<b>queue_list&lt; TYP &gt;</b>	
Modeluje kolejke oparta na liscie STL . . . . .	72
<b>stack_array&lt; TYP &gt;</b>	
Modeluje stos w oparciu o tablice . . . . .	74
<b>stack_list&lt; TYP &gt;</b>	
Modeluje stos oparty na liscie STL . . . . .	79
<b>stos_lista</b>	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	81
<b>stos_tablica</b>	
Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	84
<b>tab_aso</b>	
Modeluje tablice asocjacyjna przeznaczona do testowania szybkosci wyszukiwania . . . . .	86
<b>tablica_asocjacyjna&lt; TYP &gt;</b>	
Klasa modeluje tablice asocjacyjna . . . . .	90
<b>wezel&lt; TYP &gt;</b>	
Modeluje pojedynczy wezel drzewa . . . . .	97
<b>wierzcholek</b>	
Klasa modeluje pojecie wierzcholka grafu. Nie jest to implementacja konieczna, aczkolwiek pozwala na dwojakie interpertowanie wierzcholka grafu, wedle zyczen uzytkownika . . . . .	102

## 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>	105
<a href="#">algorytm.hh</a>	Definicja klas wykonujacych operacje na zestawie danych wejsciowych . . . . .	105
<a href="#">drzewo.hh</a>	. . . . .	107
<a href="#">graf.cpp</a>	. . . . .	109
<a href="#">graf.hh</a>	. . . . .	110
<a href="#">hashtab.hh</a>	. . . . .	111
<a href="#">kolejka.hh</a>	Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiekszajacej swój rozmiar b. powiekszajacej swój rozmiar dwukrotnie, gdy kolejka sie przepelni . . . . .	112
<a href="#">main.cpp</a>	Plik glowny . . . . .	114
<a href="#">operacje.cpp</a>	. . . . .	115
<a href="#">operacje.hh</a>	. . . . .	116
<a href="#">statystyki.cpp</a>	. . . . .	117
<a href="#">statystyki.hh</a>	Plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk . . . . .	119
<a href="#">stos.hh</a>	Plik zawiera definicje klasy Stos Zaimplementowana na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiekszajacej swój rozmiar b. powiekszajacej swój rozmiar dwukrotnie, gdy stos sie przepelni . . . . .	121
<a href="#">str_operacje.cpp</a>	. . . . .	124
<a href="#">str_operacje.hh</a>	. . . . .	126

[tablica\\_asocjacyjna.hh](#) . . . . . 128

## Rozdział 4

# Dokumentacja klas

### 4.1 Dokumentacja klasy algorytm

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

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

Diagram dziedziczenia dla algorytm

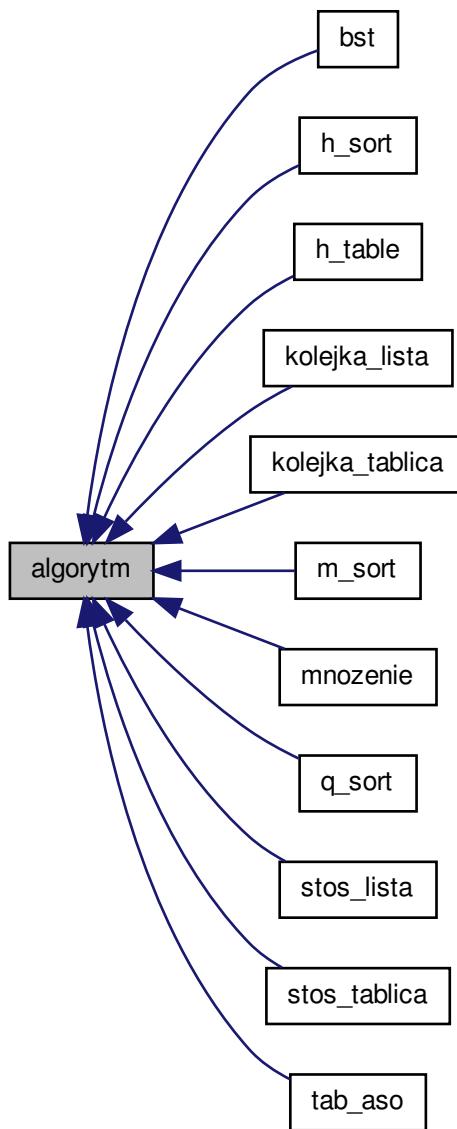
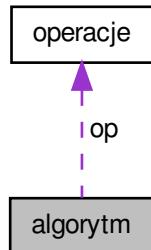


Diagram współpracy dla algorytm:



## Metody publiczne

- **algorytm** (ifstream &plik1, ifstream &plik2, int N, int M)  
*konstruktor kopujacy - przekazuje informacje o nazwach plikow, ktore zapisywane sa do pol klasy*
- void **wykonaj** (ofstream &out)  
*funkcja dokonuje operacji na pliku wejsciowym, wywoluje metody odpowiedzialne za pomiar czasu oraz za porownanie wyniku operacji z plikiem wzorcowym*
- bool **wczytaj** (ifstream &plik)  
*Metoda wczytuje plik wejsciowy do tablicy dane oraz do obiektu op klasy operacje.*
- void **set\_N** (int wart)  
*metoda ustawia wartosc n*
- bool **wczytaj\_wzor** (ifstream &plik)  
*Metoda wczytuje plik wzorcowy do tablicy dane\_wz.*
- virtual float **przelicz** ()  
*Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.*
- bool **porownaj** ()  
*porownuje przetworzony dane z danymi wzorcowymi*
- int **ile\_danych** ()
- float \* **jaki\_czas** ()
- void **wlacz\_zegar** ()  
*Metoda wlacza pomiar czasu poprzez wlaczenie funkcji gettimeofday i przechowanie czasu w zmiennej start.*
- void **wylacz\_zegar** ()  
*Metoda wyacza pomiar czasu poprzez wlaczenie funkcji gettimeofday i przechowanie czasu w zmiennej end.*

- void **zapisz\_do\_csv** (ofstream &out)
 

*Metoda zapisuje tablice czas do pliku wyjscie.csv.*
- void **zapisz\_do\_gnuplot** (ofstream &out, float sr, float od)
 

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

### Atrybuty publiczne

- float \* **czas**

*zawiera wyniki dzialania algorytmu*

### Atrybuty chronione

- float \* **dane**

*Tablica liczb wczytana z pliku.*
- float \* **dane\_wz**

*tablica liczb zawartych w pliku wzorcowym*
- int **n**

*ilosc danych w pliku*
- int **m**

*ilosc powtorzen*
- **operacje op**

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

#### 4.1.1 Opis szczegółowy

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

Definicja w linii 36 pliku algorytm.hh.

#### 4.1.2 Dokumentacja konstruktora i destruktora

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

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

#### 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 79 pliku algorytm.hh.

### 4.1.3 Dokumentacja funkcji składowych

#### 4.1.3.1 int algorytm::ile\_danych( )

Zwroca

ilosc liczb wejsciowych

Definicja w linii 31 pliku algorytm.cpp.

#### 4.1.3.2 float \* algorytm::jaki\_czas( )

Zwroca

tablica czas z danymi pomiarowymi czasu wykonywania algorytmu

Definicja w linii 34 pliku algorytm.cpp.

#### 4.1.3.3 bool algorytm::porownaj( )

porownuje przetworzony dane z danymi wzorcowymi

Zwroca

true - gdy pliki zgodne false - w przeciwnym przypadku

Definicja w linii 99 pliku algorytm.cpp.

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

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana w [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 93 pliku algorytm.hh.

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

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

##### Parametry

in	plik	- strumien pliku wejsciowego
----	------	------------------------------

Definicja w linii 10 pliku algorytm.cpp.

#### 4.1.3.7 bool algorytm::wczytaj\_wzor( ifstream & plik )

Metoda wczytuje plik wzorcowy do tablicy dane\_wz.

##### Parametry

in	plik	- strumien pliku wejsciowego
----	------	------------------------------

Definicja w linii 21 pliku algorytm.cpp.

#### 4.1.3.8 void algorytm::wlacz\_zegar( )

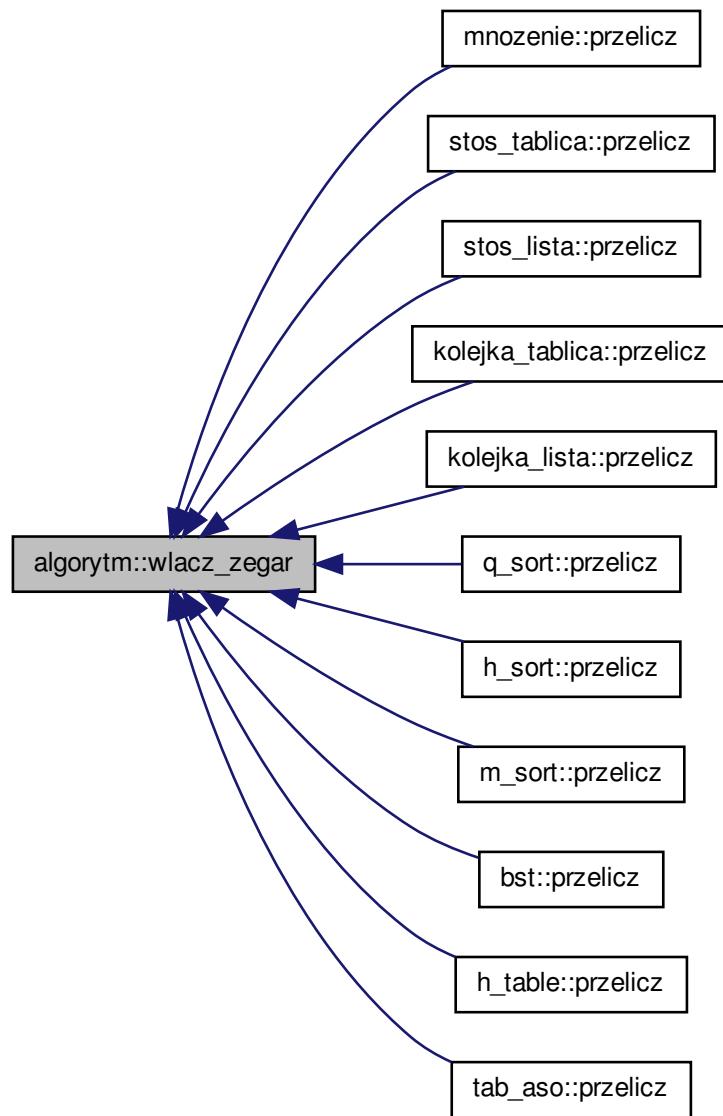
Metoda wlacza pomiar czasu poprzez wlaczenie funkcji gettimeofday i przechowanie czasu w zmiennej start.

Zwrota

start - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

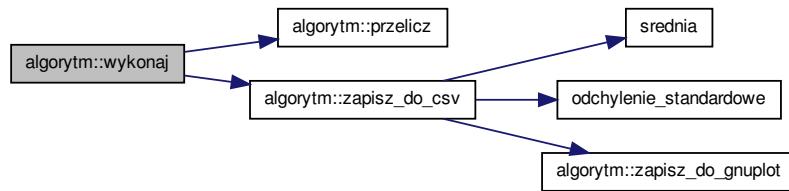


#### 4.1.3.9 void algorytm::wykonaj ( ostream & out )

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:



#### 4.1.3.10 void algorytm::wylacz\_zegar ( )

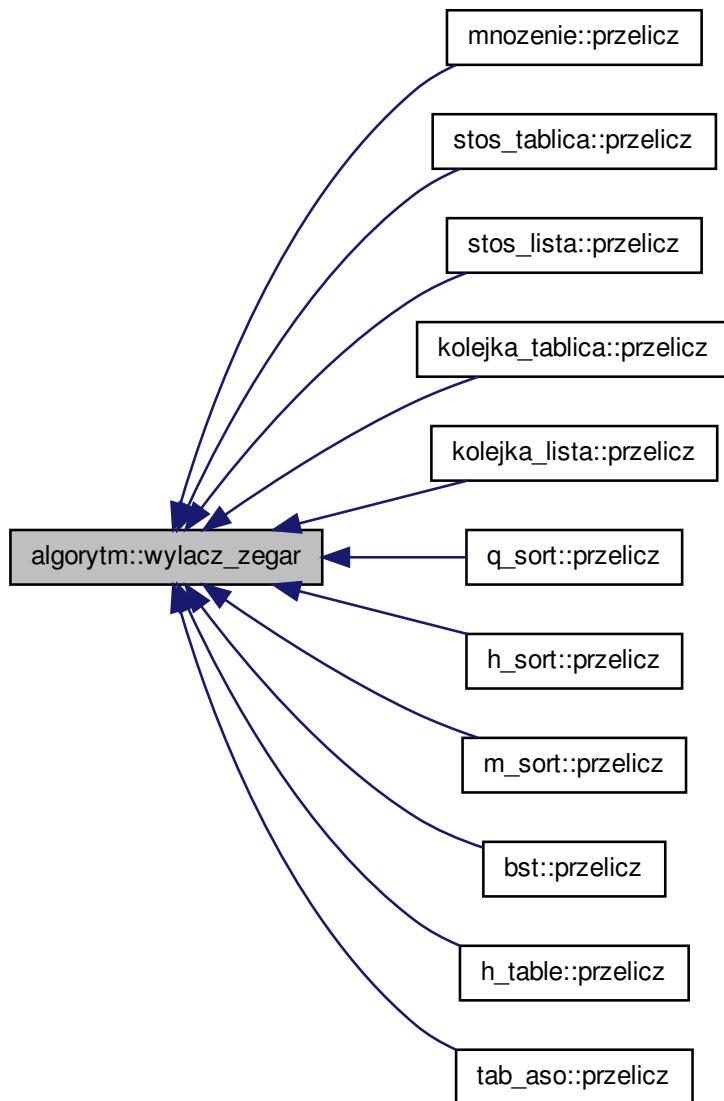
Metoda wyacza pomiar czasu poprzez włączenie funkcji gettimeofday i przechowywanie czasu w zmiennej end.

Zwroca

end - zmienna pamiętająca czas poprzedzający wykonanie algorytmu

Definicja w linii 49 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

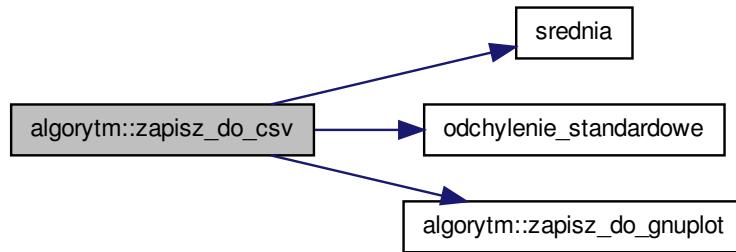


#### 4.1.3.11 void algorytm::zapisz\_do\_csv ( ostream & out )

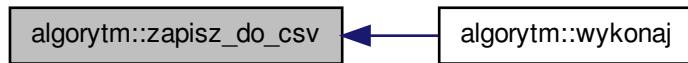
Metoda zapisuje tablice `czas` do pliku `wyjscie.csv`.

Definicja w linii 62 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

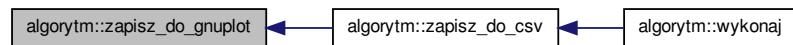


#### 4.1.3.12 void algorytm::zapisz\_do\_gnuplot ( ostream & out, float sr, float od )

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 69 pliku algorytm.hh.

##### 4.1.4.2 double algorytm::czas1 [protected]

Definicja w linii 64 pliku algorytm.hh.

##### 4.1.4.3 double algorytm::czas2 [protected]

Definicja w linii 64 pliku algorytm.hh.

##### 4.1.4.4 float\* algorytm::dane [protected]

Tablica liczb wczytana z pliku.

Definicja w linii 44 pliku algorytm.hh.

##### 4.1.4.5 float\* algorytm::dane\_wz [protected]

tablica liczb zawartych w pliku wzorcowym

Definicja w linii 49 pliku algorytm.hh.

##### 4.1.4.6 int algorytm::m [protected]

ilosc powtorzen

Definicja w linii 59 pliku algorytm.hh.

##### 4.1.4.7 int algorytm::n [protected]

ilosc danych w pliku

Definicja w linii 55 pliku algorytm.hh.

##### 4.1.4.8 operacje algorytm::op [protected]

klasa zawierajaca tablice i metody do operacji na niej

Definicja w linii 63 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [algorytm.hh](#)

- [algorytm.cpp](#)

## 4.2 Dokumentacja klasy bst

Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla bst

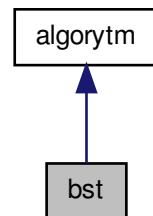
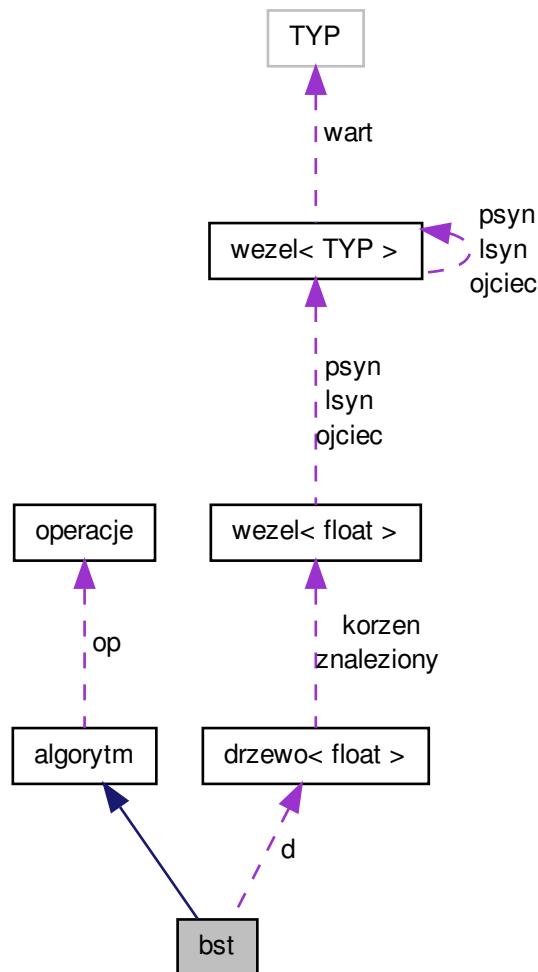


Diagram współpracy dla bst:



### Metody publiczne

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

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorymem.*

## Atrybuty prywatne

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

### 4.2.1 Opis szczegółowy

Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.

Definicja w linii 226 pliku algorytm.hh.

### 4.2.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 231 pliku algorytm.hh.

#### 4.2.2.2 `bst::~bst ( )` [inline]

Definicja w linii 239 pliku algorytm.hh.

### 4.2.3 Dokumentacja funkcji składowych

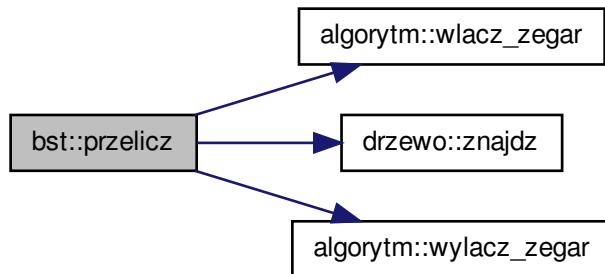
#### 4.2.3.1 `float bst::przelicz ( )` [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 204 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.2.3.2 void bst::wczytaj\_klucze ( ifstream & plik )

Definicja w linii 199 pliku algorytm.cpp.

#### 4.2.4 Dokumentacja atrybutów składowych

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

Definicja w linii 227 pliku algorytm.hh.

##### 4.2.4.2 string\* bst::klucze [private]

Definicja w linii 228 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.3 Dokumentacja szablonu klasy drzewo< TYP >

modeluje binarne drzewo przeszukiwan

```
#include <drzewo.hh>
```

### Metody publiczne

- **drzewo ()**  
*konstruktor bezparametryczny*
- **drzewo (string k, TYP v)**  
*konstruktor parametryczny - przypisuje korzeniowi klucz i wartosc*
- **void dodaj\_wezel (wezel< TYP > \*W)**  
*dodaje wezel do drzewa*
- **void dodaj (string k, TYP v)**  
*dodaje wezel do drzewa*
- **bool znajdz (string k)**  
*szuka wezla o zadanym kluczu*
- **bool szukaj (string k, wezel< TYP > \*w)**  
*sprawdza, czy w danym wezle znajduje sie szukany klucz*
- **void usun (string k)**  
*usuwa element o kluczu k, jezeli zostanie on znaleizony*
- **void czysc (wezel< TYP > \*w)**  
*rekursywne czyszczenie wezla*
- **void wyczysc ()**  
*czysci całe drzewo*

### Atrybuty publiczne

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

#### 4.3.1 Opis szczegółowy

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

modeluje binarne drzewo przeszukiwan

Definicja w linii 70 pliku drzewo.hh.

#### 4.3.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 77 pliku drzewo.hh.

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

konstruktor parametryczny - przypisuje korzeniowi klucz i wartosc

Definicja w linii 79 pliku drzewo.hh.

### 4.3.3 Dokumentacja funkcji składowych

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

rekursywne czyszczenie wezla

#### Parametry

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

Definicja w linii 180 pliku drzewo.hh.

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

dodaje wezel do drzewa

#### Parametry

in	k	- klucz wezla
in	v	= wartosc wezla

Definicja w linii 91 pliku drzewo.hh.

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

dodaje wezel do drzewa

#### Parametry

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

Definicja w linii 83 pliku drzewo.hh.

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

sprawdza, czy w danym wezle znajduje sie szukany klucz

**Parametry**

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

**Zwraca**

true, gdy znaleziono, false w przeciwnym przypadku

Definicja w linii 109 pliku drzewo.hh.

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

usuwa element o kluczu *k*, jeżeli zostanie on znaleziony

**Parametry**

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

Definicja w linii 124 pliku drzewo.hh.

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

czyszczy całe drzewo

Definicja w linii 188 pliku drzewo.hh.

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

szukana wezla o zadanym kluczu

**Parametry**

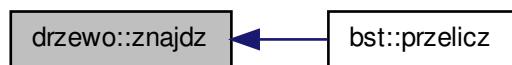
in	<i>k</i>	- klucz
----	----------	---------

**Zwraca**

true, gdy znaleziono, w przeciwnym wypadku zwraca false

Definicja w linii 100 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



#### 4.3.4 Dokumentacja atrybutów składowych

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

korzen drzewa

Definicja w linii 73 pliku drzewo.hh.

##### 4.3.4.2 template<typename TYP> wezel<TYP>\* drzewo< TYP >::zalezony

zalezony wezel w drzewie

Definicja w linii 75 pliku drzewo.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

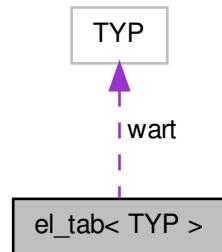
- drzewo.hh

## 4.4 Dokumentacja szablonu klasy el\_tab< TYP >

pojedynczy element tablicy haszujacej

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

Diagram współpracy dla el\_tab< TYP >:



### Metody publiczne

- `el_tab ()`
- `~el_tab ()`

### Atrybuty publiczne

- string `klucz`  
*identyfikator*
- TYP `wart`  
*wartosc pola*
- bool `zajety`  
*flaga informujaca, czy pole jest zajete*

#### 4.4.1 Opis szczegółowy

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

pojedynczy element tablicy haszujacej

Definicja w linii 11 pliku hashtab.hh.

#### 4.4.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 26 pliku hashtab.hh.

4.4.2.2 template<typename TYP> el\_tab<TYP>::~el\_tab( ) [inline]

Definicja w linii 27 pliku hashtab.hh.

#### 4.4.3 Dokumentacja atrybutów składowych

4.4.3.1 template<typename TYP> string el\_tab<TYP>::klucz

identyfikator

Definicja w linii 16 pliku hashtab.hh.

4.4.3.2 template<typename TYP> TYP el\_tab<TYP>::wart

wartosc pola

Definicja w linii 21 pliku hashtab.hh.

4.4.3.3 template<typename TYP> bool el\_tab<TYP>::zajety

flaga informujaca, czy pole jest zajete

Definicja w linii 25 pliku hashtab.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

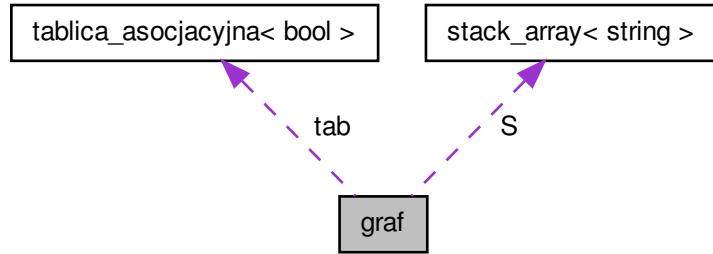
- [hashtab.hh](#)

## 4.5 Dokumentacja klasy graf

Klasa modeluje pojęcie grafu w oparciu o liste incydencji. Operacje na grafie możliwe są na dwa sposoby \n 1. Podając wierzchołek grafu jako parametr metody \n 2. Podając id wierzchołka jako parametr metody.

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

Diagram współpracy dla graf:



## Metody publiczne

- **graf ()**  
*Konstruktor nieparametryczny - ustala sposob zarzadzania pamiecia na stosie.*
- void **dodaj\_wierzcholek ()**  
*Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.*
- bool **czy\_sasiad (unsigned int id1, unsigned int id2)**  
*Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez identyfikatory.*
- bool **czy\_sasiad (wierzcholek w1, wierzcholek w2)**  
*Sprawdza czy podane wierzcholki sa polaczone krawedzia - odwolanie poprzez obiekt klasy wierzcholek.*
- void **sasiedztwo (wierzcholek w)**  
*wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez obiekt typu wierzcholek*
- void **dodaj\_wierzcholek (wierzcholek w)**  
*Dodaje wierzcholek do wezla, wierzcholkom przypisuje sie identyfikatory bedace kolejnymi liczbami naturalnymi. Dodany wierzcholek nie posiada krawedzi incydentnych.*
- void **dodaj\_krawedz (unsigned int id1, unsigned int id2, unsigned int waga)**  
*Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez identyfikatory wierzcholkow.*
- void **dodaj\_krawedz (wierzcholek w1, wierzcholek w2, unsigned int waga)**  
*Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwolanie poprzez obiekty typu wierzcholek.*
- void **sasiedztwo (unsigned int id)**  
*wypisuje wszystkie wierzcholki polaczone krawedzia z podanym wierzcholkiem - odwolanie poprzez identyfikator wierzcholka*

- void **usun\_krawedz** (unsigned int id1, unsigned int id2)  
*usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez identyfikatory wierzcholkow*
- void **usun\_krawedz** (**wierzcholek** w1, **wierzcholek** w2)  
*usuwa krawedz spomiedzy 2 wierzcholkow - odwolanie poprzez obiekt typu wierzcholek*
- void **usun\_wierzcholek** (unsigned int id)  
*usuwa podany wierzcholek, a scislej, ustawia flagę w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika*
- void **usun\_wierzcholek** (**wierzcholek** w)  
*usuwa podany wierzcholek, a scislej, ustawia flagę w strukturze tablicy asocjacyjnej, przez co dany wierzcholek jest niewidoczny dla uzytkownika*
- void **wypisz\_liste** ()  
*wypisuje pełna liste incydencji grafu*
- void **wyczysc** ()  
*usuwa wszystkie obiekty z listy incydencji grafu*
- int **przeszukaj\_wezel** (int id)  
*Jeżeli wezel nei byl odwiedzony, odklada na stos wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.*
- void **dfs** ()  
*Metoda przeszukuje wgłąb cały graf.*

## Atrybuty prywatne

- **stack\_array< string > S**  
*stos sluzacy do przeszukiwania grafu*
- **tablica\_asocjacyjna< bool > tab**  
*struktura sluzaca do przechowywania grafu, zawiera informacje, czy wierzcholek byl odwiedzony*
- **vector< tablica\_asocjacyjna < int > > lista\_incydencji**  
*lista incydencji grafu*

### 4.5.1 Opis szczegółowy

Klasa modeluje pojęcie grafu w oparciu o liste incydencji, Operacje na grafie możliwe są na dwa sposoby \n 1. Podając wierzchołek grafu jako parametr metody \n 2. Podając id wierzchołka jako parametr metody.

Definicja w linii 36 pliku graf.hh.

### 4.5.2 Dokumentacja konstruktora i destruktora

#### 4.5.2.1 **graf::graf( ) [inline]**

Konstruktor nieparametryczny - ustala sposób zarządzania pamięcią na stosie.

Definicja w linii 46 pliku graf.hh.

### 4.5.3 Dokumentacja funkcji składowych

#### 4.5.3.1 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 są sąsiadami, false - gdy nie są

Definicja w linii 61 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



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

Sprawdza czy podane wierzchołki są połączone krawędzią - odwołanie poprzez obiekty klasy wierzcholek.

##### Parametry

in	<i>w1</i>	- pierwszy wierzchołek
in	<i>w2</i>	- drugi wierzchołek

Zwraca

true - gdy sa sasiadami, false - gdy nie sa

Definicja w linii 68 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:

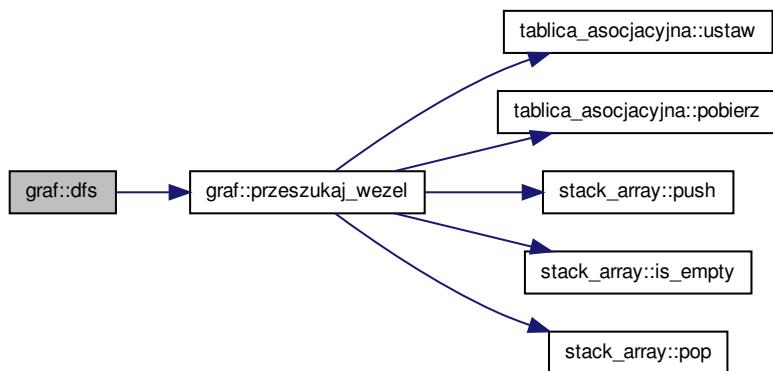


#### 4.5.3.3 void graf::dfs( )

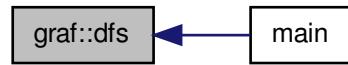
Metoda przeszukuje wgłąb cały graf.

Definicja w linii 118 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



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

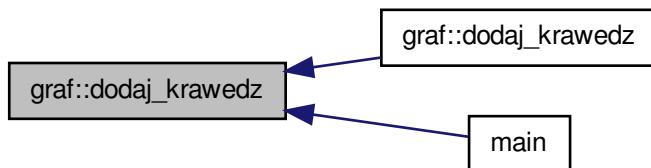
Dodaje krawedz o wadze waga pomiedzy 2 wezly - odwołanie poprzez identyfikatory wierzcholkow.

##### Parametry

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

Definicja w linii 28 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



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

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

**Parametry**

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

Definicja w linii 24 pliku graf.cpp.

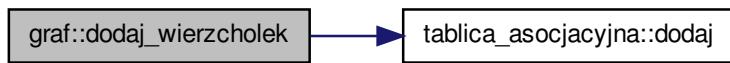
Oto graf wywołań dla tej funkcji:

**4.5.3.6 void graf::dodaj\_wierzcholek( )**

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

Definicja w linii 6 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.5.3.7 void graf::dodaj\_wierzcholek ( wierzcholek *w* )

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

Definicja w linii 14 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.5.3.8 int graf::przeszukaj\_wezel ( int *id* )

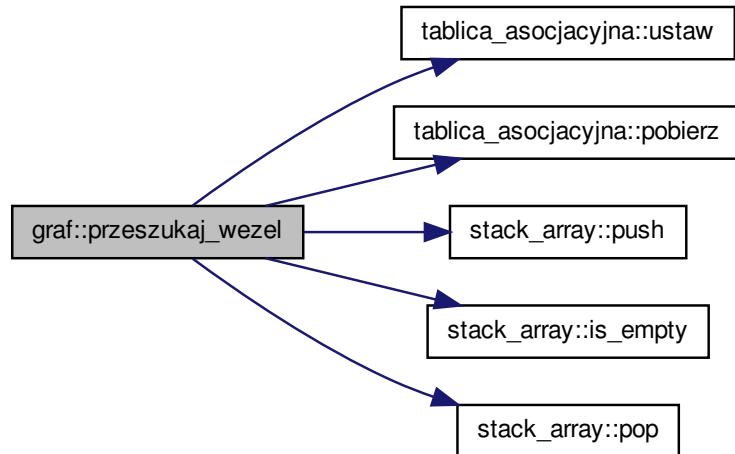
Jeżeli wezel nei byl odwiedzony, odklada na stos wszystkie jego nieodwiedzone nastepniki i rekurencyjnie je przeszukuje.

##### Parametry

in	<i>id</i>	- id wezla, ktory ma byc przeszukany
----	-----------	--------------------------------------

Definicja w linii 95 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.5.3.9 void `graf::sasiedztwo ( wierzcholek w )`

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

##### Parametry

in	w	- zadany wierzchołek
----	---	----------------------

Definicja w linii 45 pliku `graf.cpp`.

#### 4.5.3.10 void graf::sasiedztwo ( unsigned int *id* )

wypisuje wszystkie wierzchołki połączone krawędzią z podanym wierzchołkiem - odwołanie poprzez identyfikator wierzchołka

##### Parametry

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

Definicja w linii 36 pliku graf.cpp.

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

usuwa krawędź spomiędzy 2 wierzchołków - odwołanie poprzez identyfikatory wierzchołków

##### Parametry

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

Definicja w linii 73 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



#### 4.5.3.12 void graf::usun\_krawedz ( wierzcholek *w1*, wierzcholek *w2* )

usuwa krawędź spomiędzy 2 wierzchołków - odwołanie poprzez obiekt typu wierzchołek

##### Parametry

in	<i>w1</i>	- pierwszy wierzchołek
in	<i>w2</i>	- drugi wierzchołek

Definicja w linii 82 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.5.3.13 void graf::usun\_wierzcholek ( unsigned int id )

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

##### Parametry

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

Definicja w linii 85 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



#### 4.5.3.14 void graf::usun\_wierzcholek ( wierzcholek w )

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

##### Parametry

in	<i>w</i>	- wierzcholek który trzeba usunac
----	----------	-----------------------------------

Definicja w linii 91 pliku graf.cpp.

Oto graf wywołań dla tej funkcji:



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

usuwa wszystkie obiekty z listy incydencji grafu

Definicja w linii 110 pliku graf.hh.

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

wypisuje pełną listę incydencji grafu

Definicja w linii 49 pliku graf.cpp.

Oto graf wywoływań tej funkcji:



#### 4.5.4 Dokumentacja atrybutów składowych

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

lista incydencji grafu

Definicja w linii 43 pliku graf.hh.

##### 4.5.4.2 stack\_array<string> graf::S [private]

stos sluzacy do przeszukiwania grafu

Definicja w linii 39 pliku graf.hh.

#### 4.5.4.3 tablica\_asocjacyjna<bool> graf::tab [private]

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

Definicja w linii 41 pliku graf.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.6 Dokumentacja klasy h\_sort

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

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

Diagram dziedziczenia dla h\_sort

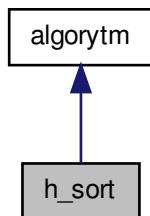
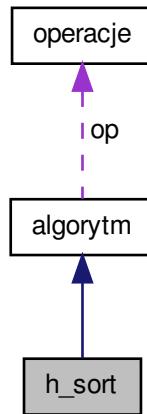


Diagram współpracy dla h\_sort:



## Metody publiczne

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

### 4.6.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez kopcowanie  
Definicja w linii 207 pliku algorytm.hh.

### 4.6.2 Dokumentacja konstruktora i destruktora

#### 4.6.2.1 **h\_sort::h\_sort** ( ifstream & plik1, ifstream & plik2, int N, int M ) [inline]

konstruktor klasy

Definicja w linii 210 pliku algorytm.hh.

### 4.6.3 Dokumentacja funkcji składowych

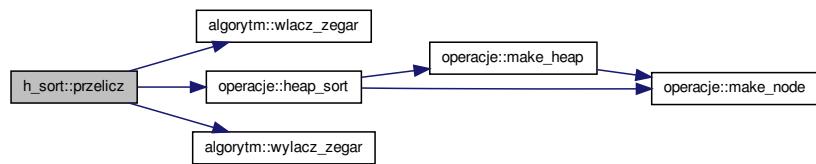
#### 4.6.3.1 float h\_sort::przelicz( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 180 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.7 Dokumentacja klasy h\_table

Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwania.

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

Diagram dziedziczenia dla h\_table

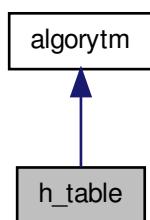
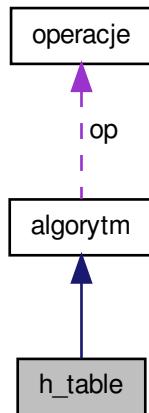


Diagram współpracy dla h\_table:



### Metody publiczne

- void `wczytaj_klucze` (ifstream &plik)  
*wczytywanie kluczy*
- `h_table` (ifstream &plik1, ifstream &plik2, ifstream &plik3, int N, int M)  
*konstruktor*
- float `przelicz` ()  
*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorymem.*

### Atrybuty prywatne

- string \* `klucze`  
*tablica kluczy*

#### 4.7.1 Opis szczegółowy

Modeluje tablice haszujaca przeznaczona do testowania szybkości wyszukiwania.

Definicja w linii 246 pliku algorytm.hh.

### 4.7.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Definicja w linii 255 pliku algorytm.hh.

### 4.7.3 Dokumentacja funkcji składowych

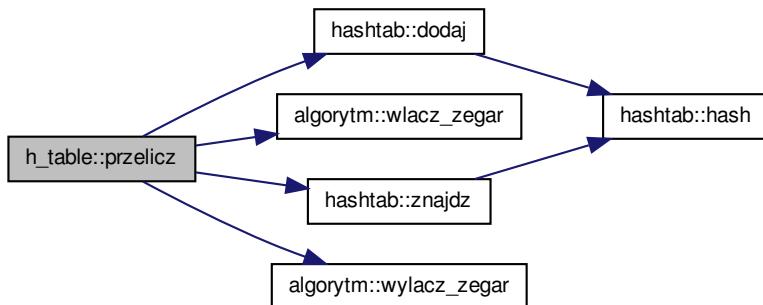
4.7.3.1 `float h_table::przelicz ( ) [virtual]`

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:



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

wczytywanie kluczy

#### Parametry

in	<code>plik</code> - strumien z kluczami uzytymi podczas testow
----	--

Definicja w linii 213 pliku algorytm.cpp.

#### 4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 string\* h\_table::klucze [private]

tablica kluczy

Definicja w linii 248 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

### 4.8 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 hasujacej*
- [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\*](#)

### 4.8.1 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`.

### 4.8.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 43 pliku `hashtab.hh`.

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

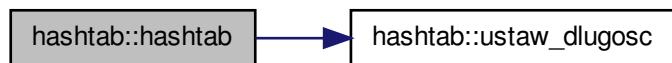
konstruktor parametryczny

#### Parametry

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

Definicja w linii 47 pliku `hashtab.hh`.

Oto graf wywołań dla tej funkcji:



### 4.8.3 Dokumentacja funkcji składowych

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

metoda dodaje element do tablicy haszujacej

Definicja w linii 59 pliku `hashtab.hh`.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



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

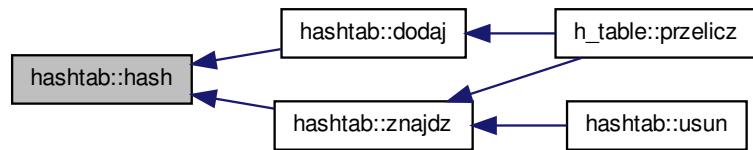
funkcja haszujaca

Zwraca

h - liczba, ktora po kompresji bedzie indeksem danego elementu

Definicja w linii 51 pliku hashtable.hh.

Oto graf wywoływań tej funkcji:

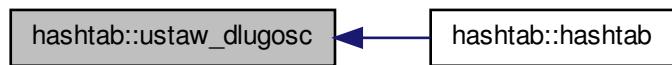


4.8.3.3 template<typename TYP> void hashtable< TYP >::ustaw\_dlugosc ( int d )  
[inline]

ustawia dlugosc tablicy

Definicja w linii 41 pliku hashtable.hh.

Oto graf wywoływań tej funkcji:



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

usuwa element jesli znajduje sie w tablicy

Parametry

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

Definicja w linii 89 pliku hashtable.hh.

Oto graf wywołań dla tej funkcji:



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

Definicja w linii 93 pliku hashtable.hh.

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

metoda szuka zadanego elementu w oparciu o klucz

**Parametry**

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

**Zwraca**

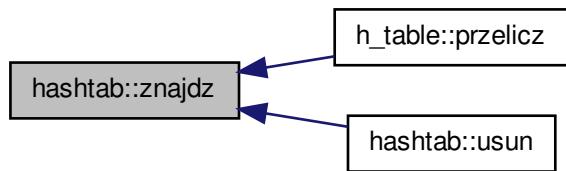
znaleziony element

Definicja w linii 74 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.8.4 Dokumentacja atrybutów składowych

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

dlugosc tablicy

Definicja w linii 36 pliku hashtab.hh.

```
4.8.4.2 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](#)

## 4.9 Dokumentacja klasy kolejka\_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka\_lista

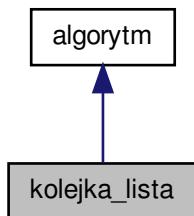
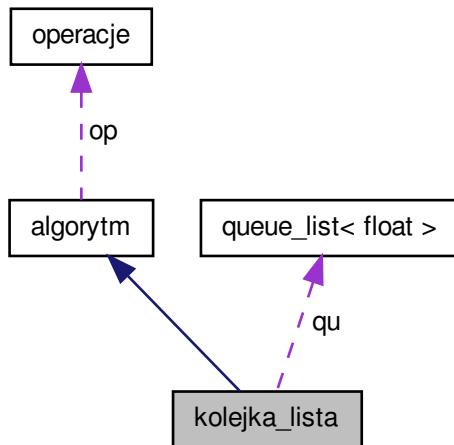


Diagram współpracy dla kolejka\_listy:



### Metody publiczne

- [kolejka\\_listy](#) (ifstream &plik1, ifstream &plik2, int N, int M)
- float [przelicz](#) ()

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.*

### Atrybuty prywatne

- [queue\\_list< float >](#) **qu**

#### 4.9.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 190 pliku algorytm.hh.

#### 4.9.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 193 pliku algorytm.hh.

#### 4.9.3 Dokumentacja funkcji składowych

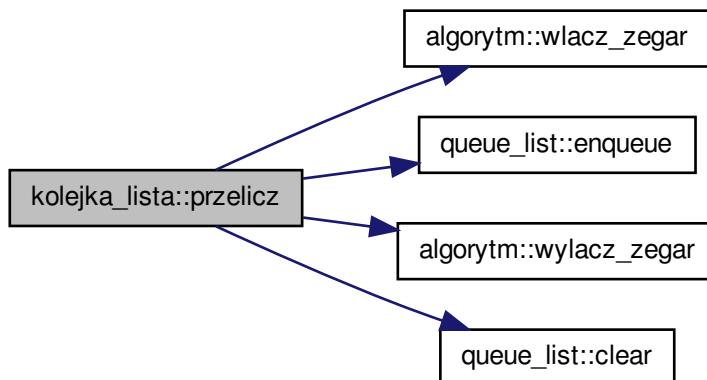
4.9.3.1 `float kolejka_list::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 160 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 `queue_list<float> kolejka_list::qu [private]`

Definicja w linii 191 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.10 Dokumentacja klasy kolejka\_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka\_tablica

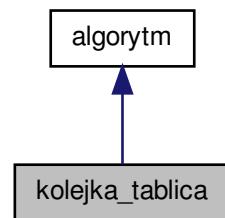
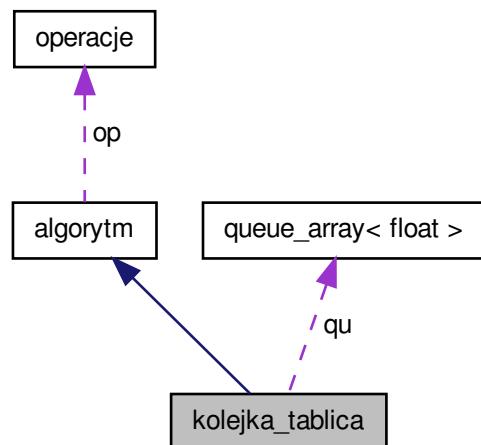


Diagram współpracy dla kolejka\_tablica:



### Metody publiczne

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

*konstruktor - ustawia flagę w zadany stan*

- `float przelicz ()`

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.*

### Atrybuty prywatne

- `queue_array< float > qu`

#### 4.10.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 178 pliku algorytm.hh.

#### 4.10.2 Dokumentacja konstruktora i destruktora

##### 4.10.2.1 `kolejka_tablica::kolejka_tablica ( ifstream & plik1, ifstream & plik2, int N, int M, flag F ) [inline]`

konstruktor - ustawia flagę w zadany stan

Definicja w linii 184 pliku algorytm.hh.

#### 4.10.3 Dokumentacja funkcji składowych

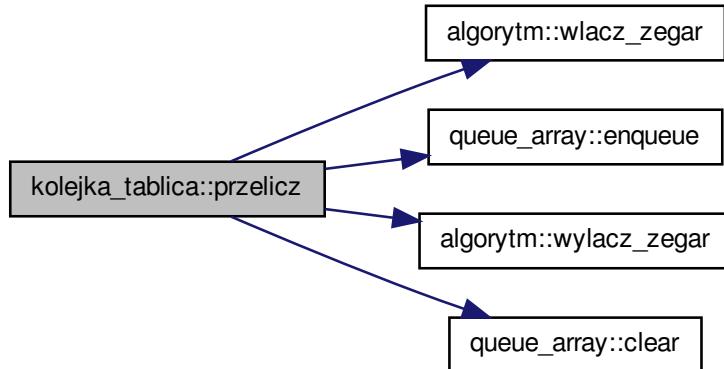
##### 4.10.3.1 `float kolejka_tablica::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 148 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.10.4 Dokumentacja atrybutów składowych

##### 4.10.4.1 `queue_array<float> kolejka_tablica::qu` [private]

Definicja w linii 179 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

#### 4.11 Dokumentacja klasy `m_sort`

klasa reprezentuje dane poddane sortowaniu przez scalanie

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

Diagram dziedziczenia dla m\_sort

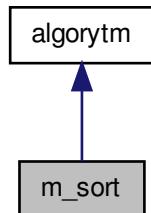
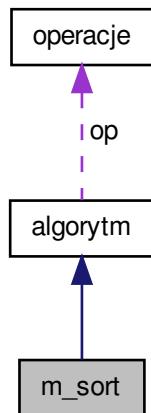


Diagram współpracy dla m\_sort:



## Metody publiczne

- **m\_sort** (ifstream &plik1, ifstream &plik2, int N, int M)  
*konstruktor*
- float **przelicz ()**  
*metoda dokonujaca sortowania danych*

#### 4.11.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez scalanie

Definicja w linii 216 pliku algorytm.hh.

#### 4.11.2 Dokumentacja konstruktora i destruktorów

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

konstruktor

Definicja w linii 219 pliku algorytm.hh.

#### 4.11.3 Dokumentacja funkcji składowych

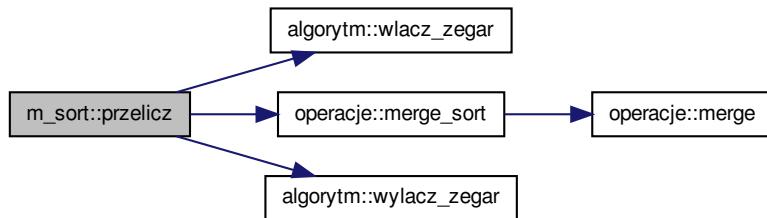
##### 4.11.3.1 `float m_sort::przelicz( ) [virtual]`

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](#)

#### 4.12 Dokumentacja klasy mnozenie

modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2

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

Diagram dziedziczenia dla mnozenie

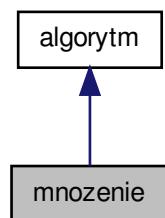
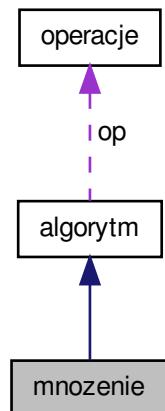


Diagram współpracy dla mnozenie:



### Metody publiczne

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

*wykonuje zalożony algorytm mnożenia elementów tablicy przez 2*

#### 4.12.1 Opis szczegółowy

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2  
 Definicja w linii 138 pliku algorytm.hh.

#### 4.12.2 Dokumentacja konstruktora i destruktora

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

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

##### Parametry

in	<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 147 pliku algorytm.hh.

#### 4.12.3 Dokumentacja funkcji składowych

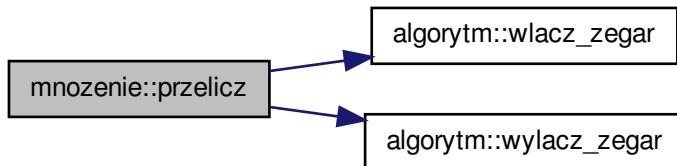
##### 4.12.3.1 **float mnozenie::przelicz ( ) [virtual]**

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

## 4.13 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 sortowania 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 scalia 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*

### 4.13.1 Opis szczegółowy

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

Definicja w linii 11 pliku operacje.hh.

### 4.13.2 Dokumentacja konstruktora i destruktora

#### 4.13.2.1 **operacje::operacje( )**

konstruktor bezparametryczny

#### 4.13.2.2 **operacje::operacje( int N ) [inline]**

konstruktor parametryczny - alokuje pamiec w dynamicznej tablicy tab

##### Parametry

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

Definicja w linii 28 pliku operacje.hh.

### 4.13.3 Dokumentacja funkcji składowych

#### 4.13.3.1 **void operacje::dodaj\_element( float e )**

metoda dodaje element do tablicy, alokujac dodatkowa pamiec

##### Parametry

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

Definicja w linii 27 pliku operacje.cpp.

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

metoda dodaje elementy do tablicy

**Parametry**

in	<i>tab2</i>	- tablica, ktorą należy dodać
in	<i>rozm</i>	- rozmiar tablicy tab2

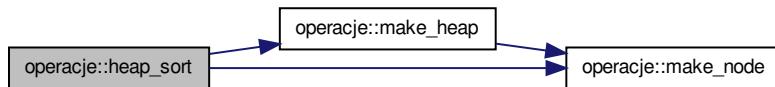
Definicja w linii 46 pliku operacje.cpp.

**4.13.3.3 void operacje::heap\_sort( )**

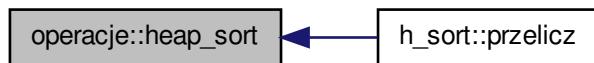
Metoda dokonuje sortowania po uprzednim utworzeniu kopca.

Definicja w linii 116 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

**4.13.3.4 void operacje::make\_heap( )**

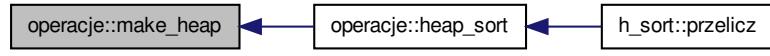
Metoda tworzy kopiec binarny.

Definicja w linii 110 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



#### 4.13.3.5 void operacje::make\_node ( int rozmiar, int i )

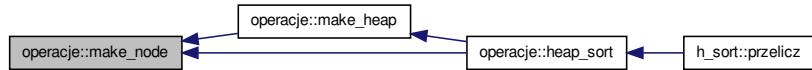
Metoda tworzy wezel drzewa, przypisujac mu 2 synow, ustawiajac ich w odpowiedniej kolejnosci (ojciec ma najwieksza wartosc)

##### 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:



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

Metoda scalą dwie części tablicy, jednocześnie ją porządkując.

## Parametry

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

Definicja w linii 130 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



## 4.13.3.7 void operacje::merge\_sort ( int poczatek, int koniec )

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

## Parametry

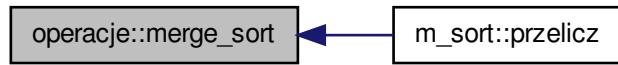
in	<i>poczatek</i>	- pierwszy indeks tablicy
in	<i>koniec</i>	- ostatni indeks tablicy

Definicja w linii 166 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



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

metoda odwraca wszystkie elementy tablicy

Definicja w linii 12 pliku operacje.cpp.

#### 4.13.3.9 void operacje::operator=( float \* tab1 )

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

##### Parametry

in	<code>tab1</code>	- tablica, której zawartość przypisujemy
----	-------------------	--

Definicja w linii 63 pliku operacje.cpp.

#### 4.13.3.10 bool operacje::operator==( float \* tab1 )

Przeciażenie operatora porównania; metoda porównuje zawartości dwóch tablic.

##### Parametry

in	<code>tab1</code>	- tablica, której wartości porównujemy
----	-------------------	--

##### Zwraca

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

Definicja w linii 69 pliku operacje.cpp.

#### 4.13.3.11 float& operacje::operator[]( int ind ) [inline]

Definicja w linii 88 pliku operacje.hh.

4.13.3.12 void operacje::quick\_sort( int *l*, int *p* )

Metoda Dokonuje sortownia szybkiego.

**Parametry**

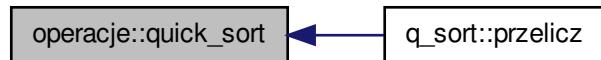
in	<i>l</i>	- pierwszy indeks tablicy
in	<i>p</i>	- ostatni indeks tablicy

Definicja w linii 77 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

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

Metoda zamienia 2 elementy tablicy.

**Parametry**

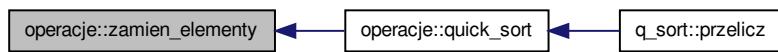
in	<i>i</i>	- element tablicy
in	<i>j</i>	- element tablicy

Zwroca

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

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



#### 4.13.4 Dokumentacja atrybutów składowych

##### 4.13.4.1 int operacje::n

ilosc elementow w tablicy

Definicja w linii 16 pliku operacje.hh.

##### 4.13.4.2 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](#)

### 4.14 Dokumentacja klasy q\_sort

klasa reprezentuje dane poddane sortowaniu szybkiemu

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

Diagram dziedziczenia dla q\_sort

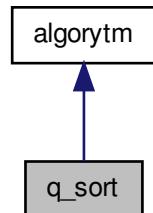
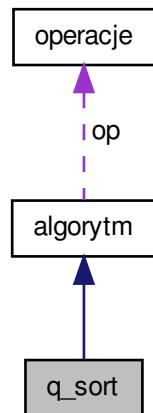


Diagram współpracy dla q\_sort:



### Metody publiczne

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

#### 4.14.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkiemu

Definicja w linii 199 pliku algorytm.hh.

#### 4.14.2 Dokumentacja konstruktora i destruktora

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

konstruktor klasy

Definicja w linii 202 pliku algorytm.hh.

#### 4.14.3 Dokumentacja funkcji składowych

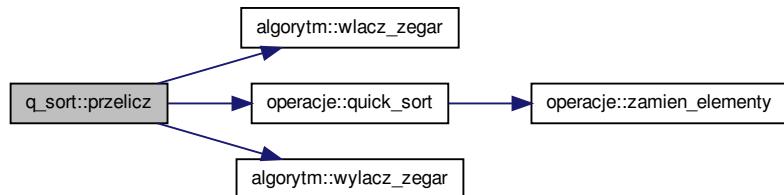
##### 4.14.3.1 `float q_sort::przelicz ( ) [virtual]`

metoda dokonujaca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 171 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

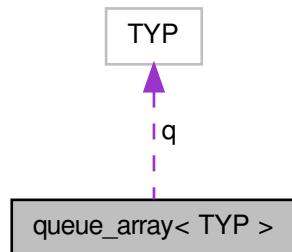
- [algorytm.hh](#)
- [algorytm.cpp](#)

#### 4.15 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 flagę na zadana pozycję*
- int **size ()**
- bool **is\_empty ()**
- void **enqueue (TYP &element)**  
*Dodaje element na początek kolejki w zależności od wybranego trybu powiększania tablicy.*
- TYP **dequeue ()**  
*usuwa element z końca kolejki*
- void **clear ()**  
*czyszcza kolejkę*

### Atrybuty publiczne

- **flag f**  
*flaga trybu zwiększenia pamięci, przyjmuje wartość : plus1 - dla trybu kazdorazowego powiększania pamięci x2 - dla trybu podwajania rozmiaru struktury*

### Atrybuty prywatne

- TYP \* **q**
- int **s**
- int **sp**

#### 4.15.1 Opis szczegółowy

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

Modeluje kolejkę w oparciu o tablice.

Definicja w linii 50 pliku kolejka.hh.

#### 4.15.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 63 pliku kolejka.hh.

4.15.2.2 `template<typename TYP> queue_array< TYP >::queue_array( flag F ) [inline]`

konstruktor parametryczny - ustawia flagę na zadana pozycję

Definicja w linii 65 pliku kolejka.hh.

#### 4.15.3 Dokumentacja funkcji składowych

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

czyszcza kolejkę

Definicja w linii 173 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



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

usuwa element z konca kolejki

Definicja w linii 129 pliku kolejka.hh.

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

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

Definicja w linii 82 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



4.15.3.4 `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.

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

Zwraca

rozmiar kolejki

Definicja w linii 70 pliku kolejka.hh.

#### 4.15.4 Dokumentacja atrybutów składowych

**4.15.4.1 template<typename TYP> flag queue\_array< TYP >::f**

flaga trybu zwiększenia pamięci , przyjmuje wartość : plus1 - dla trybu kazdorazowego powiększania pamięci x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 59 pliku kolejka.hh.

**4.15.4.2 template<typename TYP> TYP\* queue\_array< TYP >::q [private]**

Definicja w linii 51 pliku kolejka.hh.

**4.15.4.3 template<typename TYP> int queue\_array< TYP >::s [private]**

Definicja w linii 52 pliku kolejka.hh.

**4.15.4.4 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](#)

## 4.16 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](#)

### 4.16.1 Opis szczegółowy

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

Modeluje kolejkę opartą na liscie STL.

Definicja w linii 19 pliku kolejka.hh.

### 4.16.2 Dokumentacja funkcji składowych

4.16.2.1 template<typename TYP> void queue\_list< TYP >::clear( ) [inline]

czyszcí stos

Definicja w linii 41 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



4.16.2.2 template<typename TYP> TYP queue\_list< TYP >::dequeue( ) [inline]

usuwa element

Definicja w linii 35 pliku kolejka.hh.

4.16.2.3 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:



#### 4.16.2.4 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.

#### 4.16.2.5 template<typename TYP> int queue\_list< TYP >::size( ) [inline]

Zwraca

rozmiar kolejki

Definicja w linii 31 pliku kolejka.hh.

### 4.16.3 Dokumentacja atrybutów składowych

#### 4.16.3.1 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:

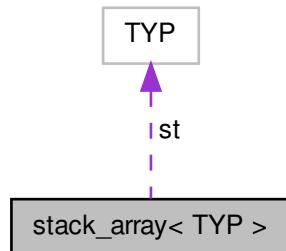
- [kolejka.hh](#)

## 4.17 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 flagę na zadana pozycję*
- **bool is\_empty ()**
- **int size ()**
- **void push (TYP element)**  
*Dodaje element na wierzch stosu w zależności od wybranego trybu powiększania tablicy.*
- **TYP pop ()**  
*zdejmuje element ze stosu*
- **void clear ()**  
*czyszczy stos*

### Atrybuty publiczne

- **flag f**  
*flaga trybu zwiększenia pamięci, przyjmuje wartości:  
plus1 - dla trybu kazdorazowego powiększania pamięci  
x2 - dla trybu podwajania rozmiaru struktury*

### Atrybuty prywatne

- **TYP \* st**
- **int s**
- **int sp**

#### 4.17.1 Opis szczegółowy

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

Modeluje stos w oparciu o tablice.

Definicja w linii 59 pliku stos.hh.

#### 4.17.2 Dokumentacja konstruktora i destruktora

4.17.2.1 template<typename TYP> **stack\_array< TYP >::stack\_array( )** [inline]

konstruktor bezparametryczny

Definicja w linii 72 pliku stos.hh.

4.17.2.2 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.

#### 4.17.3 Dokumentacja funkcji składowych

4.17.3.1 template<typename TYP> **void stack\_array< TYP >::clear( )** [inline]

czysci stos

Definicja w linii 183 pliku stos.hh.

Oto graf wywoływań tej funkcji:



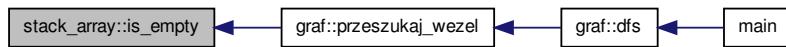
4.17.3.2 template<typename TYP> **bool stack\_array< TYP >::is\_empty( )** [inline]

Zwrota

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

Definicja w linii 79 pliku stos.hh.

Oto graf wywoływań tej funkcji:

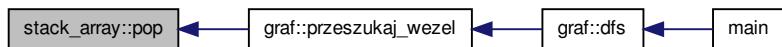


#### 4.17.3.3 template<typename TYP> TYP stack\_array< TYP >::pop ( ) [inline]

zdejmuje element ze stosu

Definicja w linii 140 pliku stos.hh.

Oto graf wywoływań tej funkcji:

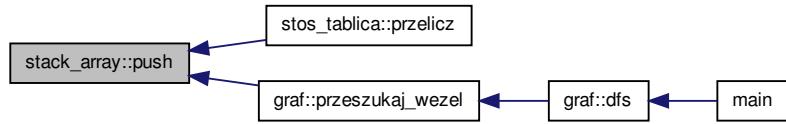


#### 4.17.3.4 template<typename TYP> void stack\_array< TYP >::push ( TYP element ) [inline]

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

Definicja w linii 91 pliku stos.hh.

Oto graf wywoływań tej funkcji:



#### 4.17.3.5 template<typename TYP> int stack\_array< TYP >::size( ) [inline]

Zwraca

rozmiar ztosu

Definicja w linii 87 pliku stos.hh.

#### 4.17.4 Dokumentacja atrybutów składowych

##### 4.17.4.1 template<typename TYP> flag stack\_array< TYP >::f

flaga trybu zwiększenia pamięci , przyjmuje wartości :

plus1 - dla trybu kazdorazowego powiększania pamięci

x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 68 pliku stos.hh.

##### 4.17.4.2 template<typename TYP> int stack\_array< TYP >::s [private]

Definicja w linii 61 pliku stos.hh.

##### 4.17.4.3 template<typename TYP> int stack\_array< TYP >::sp [private]

Definicja w linii 61 pliku stos.hh.

##### 4.17.4.4 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](#)

## 4.18 Dokumentacja szablonu klasy stack\_list< TYP >

Modeluje stos oparty na liscie STL.

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

### Metody publiczne

- bool **is\_empty** ()
- int **size** ()
- void **push** (TYP &element)

*Dodaje element na wierzch stosu.*

- TYP **pop** ()

*zdejmuje element z wierzchu stosu*

- void **clear** ()

*czysci stos*

### Atrybuty prywatne

- list< TYP > **st**

#### 4.18.1 Opis szczegółowy

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

Modeluje stos oparty na liscie STL.

Definicja w linii 22 pliku stos.hh.

#### 4.18.2 Dokumentacja funkcji składowych

##### 4.18.2.1 template<typename TYP> void stack\_list< TYP >::clear ( ) [inline]

czysci stos

Definicja w linii 50 pliku stos.hh.

Oto graf wywoływań tej funkcji:



#### 4.18.2.2 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.

#### 4.18.2.3 template<typename TYP> TYP stack\_list< TYP >::pop( ) [inline]

zdejmuje element z wierzchu stosu

Definicja w linii 42 pliku stos.hh.

#### 4.18.2.4 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:



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

Zwraca

rozmiar ztrosu

Definicja w linii 34 pliku `stos.hh`.

### 4.18.3 Dokumentacja atrybutów składowych

4.18.3.1 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](#)

## 4.19 Dokumentacja klasy `stos_list`

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla `stos_list`

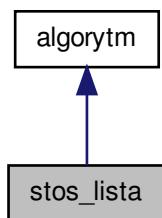
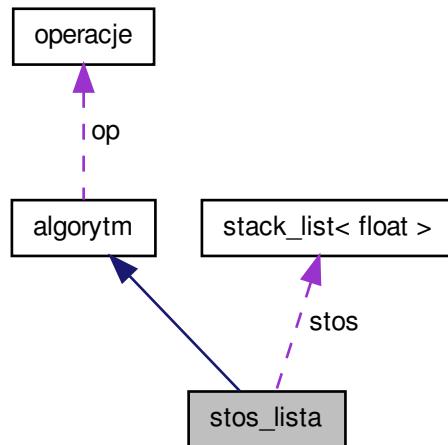


Diagram współpracy dla stos\_listy:



### Metody publiczne

- **stos\_lista** (ifstream &plik1, ifstream &plik2, int N, int M)
- float **przelicz** ()

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.*

### Atrybuty prywatne

- **stack\_list< float > stos**

#### 4.19.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 169 pliku algorytm.hh.

#### 4.19.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 172 pliku algorytm.hh.

### 4.19.3 Dokumentacja funkcji składowych

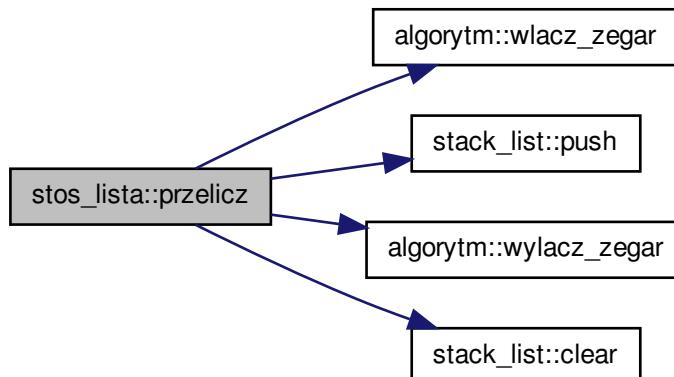
4.19.3.1 `float stos_lista::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorymem.

Reimplementowana z [algorytm](#).

Definicja w linii 136 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



### 4.19.4 Dokumentacja atrybutów składowych

4.19.4.1 `stack_list<float> stos_lista::stos [private]`

Definicja w linii 170 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.20 Dokumentacja klasy stos\_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla stos\_tablica

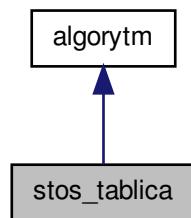
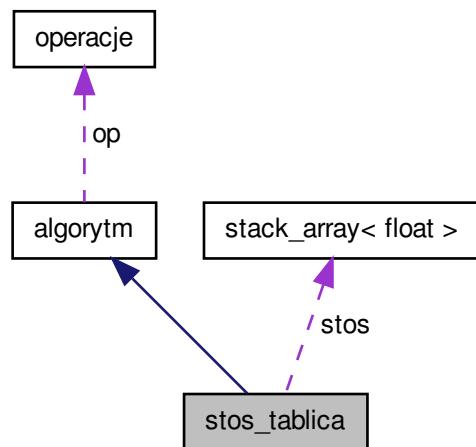


Diagram współpracy dla stos\_tablica:



### Metody publiczne

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

*konstruktor - ustawia flagę w zadany stan*

- `float przelicz ()`

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.*

### Atrybuty prywatne

- `stack_array< float > stos`

#### 4.20.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 157 pliku algorytm.hh.

#### 4.20.2 Dokumentacja konstruktora i destruktora

##### 4.20.2.1 `stos_tablica::stos_tablica ( ifstream & plik1, ifstream & plik2, int N, int M, flag F ) [inline]`

konstruktor - ustawia flagę w zadany stan

Definicja w linii 163 pliku algorytm.hh.

#### 4.20.3 Dokumentacja funkcji składowych

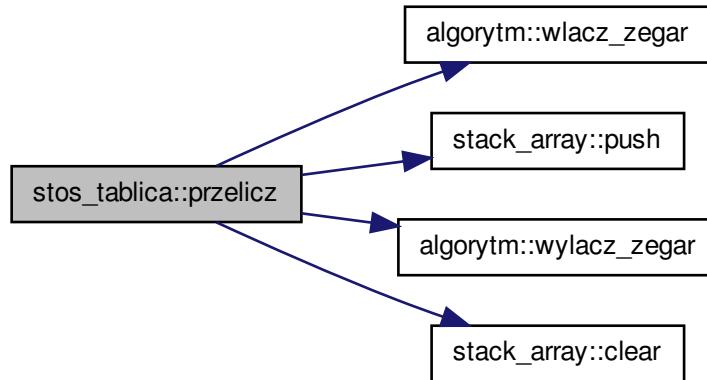
##### 4.20.3.1 `float stos_tablica::przelicz ( ) [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 125 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



#### 4.20.4 Dokumentacja atrybutów składowych

##### 4.20.4.1 stack\_array<float> stos\_tablica::stos [private]

Definicja w linii 158 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

#### 4.21 Dokumentacja klasy tab\_aso

Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla tab\_aso

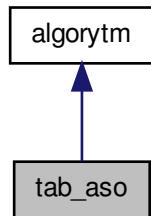
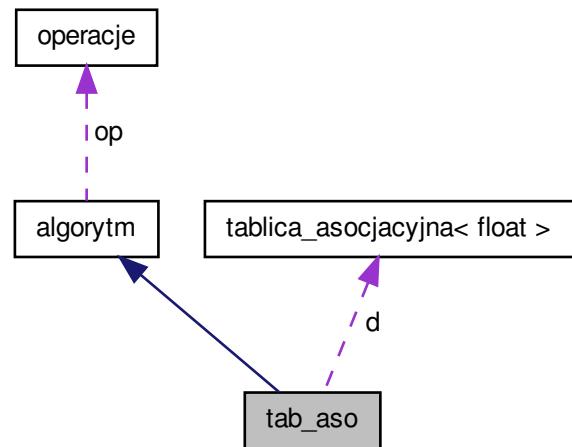


Diagram współpracy dla tab\_aso:



### Metody publiczne

- void [wczytaj\\_klucze](#) (ifstream &plik)  
*wczytywanie kluczów*
- [tab\\_aso](#) (ifstream &plik1, ifstream &plik2, ifstream &plik3, int N, int M)  
*konstruktor*

- float [przelicz \(\)](#)

*Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.*

## Atrybuty prywatne

- [tablica\\_asocjacyjna< float > d](#)

*tablica asocjacyjna*

- string \* [klucze](#)

*wczytywanie kluczów*

### 4.21.1 Opis szczegółowy

Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.

Definicja w linii 264 pliku algorytm.hh.

### 4.21.2 Dokumentacja konstruktora i destruktora

#### 4.21.2.1 [tab\\_aso::tab\\_aso \( ifstream & plik1, ifstream & plik2, ifstream & plik3, int N, int M \) \[inline\]](#)

konstruktor

Definicja w linii 277 pliku algorytm.hh.

### 4.21.3 Dokumentacja funkcji składowych

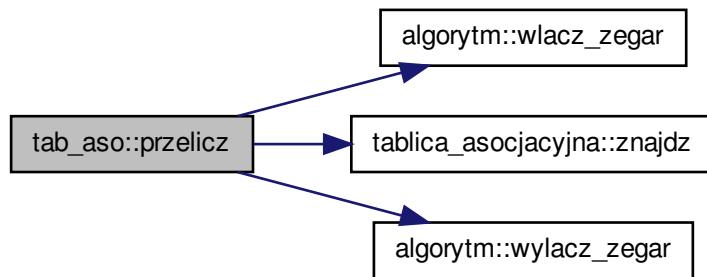
#### 4.21.3.1 [float tab\\_aso::przelicz\( \) \[virtual\]](#)

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:



#### 4.21.3.2 void tab\_aso::wczytaj\_klucze ( ifstream & plik )

wczytywanie klucz

##### Parametry

in	<i>plik</i>	- strumien z kluczmai uzytymi podczas testow
----	-------------	--

Definicja w linii 241 pliku algorytm.cpp.

#### 4.21.4 Dokumentacja atrybutów składowych

##### 4.21.4.1 tablica\_asocjacyjna<float> tab\_aso::d [private]

tablica asocjacyjna

Definicja w linii 266 pliku algorytm.hh.

##### 4.21.4.2 string\* tab\_aso::klucze [private]

wczytywanie klucz

##### Parametry

in	<i>plik</i>	- strumien z kluczmai uzytymi podczas testow
----	-------------	--

Definicja w linii 270 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 4.22 Dokumentacja szablonu klasy tablica\_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjne.

```
#include <tablica_asocjacyjna.hh>
```

### Metody publiczne

- **tablica\_asocjacyjna ()**

*Konstruktor klasy; ustawia nastepujace parametry.*

- void **dodaj** (string k, TYP v)

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

- void **usun** (string k)

*Metoda usuwa zadany element, korzystajac z funkcji znajdz.*

- TYP **pobierz** (string k)

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

- bool **znajdz** (string k)

- bool **czy\_pusta** ()

- int **zlicz\_elementy** ()

- void **wypisz** ()

- TYP **wez** (int ind)

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

- string **wez\_id** (int ind)

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

- bool **czy\_blokada** ()

*metoda sprawdza, czy dostep do tablicy jest mozliwy*

- void **zablokuj** ()

*metoda blokuje dostep do tablicy*

- void **odblokuj** ()

*metoda zezwala na dostep do tablicy*

- void **ustaw** (string k, TYP v)

*metoda zmienia wartosc w miejscu, ktore wskazuje klucz k*

## Metody prywatne

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

*Metoda ktora umieszcza wartosc oraz jej klucz w zadanym 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**

### 4.22.1 Opis szczegółowy

template<typename TYP> class tablica\_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjne.

Definicja w linii 18 pliku tablica\_asocjacyjna.hh.

### 4.22.2 Dokumentacja konstruktora i destruktora

4.22.2.1 template<typename TYP> tablica\_asocjacyjna< TYP >::tablica\_asocjacyjna  
( ) [inline]

Konstruktor klasy; ustawia nastepujace parametry.

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

Definicja w linii 122 pliku tablica\_asocjacyjna.hh.

#### 4.22.3 Dokumentacja funkcji składowych

**4.22.3.1 template<typename TYP> bool tablica\_asocjacyjna< TYP >::czy\_blokada( ) [inline]**

metoda sprawdza, czy dostep do tablicy jest mozliwy

Zwroca

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

Definicja w linii 218 pliku tablica\_asocjacyjna.hh.

**4.22.3.2 template<typename TYP> bool tablica\_asocjacyjna< TYP >::czy\_pusta( ) [inline]**

Zwroca

true, gdy stos jest pusty, false w przeciwnym wypadku

Definicja w linii 186 pliku tablica\_asocjacyjna.hh.

**4.22.3.3 template<typename TYP> void tablica\_asocjacyjna< TYP >::dodaj( string k, TYP v ) [inline]**

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

Definicja w linii 128 pliku tablica\_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.22.3.4 template<typename TYP> void tablica\_asocjacyjna< TYP >::insert ( int *ind*, string *k*, TYP *v* ) [inline, private]

Metoda ktora umieszcza wartosc oraz jej klucz w zadanym miejscu. Gdy wartosc z klu-czem 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 gospo-darowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.

Definicja w linii 35 pliku tablica\_asocjacyjna.hh.

4.22.3.5 template<typename TYP> void tablica\_asocjacyjna< TYP >::odblokuj ( ) [inline]

metoda zezwala na dostep do tablicy

Definicja w linii 226 pliku tablica\_asocjacyjna.hh.

4.22.3.6 template<typename TYP> TYP tablica\_asocjacyjna< TYP >::pobierz ( string *k* ) [inline]

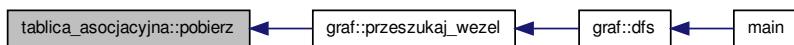
Metoda zwraca uzytkownikowi szukany element, pod warunkiem, ze jest on w zbiorze.

#### Zwraca

szukany element Gdy slownik jest pusty lub szukany element nie istnieje, uzytkow-nik zostaje o tym poinformowany

Definicja w linii 168 pliku tablica\_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.22.3.7 template<typename TYP> void tablica\_asocjacyjna< TYP >::ustaw ( string *k*, TYP *v* ) [inline]

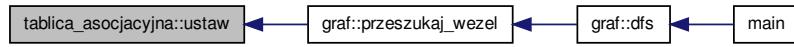
metoda zmienia wartosc w miejscu, ktore wskazuje klucz *k*

#### Parametry

in	<i>k</i>	- klucz
in	<i>v</i>	wartosc, ktora ma zastapic dotychczasowa wartosc w tabli-cy

Definicja w linii 233 pliku tablica\_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



**4.22.3.8 template<typename TYP> void tablica\_asocjacyjna< TYP >::usun ( string *k* )  
[inline]**

Metoda usuwa zadany element, korzystajac z funkcji znajdz.

Definicja w linii 143 pliku tablica\_asocjacyjna.hh.

**4.22.3.9 template<typename TYP> TYP tablica\_asocjacyjna< TYP >::wez ( int *ind* )  
[inline]**

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

#### Parametry

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

#### Zwraca

value[ind] - wartosc mieszczaca sie pod zadanim indeksem

Definicja w linii 202 pliku tablica\_asocjacyjna.hh.

**4.22.3.10 template<typename TYP> string tablica\_asocjacyjna< TYP >::wez\_id ( int *ind* ) [inline]**

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

#### Parametry

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

Zwroca

key[ind] - klucz mieszczacy sie pod zadanym indeksem

Definicja w linii 210 pliku tablica\_asocjacyjna.hh.

4.22.3.11 template<typename TYP> void tablica\_asocjacyjna< TYP >::wstaw ( string k, TYP v, int ind\_l, int ind\_r ) [inline, private]

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

Definicja w linii 83 pliku tablica\_asocjacyjna.hh.

4.22.3.12 template<typename TYP> void tablica\_asocjacyjna< TYP >::wypisz ( ) [inline]

Definicja w linii 193 pliku tablica\_asocjacyjna.hh.

4.22.3.13 template<typename TYP> void tablica\_asocjacyjna< TYP >::zablokuj ( ) [inline]

metoda blokuje dostep do tablicy

Definicja w linii 222 pliku tablica\_asocjacyjna.hh.

4.22.3.14 template<typename TYP> int tablica\_asocjacyjna< TYP >::zlicz\_elementy ( ) [inline]

Zwroca

ilosc elementow w strukturze

Definicja w linii 191 pliku tablica\_asocjacyjna.hh.

4.22.3.15 template<typename TYP> int tablica\_asocjacyjna< TYP >::znajdz ( string k, int ind\_l, int ind\_r ) [inline, private]

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

Zwrota

indeks szukanego elementu

Definicja w linii 100 pliku tablica\_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



4.22.3.16 template<typename TYP> bool tablica\_asocjacyjna< TYP >::znajdz ( string k ) [inline]

Definicja w linii 178 pliku tablica\_asocjacyjna.hh.

#### 4.22.4 Dokumentacja atrybutów składowych

4.22.4.1 template<typename TYP> bool tablica\_asocjacyjna< TYP >::blok [private]

Definicja w linii 29 pliku tablica\_asocjacyjna.hh.

4.22.4.2 template<typename TYP> bool tablica\_asocjacyjna< TYP >::found [private]

flaga informujaca o tym, czy dany klucz znalezione w zbiorze

Definicja w linii 28 pliku tablica\_asocjacyjna.hh.

4.22.4.3 template<typename TYP> string\* tablica\_asocjacyjna< TYP >::key [private]

Tablica zawierajaca klucze poszukiwan.

Definicja w linii 20 pliku tablica\_asocjacyjna.hh.

4.22.4.4 template<typename TYP> int tablica\_asocjacyjna< TYP >::s [private]

rozmiar tablicy

Definicja w linii 24 pliku tablica\_asocjacyjna.hh.

4.22.4.5 template<typename TYP> int tablica\_asocjacyjna< TYP >::sp [private]

rozmiar danych zapelniajacych tablice

Definicja w linii 26 pliku tablica\_asocjacyjna.hh.

4.22.4.6 template<typename TYP> TYP\* tablica\_asocjacyjna< TYP >::value [private]

Tablica zawierajaca wartosci.

Definicja w linii 22 pliku tablica\_asocjacyjna.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

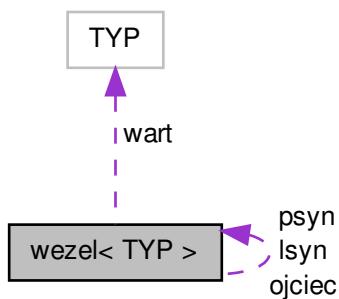
- [tablica\\_asocjacyjna.hh](#)

## 4.23 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*

#### 4.23.1 Opis szczegółowy

**template<typename TYP> class wezel< TYP >**

modeluje pojedynczy wezel drzewa

Definicja w linii 14 pliku drzewo.hh.

#### 4.23.2 Dokumentacja konstruktora i destruktora

##### 4.23.2.1 **template<typename TYP> wezel< TYP >::wezel ( ) [inline]**

konstruktor bezparametryczny

Definicja w linii 29 pliku drzewo.hh.

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

konstruktor parametryczny

#### Parametry

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

Definicja w linii 35 pliku drzewo.hh.

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

destruktor

Definicja w linii 37 pliku drzewo.hh.

### 4.23.3 Dokumentacja funkcji składowych

4.23.3.1 `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:



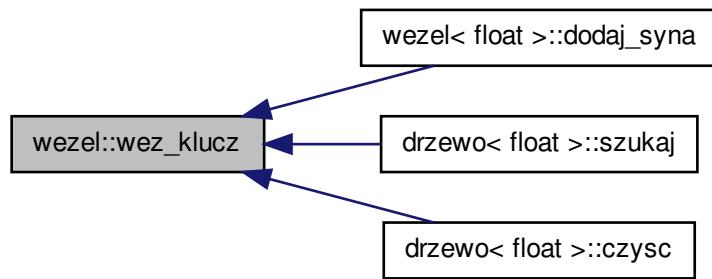
4.23.3.2 `template<typename TYP> string wezel< TYP >::wez_klucz( ) [inline]`

Zwrota

klucz wezla

Definicja w linii 41 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



#### 4.23.3.3 template<typename TYP> TYP wezel< TYP >::wez\_wart( ) [inline]

Zwrota

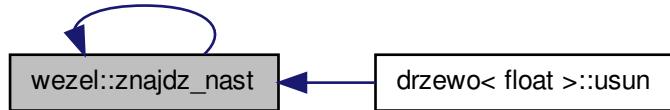
wartosc wezla

Definicja w linii 45 pliku drzewo.hh.

#### 4.23.3.4 template<typename TYP> wezel<TYP>\* wezel< TYP >::znajdz\_nast( ) [inline]

Definicja w linii 61 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



#### 4.23.4 Dokumentacja atrybutów składowych

##### 4.23.4.1 template<typename TYP> syn wezel< TYP >::flag

okresla, czyim synem jest wezel

Definicja w linii 21 pliku drzewo.hh.

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

klucz sluzacy do wyszukiwania

Definicja w linii 16 pliku drzewo.hh.

##### 4.23.4.3 template<typename TYP> wezel\* wezel< TYP >::lsyn

wskaznik na lewego syna wezla

Definicja w linii 25 pliku drzewo.hh.

##### 4.23.4.4 template<typename TYP> wezel\* wezel< TYP >::ojciec

wskaznik na ojca danego wezla

Definicja w linii 23 pliku drzewo.hh.

##### 4.23.4.5 template<typename TYP> wezel\* wezel< TYP >::psyn

wskaznik na prawego syna wezla

Definicja w linii 27 pliku drzewo.hh.

4.23.4.6 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](#)

## 4.24 Dokumentacja klasy wierzcholek

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

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

### Metody publiczne

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

*konstruktor*

### Atrybuty prywatne

- int [id](#)

*numer identyfikacyjny wierzchołka*

- int [waga](#)

*waga wezla, uzywana w stosunku do wierzchołka, z którym ow wierzchołek jest incydenty*

### Przyjaciele

- class [graf](#)

*klasa zparzystazniona*

### 4.24.1 Opis szczegółowy

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

Definicja w linii 16 pliku graf.hh.

#### 4.24.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Parametry

in	wz	- id wierzcholka
in	wg	- waga

Definicja w linii 29 pliku graf.hh.

#### 4.24.3 Dokumentacja przyjaciol i funkcji zwiazanych

4.24.3.1 **friend class graf [friend]**

klasa zparzyjazniona

Definicja w linii 18 pliku graf.hh.

#### 4.24.4 Dokumentacja atrybutów składowych

4.24.4.1 **int wierzcholek::id [private]**

numer indentyfikacyjny wierzcholka

Definicja w linii 21 pliku graf.hh.

4.24.4.2 **int wierzcholek::waga [private]**

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

Definicja w linii 23 pliku graf.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [graf.hh](#)



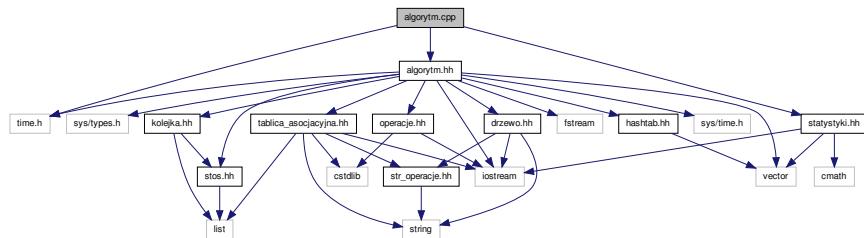
## Rozdział 5

# 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](#)

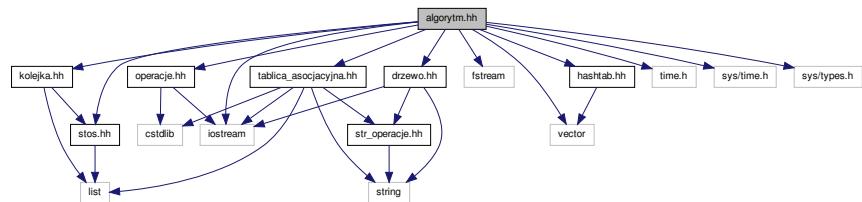
Definicja w pliku [algorytm.cpp](#).

### 5.2 Dokumentacja pliku algorytm.hh

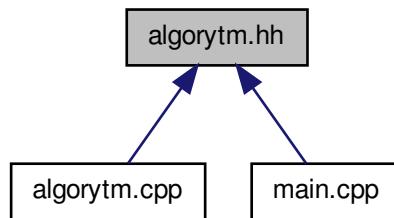
Definicja klas wykonujących operacje na zestawie danych wejściowych.

```
#include <iostream> #include <fstream> #include <vector> x
#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" x  
 #include "tablica\_asocjacyjna.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 dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2*
- class **stos\_tablica**  
*klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury*
- class **stos\_lista**  
*klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury*
- class **kolejka\_tablica**  
*klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury*
- class **kolejka\_lista**

*klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury*

- class [q\\_sort](#)

*klasa reprezentuje dane poddane sortowaniu szybkiemu*

- class [h\\_sort](#)

*klasa reprezentuje dane poddane sortowaniu przez kopcowanie*

- class [m\\_sort](#)

*klasa reprezentuje dane poddane sortowaniu przez scalanie*

- class [bst](#)

*Modeluje drzewo binarne przeznaczone do testowania szybkości wyszukiwania.*

- class [h\\_table](#)

*Modeluje tablice haszujaca przeznaczona do testowania szybkości wyszukiwania.*

- class [tab\\_aso](#)

*Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.*

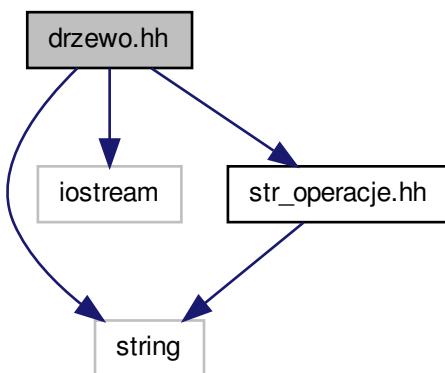
### 5.2.1 Opis szczegółowy

Definicja klas wykonujących operacje na zestawie danych wejściowych.

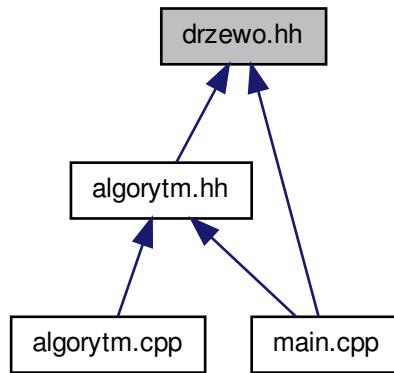
Definicja w pliku [algorytm.hh](#).

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

## Wyliczenia

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

### 5.3.1 Opis szczegółowy

Plik zawiera definicje klasy reprezentującej drzewo binarne

Definicja w pliku `drzewo.hh`.

### 5.3.2 Dokumentacja typów wyliczanych

#### 5.3.2.1 enum `syn`

typ wyliczeniowy, określa, czyim synem jest dany element drzewa

Wartości wyliczeń:

*lewy*

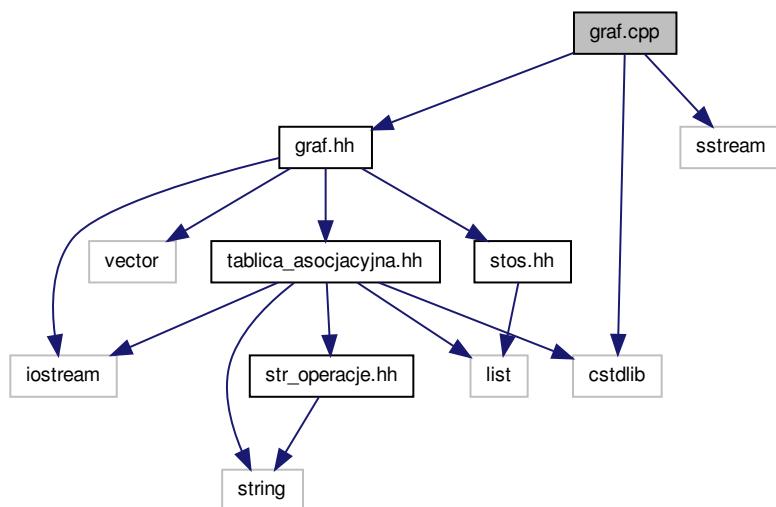
*zaden*

*prawy*

Definicja w linii 11 pliku drzewo.hh.

## 5.4 Dokumentacja pliku graf.cpp

```
#include "graf.hh" #include <sstream> #include <cstdlib> ×  
Wykres zależności załączania dla graf.cpp:
```



### Zmienne

- `tablica_asocjacyjna< int > vec`

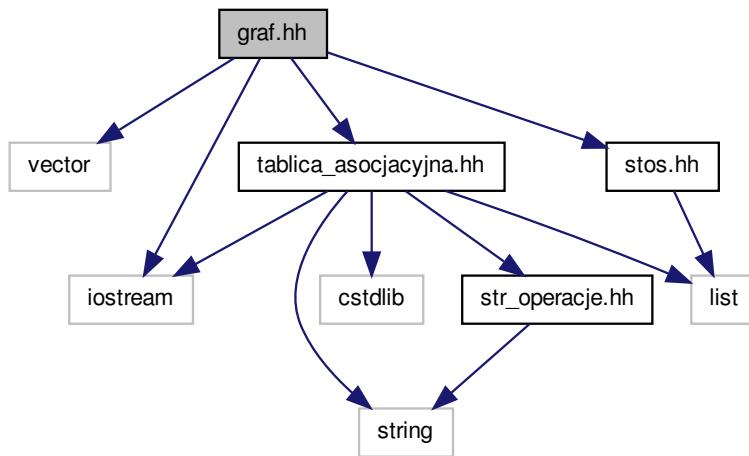
#### 5.4.1 Dokumentacja zmiennych

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

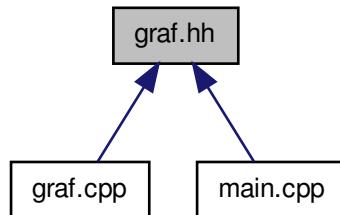
Definicja w linii 4 pliku graf.cpp.

## 5.5 Dokumentacja pliku graf.hh

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

- class [graf](#)

Klasa modeluje pojęcie grafu w oparciu o liste incydencji. Operacje na grafie możliwe są na dwa sposoby \n 1. Podając wierzchołek grafu jako parametr metody \n 2. - Podając id wierzchołka jako parametr metody.

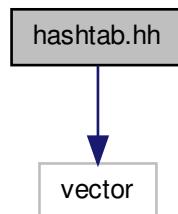
### 5.5.1 Opis szczegółowy

Plik zawiera definicje klasy wierzchołek i klasy graf

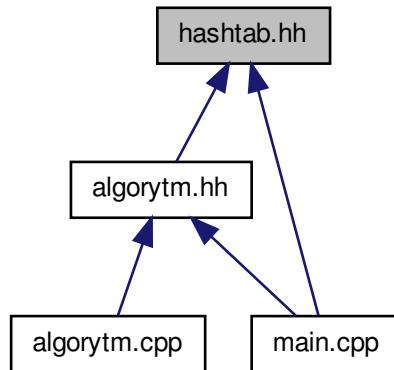
Definicja w pliku [graf.hh](#).

## 5.6 Dokumentacja pliku hashtable.hh

#include <vector> Wykres zależności załączania dla hashtable.hh:



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



## Komponenty

- class `el_tab< TYP >`  
*pojedynczy element tablicy haszujacej*
- class `hashtab< TYP >`  
*modeluje tablice haszujaca w oparciu o kontener klasy `el_tab`*

### 5.6.1 Opis szczegółowy

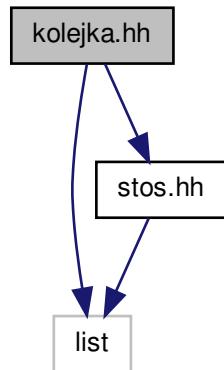
Plik zawiera definicje klasy reprezentującej tablice haszujaca

Definicja w pliku `hashtab.hh`.

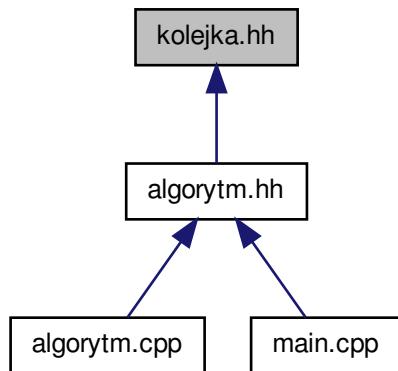
## 5.7 Dokumentacja pliku kolejka.hh

Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kązdrozowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy kolejka się przepieni.

```
#include <list> #include "stos.hh" Wykres zależności załączania dla kolejka.hh:
```



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



## Komponenty

- class `queue_list< TYP >`

*Modeluje kolejke oparta na liscie STL.*

- class `queue_array< TYP >`

*Modeluje kolejki w oparciu o tablice.*

### 5.7.1 Opis szczegółowy

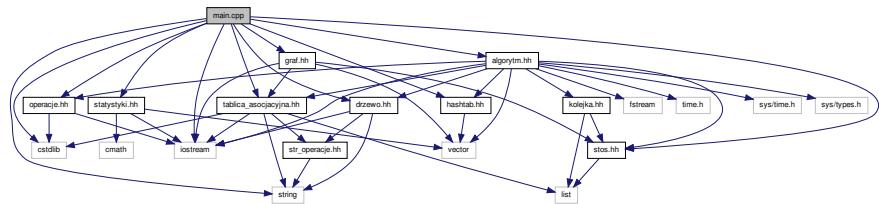
Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy kolejka się przepelni.

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

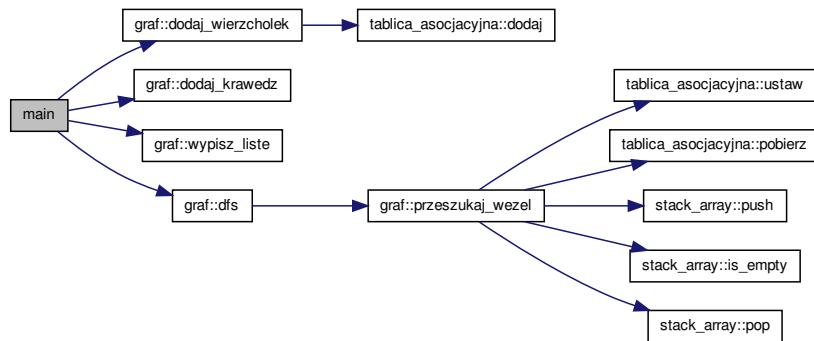
Definicja w pliku [main.cpp](#).

### **5.8.2 Dokumentacja funkcji**

### 5.8.2.1 int main ( )

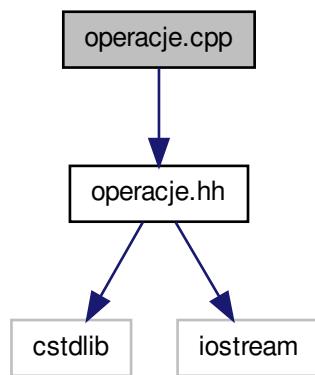
Definicja w linii 19 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



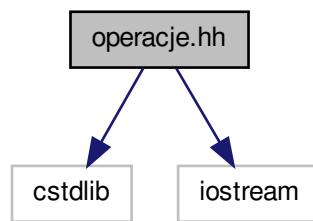
## 5.9 Dokumentacja pliku operacje.cpp

#include "operacje.hh" Wykres zależności załączania dla operacje.cpp:

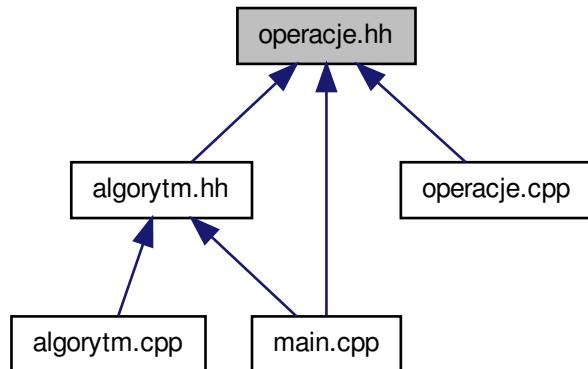


## 5.10 Dokumentacja pliku operacje.hh

#include <cstdlib> #include <iostream> Wykres zależności załączania dla operacje.hh:



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



## Komponenty

- class [operacje](#)

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

## Definicje

- `#define ROZMIAR 9`

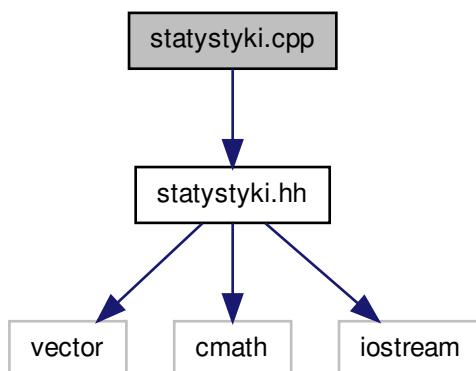
### 5.10.1 Dokumentacja definicji

#### 5.10.1.1 `#define ROZMIAR 9`

Definicja w linii 3 pliku operacje.hh.

## 5.11 Dokumentacja pliku statystyki.cpp

`#include "statystyki.hh"` Wykres zależności załączania dla statystyki.cpp:



## Funkcje

- float `srednia` (float \*tab, int rozmiar)  
*funckja oblicza wartosc srednia*
- float `odchylenie_standardowe` (float `srednia`, float \*tab, int rozmiar)  
*funckja oblicza odchylenie standardowe*

### 5.11.1 Dokumentacja funkcji

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

funckja oblicza odchylenie standardowe

**Parametry**

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

**Zwroca**

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:

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

funckja oblicza wartosc srednia

**Parametry**

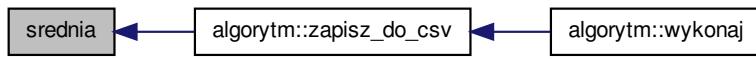
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

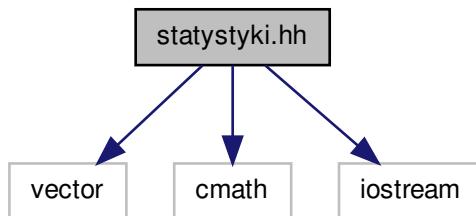
Oto graf wywoływań tej funkcji:



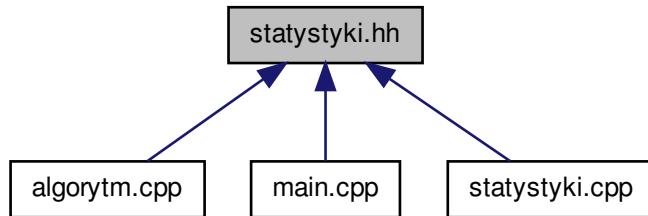
## 5.12 Dokumentacja pliku statystyki.hh

plik zawiera dekalracje funkcji odpowiedzialnych za przeprowadznaie statystyk

```
#include <vector> #include <cmath> #include <iostream> x  
Wykres zależności załączania dla statystyki.hh:
```



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



## Funkcje

- float [srednia](#) (float \*tab, int rozmiar)  
*funcja oblicza wartosc srednia*
- float [odchylenie\\_standardowe](#) (float [srednia](#), float \*tab, int rozmiar)  
*funcja oblicza odchylenie standardowe*

### 5.12.1 Opis szczegółowy

plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk  
Definicja w pliku [statystyki.hh](#).

### 5.12.2 Dokumentacja funkcji

#### 5.12.2.1 float [odchylenie\\_standardowe](#) ( float [srednia](#), float \* [tab](#), int [rozmiar](#) )

funcja oblicza odchylenie standardowe

#### Parametry

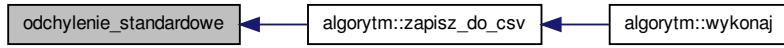
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

#### Zwroca

odchylenie standardowe

Definicja w linii 16 pliku [statystyki.cpp](#).

Oto graf wywoływań tej funkcji:



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

funcja oblicza wartosc srednia

##### Parametry

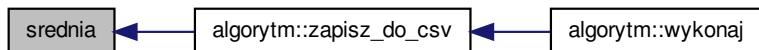
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>rozmiar</i>	- rozmiar tablicy

##### Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

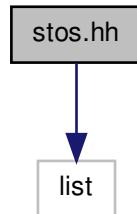
Oto graf wywoływań tej funkcji:



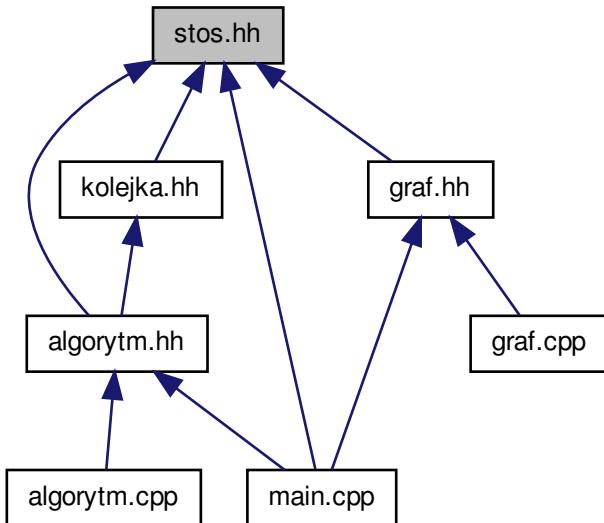
## 5.13 Dokumentacja pliku stos.hh

Plik zawiera definicje klasy `Stos`. Zaimplementowana na 2 sposoby 1. Za pomocą listy.  
2. Za pomocą tablicy a. kazdorazowo powiekszajacej swój rozmiar b. powiekszajacej  
swoj rozmiar dwukrotnie, gdy stos sie przepelní.

```
#include <list> Wykres zależności załączania dla stos.hh:
```



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



## Komponenty

- class [stack\\_list< TYP >](#)  
*Modeluje stos oparty na liscie STL.*

- class `stack_array< TYP >`

*Modeluje stos w oparciu o tablice.*

## Wyliczenia

- enum `flag { plus1, x2 }`

*typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci*

### 5.13.1 Opis szczegółowy

Plik zawiera definicje klasy `Stos`. Zaimplementowana na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. każdorazowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy stos się przepelni.

Definicja w pliku `stos.hh`.

### 5.13.2 Dokumentacja typów wyliczanych

#### 5.13.2.1 enum `flag`

*typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci*

Wartości wyliczeń:

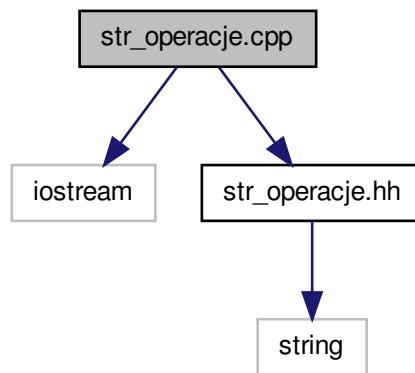
**`plus1`**

**`x2`**

Definicja w linii 17 pliku `stos.hh`.

## 5.14 Dokumentacja pliku str\_operacje.cpp

#include <iostream> #include "str\_operacje.hh" Wykres zależności załączania dla str\_operacje.cpp:



### Funkcje

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

#### 5.14.1 Dokumentacja funkcji

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

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwroca

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

Definicja w linii 6 pliku str\_operacje.cpp.

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

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 26 pliku str\_operacje.cpp.

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

true, gdy łańcuchy są identyczne

Definicja w linii 34 pliku str\_operacje.cpp.

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

funkcja służąca do alfabetycznego porządkowania napisów

**Zwroca**

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

Definicja w linii 17 pliku str\_operacje.cpp.

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

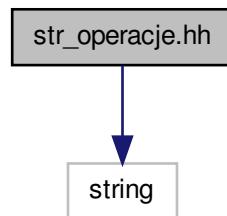
**Zwroca**

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

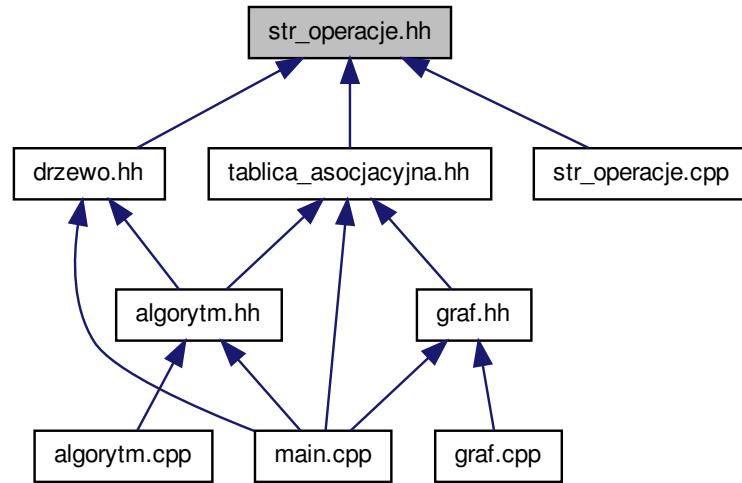
Definicja w linii 30 pliku str\_operacje.cpp.

### 5.15 Dokumentacja pliku str\_operacje.hh

#include <string> Wykres zależności załączania dla str\_operacje.hh:



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



## Funkcje

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

### 5.15.1 Dokumentacja funkcji

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

funkcja sluzaca do alfabetycznego porzadkowania napisow

##### Zwroca

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

Definicja w linii 6 pliku str\_operacje.cpp.

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

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 26 pliku str\_operacje.cpp.

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

true, gdy łańcuchy są identyczne

Definicja w linii 34 pliku str\_operacje.cpp.

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

funkcja służąca do alfabetycznego porządkowania napisów

**Zwroca**

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

Definicja w linii 17 pliku str\_operacje.cpp.

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

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

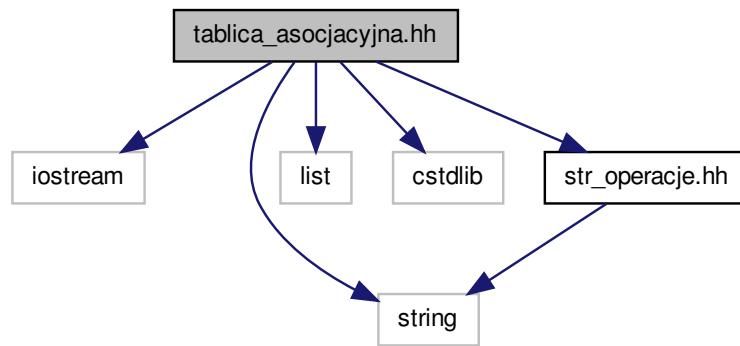
Definicja w linii 30 pliku str\_operacje.cpp.

## 5.16 Dokumentacja pliku strona.dox

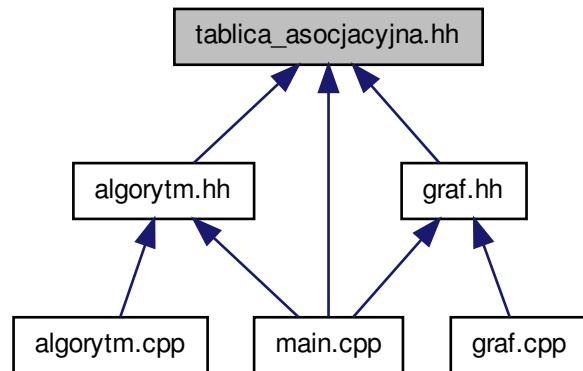
### 5.17 Dokumentacja pliku tablica\_asocjacyjna.hh

```
#include <iostream> #include <string> #include <list> x
#include <cstdlib> #include "str_operacje.hh" Wykres zależno-
```

ści załączania dla tablica\_asocjacyjna.hh:



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



## Komponenty

- class `tablica_asocjacyjna< TYP >`

*Klasa modeluje tablice asocjacyjne.*

### 5.17.1 Opis szczegółowy

Plik zawiera definicje klasy [tablica\\_asocjacyjna](#), oraz definicje funkcji pomocniczych jako przeciążeń operatorów porównania dla klasy typu string

Definicja w pliku [tablica\\_asocjacyjna.hh](#).