

PAMSI - pwilkosz

1.0

Wygenerowano przez Doxygen 1.7.6.1

Sun Apr 13 2014 23:41:23

Spis treści

1 Indeks klas	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy algorytm	7
4.1.1 Opis szczegółowy	10
4.1.2 Dokumentacja konstruktora i destruktora	10
4.1.2.1 algorytm	10
4.1.3 Dokumentacja funkcji składowych	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	16

4.1.3.12	<i>zapisz_do_gnuplot</i>	16
4.1.4	Dokumentacja atrybutów składowych	17
4.1.4.1	<i>czas</i>	17
4.1.4.2	<i>czas1</i>	17
4.1.4.3	<i>czas2</i>	17
4.1.4.4	<i>dane</i>	17
4.1.4.5	<i>dane_wz</i>	17
4.1.4.6	<i>m</i>	17
4.1.4.7	<i>n</i>	17
4.1.4.8	<i>op</i>	17
4.2	Dokumentacja klasy bst	18
4.2.1	Opis szczegółowy	20
4.2.2	Dokumentacja konstruktora i destruktora	20
4.2.2.1	<i>bst</i>	20
4.2.2.2	<i>~bst</i>	20
4.2.3	Dokumentacja funkcji składowych	20
4.2.3.1	<i>przelicz</i>	20
4.2.3.2	<i>wczytaj_klucze</i>	21
4.2.4	Dokumentacja atrybutów składowych	21
4.2.4.1	<i>d</i>	21
4.2.4.2	<i>klucze</i>	21
4.3	Dokumentacja szablonu klasy drzewo< TYP >	21
4.3.1	Opis szczegółowy	22
4.3.2	Dokumentacja konstruktora i destruktora	22
4.3.2.1	<i>drzewo</i>	22
4.3.2.2	<i>drzewo</i>	23
4.3.3	Dokumentacja funkcji składowych	23
4.3.3.1	<i>czysc</i>	23
4.3.3.2	<i>dodaj</i>	23
4.3.3.3	<i>dodaj_wezel</i>	23
4.3.3.4	<i>szukaj</i>	23
4.3.3.5	<i>usun</i>	24
4.3.3.6	<i>wyczysc</i>	24
4.3.3.7	<i>znajdz</i>	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 h_sort	27
4.5.1 Opis szczegółowy	29
4.5.2 Dokumentacja konstruktora i destruktora	29
4.5.2.1 h_sort	29
4.5.3 Dokumentacja funkcji składowych	29
4.5.3.1 przelicz	29
4.6 Dokumentacja klasy h_table	29
4.6.1 Opis szczegółowy	31
4.6.2 Dokumentacja konstruktora i destruktora	31
4.6.2.1 h_table	31
4.6.3 Dokumentacja funkcji składowych	31
4.6.3.1 przelicz	31
4.6.3.2 wczytaj_klucze	32
4.6.4 Dokumentacja atrybutów składowych	32
4.6.4.1 klucze	32
4.7 Dokumentacja szablonu klasy hashtable< TYP >	32
4.7.1 Opis szczegółowy	33
4.7.2 Dokumentacja konstruktora i destruktora	33
4.7.2.1 hashtable	33
4.7.2.2 hashtable	34
4.7.3 Dokumentacja funkcji składowych	34
4.7.3.1 dodaj	34

4.7.3.2	hash	35
4.7.3.3	ustaw_dlugosc	35
4.7.3.4	usun	36
4.7.3.5	wypisz	36
4.7.3.6	znajdz	36
4.7.4	Dokumentacja atrybutów składowych	37
4.7.4.1	dlugosc	37
4.7.4.2	tab	37
4.8	Dokumentacja klasy kolejka_lista	38
4.8.1	Opis szczegółowy	39
4.8.2	Dokumentacja konstruktora i destruktora	39
4.8.2.1	kolejka_lista	39
4.8.3	Dokumentacja funkcji składowych	39
4.8.3.1	przelicz	39
4.8.4	Dokumentacja atrybutów składowych	40
4.8.4.1	qu	40
4.9	Dokumentacja klasy kolejka_tablica	40
4.9.1	Opis szczegółowy	42
4.9.2	Dokumentacja konstruktora i destruktora	42
4.9.2.1	kolejka_tablica	42
4.9.3	Dokumentacja funkcji składowych	42
4.9.3.1	przelicz	42
4.9.4	Dokumentacja atrybutów składowych	43
4.9.4.1	qu	43
4.10	Dokumentacja klasy m_sort	43
4.10.1	Opis szczegółowy	45
4.10.2	Dokumentacja konstruktora i destruktora	45
4.10.2.1	m_sort	45
4.10.3	Dokumentacja funkcji składowych	45
4.10.3.1	przelicz	45
4.11	Dokumentacja klasy mnozenie	45
4.11.1	Opis szczegółowy	47
4.11.2	Dokumentacja konstruktora i destruktora	47
4.11.2.1	mnozenie	47

4.11.3 Dokumentacja funkcji składowych	47
4.11.3.1 przelicz	47
4.12 Dokumentacja klasy operacje	48
4.12.1 Opis szczegółowy	49
4.12.2 Dokumentacja konstruktora i destruktora	49
4.12.2.1 operacje	49
4.12.2.2 operacje	49
4.12.3 Dokumentacja funkcji składowych	49
4.12.3.1 dodaj_element	49
4.12.3.2 dodaj_elementy	49
4.12.3.3 heap_sort	50
4.12.3.4 make_heap	50
4.12.3.5 make_node	51
4.12.3.6 merge	52
4.12.3.7 merge_sort	52
4.12.3.8 odwroc_tablice	53
4.12.3.9 operator=	53
4.12.3.10 operator==	53
4.12.3.11 operator[]	53
4.12.3.12 quick_sort	54
4.12.3.13 zamien_elementy	54
4.12.4 Dokumentacja atrybutów składowych	55
4.12.4.1 n	55
4.12.4.2 tab	55
4.13 Dokumentacja klasy q_sort	55
4.13.1 Opis szczegółowy	57
4.13.2 Dokumentacja konstruktora i destruktora	57
4.13.2.1 q_sort	57
4.13.3 Dokumentacja funkcji składowych	57
4.13.3.1 przelicz	57
4.14 Dokumentacja szablonu klasy queue_array< TYP >	57
4.14.1 Opis szczegółowy	59
4.14.2 Dokumentacja konstruktora i destruktora	59
4.14.2.1 queue_array	59

4.14.2.2	queue_array	59
4.14.3	Dokumentacja funkcji składowych	59
4.14.3.1	clear	59
4.14.3.2	dequeue	60
4.14.3.3	enqueue	60
4.14.3.4	is_empty	60
4.14.3.5	size	60
4.14.4	Dokumentacja atrybutów składowych	60
4.14.4.1	f	61
4.14.4.2	q	61
4.14.4.3	s	61
4.14.4.4	sp	61
4.15	Dokumentacja szablonu klasy queue_list< TYP >	61
4.15.1	Opis szczegółowy	62
4.15.2	Dokumentacja funkcji składowych	62
4.15.2.1	clear	62
4.15.2.2	dequeue	62
4.15.2.3	enqueue	62
4.15.2.4	is_empty	63
4.15.2.5	size	63
4.15.3	Dokumentacja atrybutów składowych	63
4.15.3.1	q	63
4.16	Dokumentacja szablonu klasy stack_array< TYP >	63
4.16.1	Opis szczegółowy	65
4.16.2	Dokumentacja konstruktora i destruktora	65
4.16.2.1	stack_array	65
4.16.2.2	stack_array	65
4.16.3	Dokumentacja funkcji składowych	65
4.16.3.1	clear	65
4.16.3.2	is_empty	65
4.16.3.3	pop	66
4.16.3.4	push	66
4.16.3.5	size	66
4.16.4	Dokumentacja atrybutów składowych	66

4.16.4.1	f	66
4.16.4.2	s	67
4.16.4.3	sp	67
4.16.4.4	st	67
4.17	Dokumentacja szablonu klasy stack_list< TYP >	67
4.17.1	Opis szczegółowy	68
4.17.2	Dokumentacja funkcji składowych	68
4.17.2.1	clear	68
4.17.2.2	is_empty	68
4.17.2.3	pop	68
4.17.2.4	push	68
4.17.2.5	size	69
4.17.3	Dokumentacja atrybutów składowych	69
4.17.3.1	st	69
4.18	Dokumentacja klasy stos_lista	69
4.18.1	Opis szczegółowy	71
4.18.2	Dokumentacja konstruktora i destruktora	71
4.18.2.1	stos_lista	71
4.18.3	Dokumentacja funkcji składowych	71
4.18.3.1	przelicz	71
4.18.4	Dokumentacja atrybutów składowych	72
4.18.4.1	stos	72
4.19	Dokumentacja klasy stos_tablica	72
4.19.1	Opis szczegółowy	73
4.19.2	Dokumentacja konstruktora i destruktora	73
4.19.2.1	stos_tablica	74
4.19.3	Dokumentacja funkcji składowych	74
4.19.3.1	przelicz	74
4.19.4	Dokumentacja atrybutów składowych	74
4.19.4.1	stos	74
4.20	Dokumentacja klasy tab_aso	75
4.20.1	Opis szczegółowy	76
4.20.2	Dokumentacja konstruktora i destruktora	76
4.20.2.1	tab_aso	76

4.20.3 Dokumentacja funkcji składowych	76
4.20.3.1 przelicz	76
4.20.3.2 wczytaj_klucze	77
4.20.4 Dokumentacja atrybutów składowych	77
4.20.4.1 d	77
4.20.4.2 klucze	77
4.21 Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >	78
4.21.1 Opis szczegółowy	79
4.21.2 Dokumentacja konstruktora i destruktora	79
4.21.2.1 tablica_asocjacyjna	79
4.21.3 Dokumentacja funkcji składowych	79
4.21.3.1 czy_pusta	79
4.21.3.2 dodaj	80
4.21.3.3 insert	80
4.21.3.4 pobierz	80
4.21.3.5 usun	80
4.21.3.6 wstaw	80
4.21.3.7 wypisz	81
4.21.3.8 zlicz_elementy	81
4.21.3.9 znajdz	81
4.21.3.10 znajdz	81
4.21.4 Dokumentacja atrybutów składowych	81
4.21.4.1 found	82
4.21.4.2 key	82
4.21.4.3 s	82
4.21.4.4 sp	82
4.21.4.5 value	82
4.22 Dokumentacja szablonu klasy wezel< TYP >	82
4.22.1 Opis szczegółowy	84
4.22.2 Dokumentacja konstruktora i destruktora	84
4.22.2.1 wezel	84
4.22.2.2 ~wezel	84
4.22.2.3 ~wezel	84
4.22.3 Dokumentacja funkcji składowych	84

4.22.3.1 <code>dodaj_syna</code>	84
4.22.3.2 <code>wez_klucz</code>	85
4.22.3.3 <code>wez_wart</code>	85
4.22.3.4 <code>znajdz_nast</code>	86
4.22.4 Dokumentacja atrybutów składowych	86
4.22.4.1 <code>flag</code>	86
4.22.4.2 <code>klucz</code>	86
4.22.4.3 <code>Isyn</code>	86
4.22.4.4 <code>ojciec</code>	86
4.22.4.5 <code>psyn</code>	87
4.22.4.6 <code>wart</code>	87
5 Dokumentacja plików	89
5.1 Dokumentacja pliku <code>algorytm.cpp</code>	89
5.1.1 Opis szczegółowy	89
5.2 Dokumentacja pliku <code>algorytm.hh</code>	89
5.2.1 Opis szczegółowy	91
5.3 Dokumentacja pliku <code>drzewo.hh</code>	91
5.3.1 Opis szczegółowy	92
5.3.2 Dokumentacja typów wyliczanych	92
5.3.2.1 <code>syn</code>	92
5.4 Dokumentacja pliku <code>hashtab.hh</code>	93
5.4.1 Opis szczegółowy	94
5.5 Dokumentacja pliku <code>kolejka.hh</code>	94
5.5.1 Opis szczegółowy	95
5.6 Dokumentacja pliku <code>main.cpp</code>	95
5.6.1 Opis szczegółowy	96
5.6.2 Dokumentacja funkcji	96
5.6.2.1 <code>main</code>	96
5.7 Dokumentacja pliku <code>operacje.cpp</code>	97
5.8 Dokumentacja pliku <code>operacje.hh</code>	97
5.8.1 Dokumentacja definicji	98
5.8.1.1 ROZMIAR	98
5.9 Dokumentacja pliku <code>statystyki.cpp</code>	99

5.9.1 Dokumentacja funkcji	99
5.9.1.1 odchylenie_standardowe	99
5.9.1.2 srednia	100
5.10 Dokumentacja pliku statystyki.hh	100
5.10.1 Opis szczegółowy	101
5.10.2 Dokumentacja funkcji	102
5.10.2.1 odchylenie_standardowe	102
5.10.2.2 srednia	102
5.11 Dokumentacja pliku stos.hh	103
5.11.1 Opis szczegółowy	104
5.11.2 Dokumentacja typów wyliczanych	105
5.11.2.1 flag	105
5.12 Dokumentacja pliku str_operacje.cpp	105
5.12.1 Dokumentacja funkcji	106
5.12.1.1 operator<	106
5.12.1.2 operator<=	106
5.12.1.3 operator==	106
5.12.1.4 operator>	106
5.12.1.5 operator>=	106
5.13 Dokumentacja pliku str_operacje.hh	107
5.13.1 Dokumentacja funkcji	108
5.13.1.1 operator<	108
5.13.1.2 operator<=	108
5.13.1.3 operator==	108
5.13.1.4 operator>	108
5.13.1.5 operator>=	109
5.14 Dokumentacja pliku strona.dox	109
5.15 Dokumentacja pliku tablica_asocjacyjna.hh	109
5.15.1 Opis szczegółowy	110

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	27
h_table	29
kolejka_list	38
kolejka_tablica	40
m_sort	43
mnozenie	45
q_sort	55
stos_list	69
stos_tablica	72
tab_aso	75
drzewo< TYP >	21
el_tab< TYP >	25
hashtab< TYP >	32
operacje	48
queue_array< TYP >	57
queue_list< TYP >	61
stack_array< TYP >	63
stack_list< TYP >	67
tablica_asocjacyjna< TYP >	78
wzel< TYP >	82

Rozdział 2

Indeks klas

2.1 Lista klas

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

alorytm	Definicja klasy alorytm Jest to klasa bazowa, ktora ma za zadanie wczytac, przetworzyc i porownac dane z plikiem wzorcowym	7
bst	Modeluje drzewo binarne przeznaczone do testowania szybkosci wyszukiwnaia	18
drzewo< TYP >	Modeluje binarne drzewo przeszukiwan	21
el_tab< TYP >	Pojedynczy element tablicy haszujacej	25
h_sort	Klasa reprezentuje dane poddane sortowaniu przez kopcowanie	27
h_table	Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwnaia	29
hashtab< TYP >	Modeluje tablice haszujaca w oparciu o kontener klasy el_tab	32
kolejka_lista	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	38
kolejka_tablica	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury	40
m_sort	Klasa reprezentuje dane poddane sortowaniu przez scalanie	43
mnozenie	Modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2	45
operacje	Klasa modeluje tablice z danymi i metody sluzace do operacji na niej	48

q_sort	Klasa reprezentuje dane poddane sortowaniu szybkoemu	55
queue_array< TYP >	Modeluje kolejke w oparciu o tablice	57
queue_list< TYP >	Modeluje kolejke oparta na liscie STL	61
stack_array< TYP >	Modeluje stos w oparciu o tablice	63
stack_list< TYP >	Modeluje stos oparty na liscie STL	67
stos_lista	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	69
stos_tablica	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	72
tab_aso	Modeluje tablice asocjacyjna przeznaczona do testowania szybkosci wyszukiwania	75
tablica_asocjacyjna< TYP >	Klasa modeluje tablice asocjacyjna	78
wezel< TYP >	Modeluje pojedynczy wezel drzewa	82

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

alorytm.cpp	Plik zawiera definicje metod klas zdefiniowanych w pliku alorytm.hh	89
alorytm.hh	Definicja klas wykonujacych operacje na zestawie danych wejsciowych	89
drzewo.hh	91
hashtab.hh	93
kolejka.hh	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 się przepelni	94
main.cpp	Plik glowny	95
operacje.cpp	97
operacje.hh	97
statystyki.cpp	99
statystyki.hh	Plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk	100
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 swój rozmiar dwukrotnie, gdy stos się przepelni	103
str_operacje.cpp	105
str_operacje.hh	107
tablica_asocjacyjna.hh	109

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

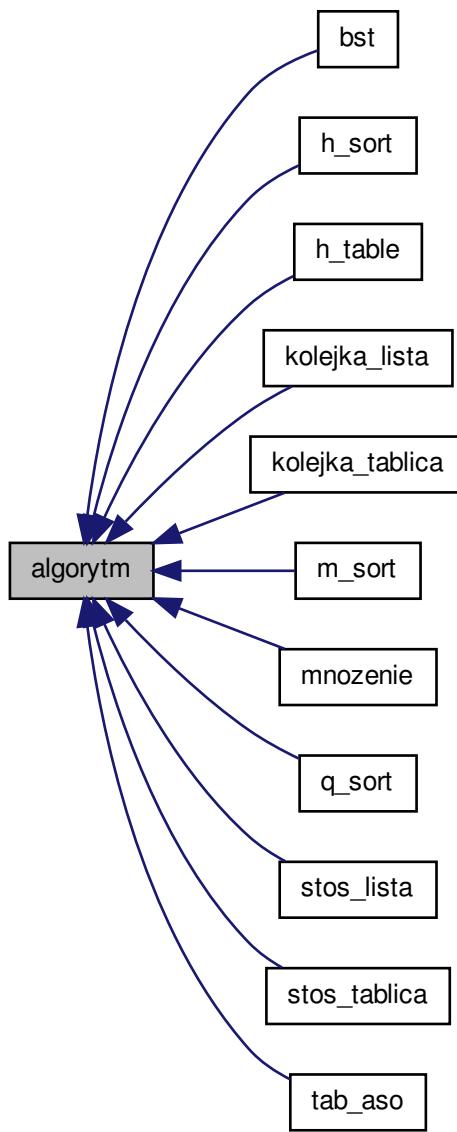
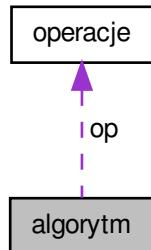


Diagram współpracy dla algorytm:



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.

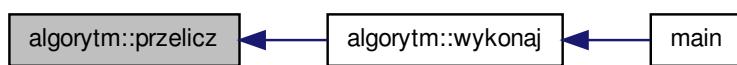
4.1.3.4 float algorytm::przelicz () [virtual]

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.

Oto graf wywoływań tej funkcji:

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

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

Parametry

in	plik	- strumien pliku wejściowego
----	------	------------------------------

Definicja w linii 10 pliku algorytm.cpp.

4.1.3.7 bool algorytm::wczytaj_wzor(ifstream & plik)

Metoda wczytuje plik wzorcowy do tablicy dane_wz.

Parametry

in	plik	- strumien pliku wejściowego
----	------	------------------------------

Definicja w linii 21 pliku algorytm.cpp.

4.1.3.8 void algorytm::wlacz_zegar()

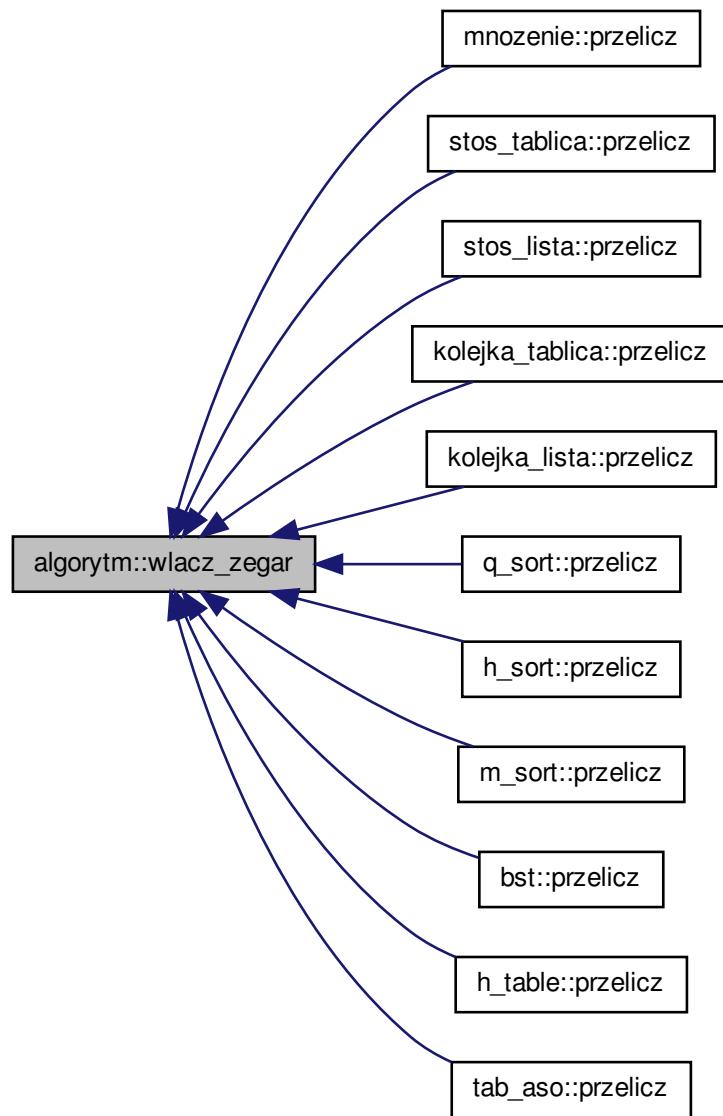
Metoda wlacza pomiar czasu poprzez wlaczenie funkcji gettimeofday i przechowanie czasu w zmiennej start.

Zwrota

start - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

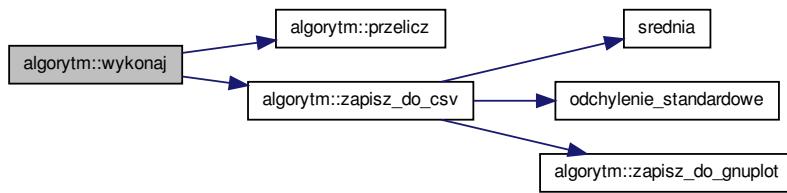


4.1.3.9 void algorytm::wykonaj (ostream & out)

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

Definicja w linii 78 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.1.3.10 void algorytm::wylacz_zegar ()

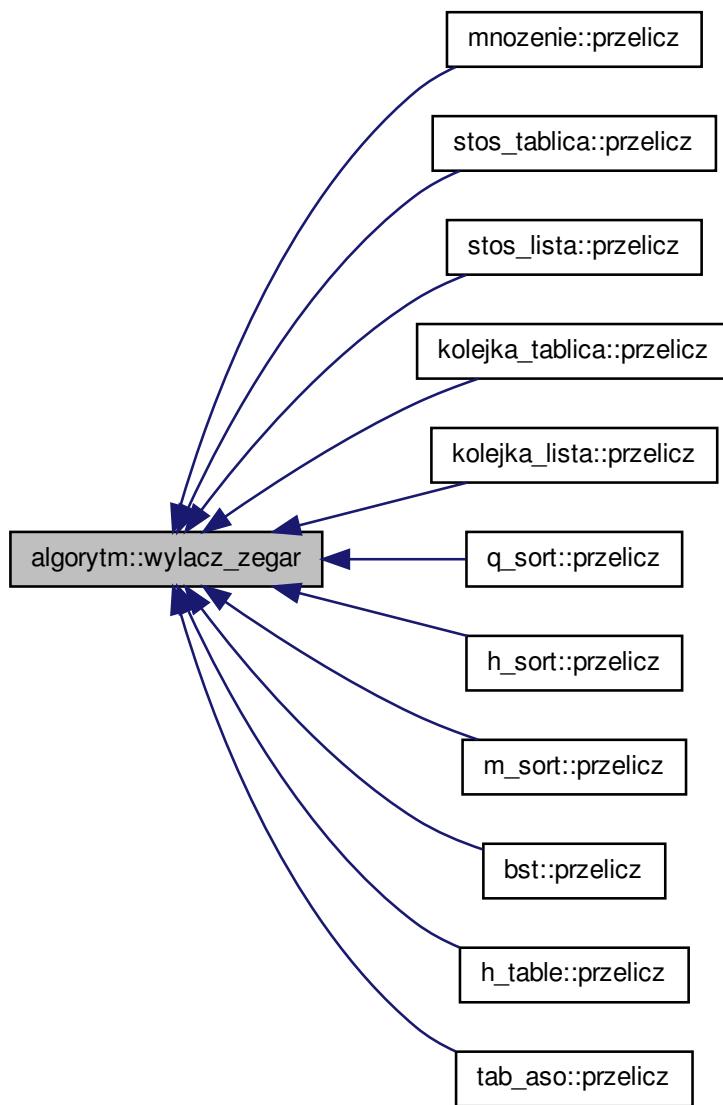
Metoda wyacza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowywanie czasu w zmiennej `end`.

Zwraca

end - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 49 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

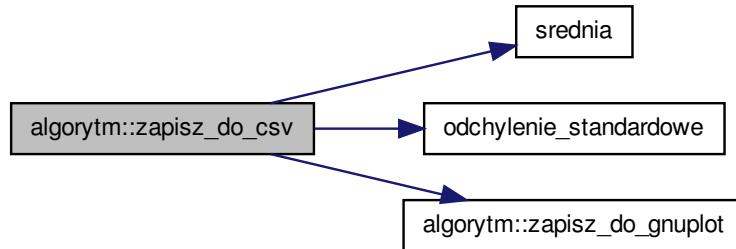


4.1.3.11 void algorytm::zapisz_do_csv (ostream & out)

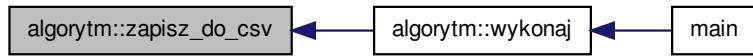
Metoda zapisuje tablice czas do pliku wyjscie.csv.

Definicja w linii 62 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

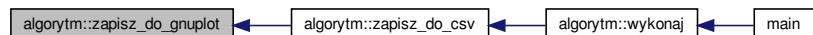


4.1.3.12 void algorytm::zapisz_do_gnuplot (ostream & out, float sr, float od)

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

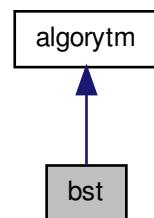
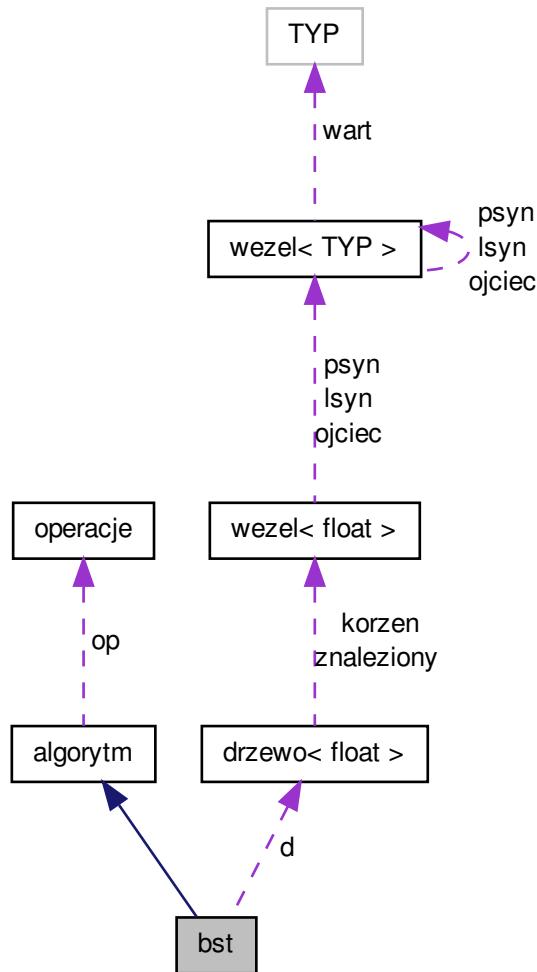


Diagram współpracy dla bst:



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

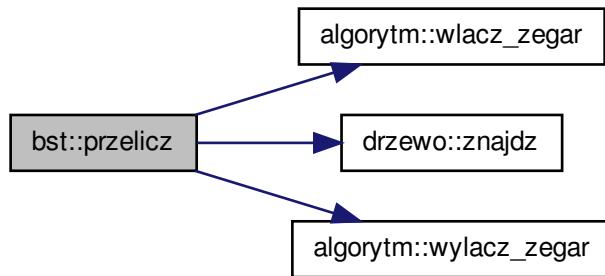
4.2.3.1 `float bst::przelicz()` [virtual]

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	<i>w</i>	- czyszczony wezel
----	----------	--------------------

Definicja w linii 180 pliku drzewo.hh.

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

dodaje wezel do drzewa

Parametry

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

Definicja w linii 91 pliku drzewo.hh.

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

dodaje wezel do drzewa

Parametry

in	<i>W</i>	- utworzony uprzednio wezel
----	----------	-----------------------------

Definicja w linii 83 pliku drzewo.hh.

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

sprawdza, czy w danym wezle znajduje sie szukany klucz

Parametry

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

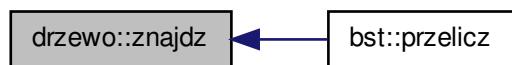
in	<i>k</i>	- klucz
----	----------	---------

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:

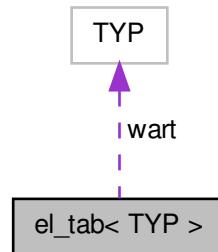
- drzewo.hh

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 h_sort

Klasa reprezentuje dane poddane sortowaniu przez kopcowanie

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

Diagram dziedziczenia dla h_sort

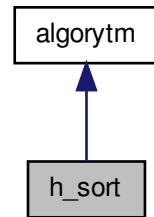
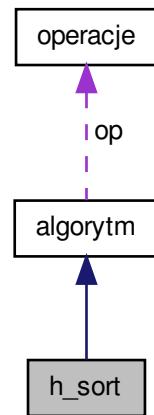


Diagram współpracy dla h_sort:



Metody publiczne

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

4.5.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

Definicja w linii 207 pliku algorytm.hh.

4.5.2 Dokumentacja konstruktora i destruktora

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

konstruktor klasy

Definicja w linii 210 pliku algorytm.hh.

4.5.3 Dokumentacja funkcji składowych

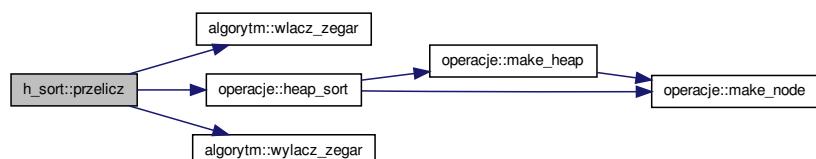
4.5.3.1 `float h_sort::przelicz() [virtual]`

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](#)

4.6 Dokumentacja klasy h_table

Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwania.

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

Diagram dziedziczenia dla h_table

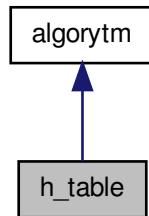
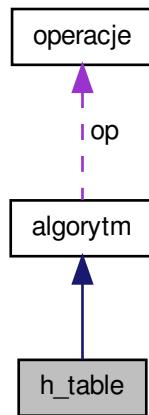


Diagram współpracy dla h_table:



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 algorytmem.

Atrybuty prywatne

- string * **klucze**

tablica kluczy

4.6.1 Opis szczegółowy

Modeluje tablice haszujaca przeznaczona do testowania szybkosci wyszukiwania.

Definicja w linii 246 pliku algorytm.hh.

4.6.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Definicja w linii 255 pliku algorytm.hh.

4.6.3 Dokumentacja funkcji składowych

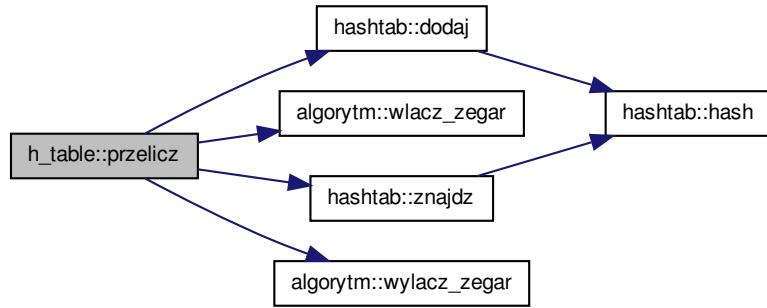
4.6.3.1 **float h_table::przelicz** () [virtual]

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:



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

wczytywanie kluczów

Parametry

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

Definicja w linii 213 pliku algorytm.cpp.

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 string* `h_table::klucze` [private]

tablica kluczów

Definicja w linii 248 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.7 Dokumentacja szablonu klasy `hashtab< TYP >`

modeluje tablice haszujące 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

4.7.1 Opis szczegółowy

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

modeluje tablice haszujaca w oparciu o kontener klasy `el_tab`

Definicja w linii 34 pliku hashtab.hh.

4.7.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 43 pliku hashtab.hh.

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

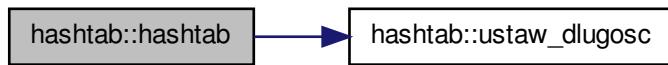
konsruktor parametryczny

Parametry

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

Definicja w linii 47 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



4.7.3 Dokumentacja funkcji składowych

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

metoda dodaje element do tablicy hasujacej

Definicja w linii 59 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



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

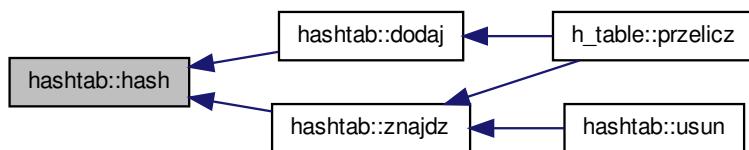
funkcja haszujaca

Zwraca

h - liczba, ktora po kompresji bedzie indeksem danego elementu

Definicja w linii 51 pliku `hashtab.hh`.

Oto graf wywoływań tej funkcji:

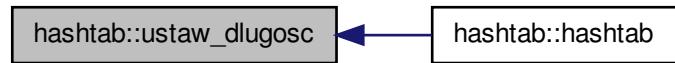


4.7.3.3 `template<typename TYP> void hashtab< TYP >::ustaw_dlugosc (int d)`
[inline]

ustawia dlugosc tablicy

Definicja w linii 41 pliku `hashtab.hh`.

Oto graf wywołań tej funkcji:



4.7.3.4 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:



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

Definicja w linii 93 pliku hashtable.hh.

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

metoda szuka zadanego elementu w oparciu o klucz

Parametry

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

Zwraca

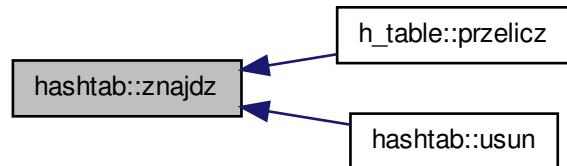
znaleziony element

Definicja w linii 74 pliku hashtab.hh.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.7.4 Dokumentacja atrybutów składowych

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

`dlugosc` tablicy

Definicja w linii 36 pliku hashtab.hh.

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

tablica haszująca

Definicja w linii 38 pliku hashtab.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [hashtab.hh](#)

4.8 Dokumentacja klasy kolejka_list

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka_list

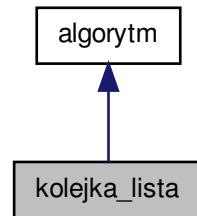
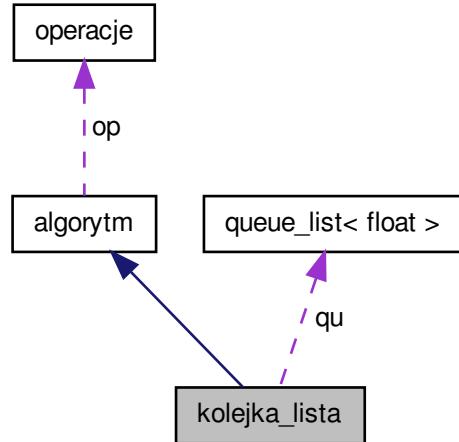


Diagram współpracy dla kolejka_list:



Metody publiczne

- `kolejka_lista` (`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`

4.8.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 190 pliku algorytm.hh.

4.8.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 193 pliku algorytm.hh.

4.8.3 Dokumentacja funkcji składowych

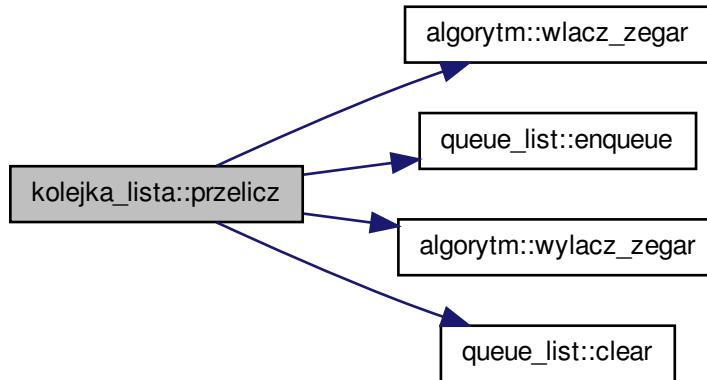
4.8.3.1 `float kolejka_lista::przelicz () [virtual]`

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:



4.8.4 Dokumentacja atrybutów składowych

4.8.4.1 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](#)

4.9 Dokumentacja klasy kolejka_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka_tablica

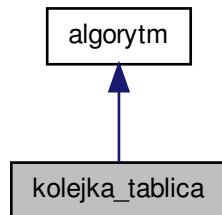
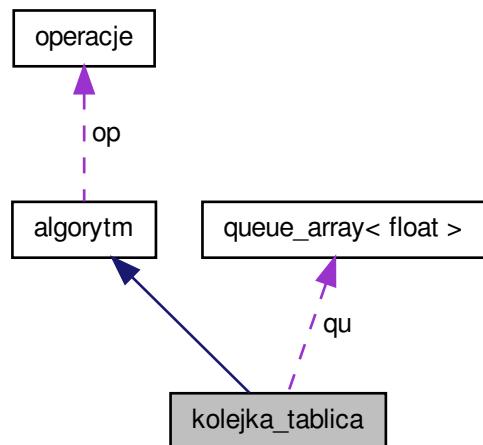


Diagram współpracy dla kolejka_tablica:



Metody publiczne

- `kolejka_tablica` (ifstream &plik1, ifstream &plik2, int N, int M, flag F)
konstruktor - ustawia flagę w zadanym stanie
- float `przelicz ()`
Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- `queue_array< float > qu`

4.9.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 178 pliku algorytm.hh.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 `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.

4.9.3 Dokumentacja funkcji składowych

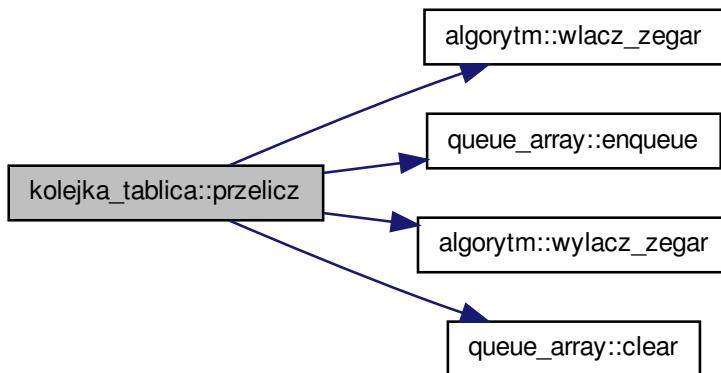
4.9.3.1 `float kolejka_tablica::przelicz () [virtual]`

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:



4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 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](#)

4.10 Dokumentacja klasy m_sort

Klasa reprezentuje dane poddane sortowaniu przez scalanie

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

Diagram dziedziczenia dla m_sort

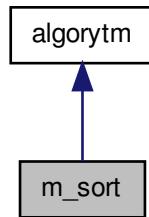
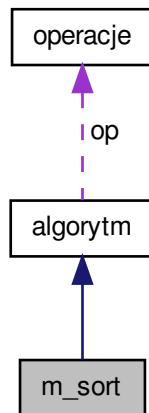


Diagram współpracy dla m_sort:



Metody publiczne

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

4.10.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez scalanie

Definicja w linii 216 pliku algorytm.hh.

4.10.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Definicja w linii 219 pliku algorytm.hh.

4.10.3 Dokumentacja funkcji składowych

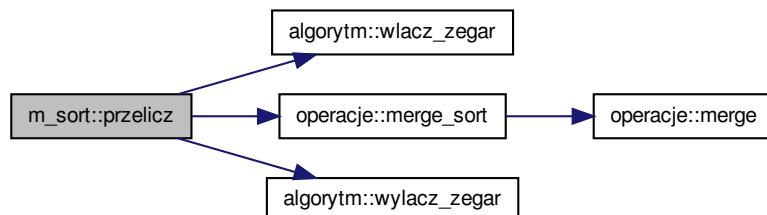
4.10.3.1 `float m_sort::przelicz() [virtual]`

metoda dokonujaca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 189 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.11 Dokumentacja klasy mnozenie

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2

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

Diagram dziedziczenia dla mnozenie

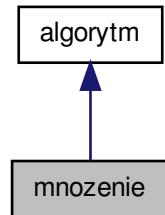
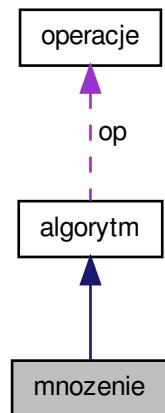


Diagram współpracy dla mnozenie:



Metody publiczne

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

wykonuje założony algorytm mnożenia elementów tablicy przez 2

4.11.1 Opis szczegółowy

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

4.11.2 Dokumentacja konstruktora i destruktora

4.11.2.1 **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.

4.11.3 Dokumentacja funkcji składowych

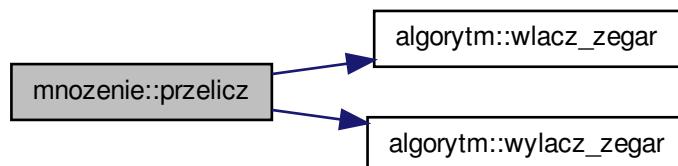
4.11.3.1 **float mnozenie::przelicz () [virtual]**

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](#)

4.12 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 sortownia 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 klas.
- **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

4.12.1 Opis szczegółowy

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

Definicja w linii 11 pliku operacje.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 operacje::operacje()

konstruktor bezparametryczny

4.12.2.2 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 właśnie rozmiarze
----	---	--

Definicja w linii 28 pliku operacje.hh.

4.12.3 Dokumentacja funkcji składowych

4.12.3.1 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.

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

metoda dodaje elementy do tablicy

Parametry

in	<i>tab2</i>	- tablica, ktora nalezy dolaczyc
in	<i>rozmiar</i>	- rozmiar tablicy tab2

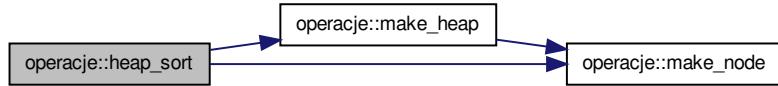
Definicja w linii 46 pliku operacje.cpp.

4.12.3.3 void operacje::heap_sort()

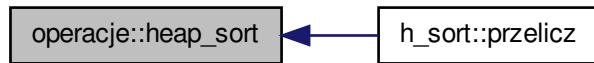
Metoda dokonuje sortowania po uprzednim utworzeniu kopca.

Definicja w linii 116 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

**4.12.3.4 void operacje::make_heap()**

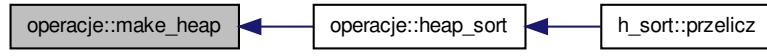
Metoda tworzy kopiec binarny.

Definicja w linii 110 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.12.3.5 void operacje::make_node (int rozmiar, int i)

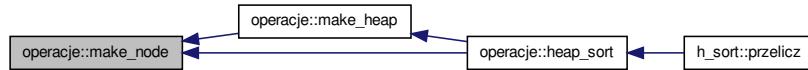
Metoda tworzy wezel drzewa, przypisując mu 2 synów, ustawiając ich w odpowiedniej kolejności (ojciec ma największą wartość)

Parametry

in	rozmiar	- rozmiar tablicy
in	i	- indeks elementu, do którego przypisujemy synów

Definicja w linii 95 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



4.12.3.6 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:



4.12.3.7 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

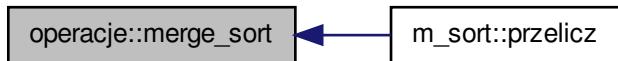
in	<i>poczatek</i>	- pierwszy indeks tablicy
in	<i>koniec</i>	- ostatni indeks tablicy

Definicja w linii 166 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.12.3.8 void operacje::odwroc_tablice()

metoda odwraca wszystkie elementy tablicy

Definicja w linii 12 pliku operacje.cpp.

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

Przeciażenie operatora przypisania; przypisuje elementy tablicy `tab1` do tablicy bieżącej polecem klasy.

Parametry

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

Definicja w linii 63 pliku operacje.cpp.

4.12.3.10 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.

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

Definicja w linii 88 pliku operacje.hh.

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

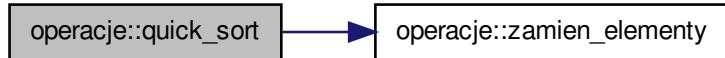
Metoda Dokonuje sortownia szybkiego.

Parametry

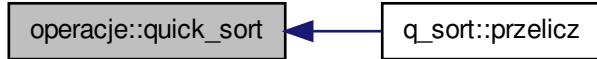
in	<i>l</i>	- pierwszy indeks tablicy
in	<i>p</i>	- ostatni indeks tablicy

Definicja w linii 77 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

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

Metoda zamienia 2 elementy tablicy.

Parametry

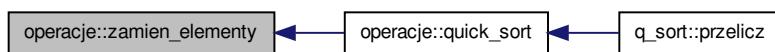
in	<i>i</i>	- element tablicy
in	<i>j</i>	- element tablicy

Zwraca

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

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



4.12.4 Dokumentacja atrybutów składowych

4.12.4.1 int operacje::n

ilosc elementow w tablicy

Definicja w linii 16 pliku operacje.hh.

4.12.4.2 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](#)

4.13 Dokumentacja klasy q_sort

Klasa reprezentuje dane poddane sortowaniu szybkiemu

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

Diagram dziedziczenia dla q_sort

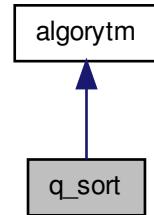
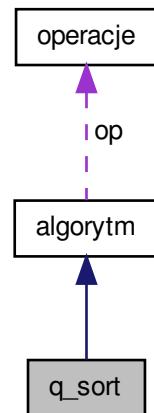


Diagram współpracy dla q_sort:



Metody publiczne

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

4.13.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkiemu

Definicja w linii 199 pliku algorytm.hh.

4.13.2 Dokumentacja konstruktora i destruktora

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

konstruktor klasy

Definicja w linii 202 pliku algorytm.hh.

4.13.3 Dokumentacja funkcji składowych

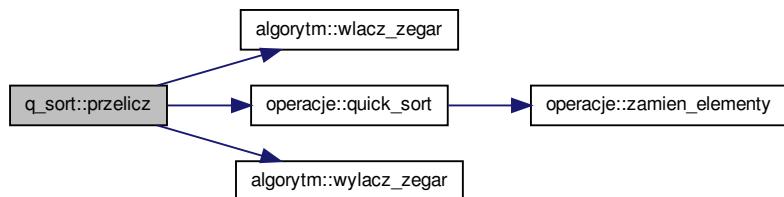
4.13.3.1 float q_sort::przelicz () [virtual]

metoda dokonująca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 171 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

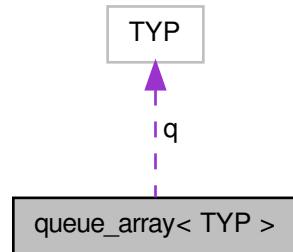
- [algorytm.hh](#)
- [algorytm.cpp](#)

4.14 Dokumentacja szablonu klasy queue_array< TYP >

Modeluje kolejkę 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**

4.14.1 Opis szczegółowy

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

Modeluje kolejkę w oparciu o tablice.

Definicja w linii 50 pliku kolejka.hh.

4.14.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 63 pliku kolejka.hh.

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

konstruktor parametryczny - ustawia flage na zadana pozycje

Definicja w linii 65 pliku kolejka.hh.

4.14.3 Dokumentacja funkcji składowych

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

czyszc kolejke

Definicja w linii 173 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



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

usuwa element z konca kolejki

Definicja w linii 129 pliku kolejka.hh.

4.14.3.3 `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:



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

Zwroca

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

Definicja w linii 75 pliku kolejka.hh.

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

Zwroca

rozmiar kolejki

Definicja w linii 70 pliku kolejka.hh.

4.14.4 Dokumentacja atrybutów składowych

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

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

Definicja w linii 59 pliku kolejka.hh.

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

Definicja w linii 51 pliku kolejka.hh.

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

Definicja w linii 52 pliku kolejka.hh.

4.14.4.4 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](#)

4.15 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](#)

4.15.1 Opis szczegółowy

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

Modeluje kolejkę opartą na liscie STL.

Definicja w linii 19 pliku kolejka.hh.

4.15.2 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 41 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



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

usuwa element

Definicja w linii 35 pliku kolejka.hh.

```
4.15.2.3 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:



4.15.2.4 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.

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

Zwraca

rozmiar kolejki

Definicja w linii 31 pliku kolejka.hh.

4.15.3 Dokumentacja atrybutów składowych

4.15.3.1 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:

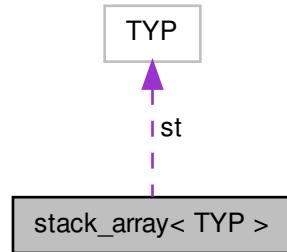
- [kolejka.hh](#)

4.16 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 ()**
czyszcza stos

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 * st**
- **int s**
- **int sp**

4.16.1 Opis szczegółowy

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

Modeluje stos w oparciu o tablice.

Definicja w linii 59 pliku stos.hh.

4.16.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 72 pliku stos.hh.

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

konstruktor parametryczny - ustawia flagę na zadana pozycję

Definicja w linii 74 pliku stos.hh.

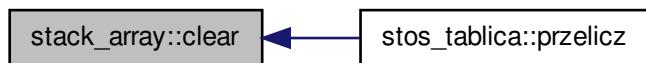
4.16.3 Dokumentacja funkcji składowych

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

czyszcza stos

Definicja w linii 181 pliku stos.hh.

Oto graf wywoływań tej funkcji:



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

Zwroca

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

Definicja w linii 79 pliku stos.hh.

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

zdejmuje element ze stosu

Definicja w linii 138 pliku stos.hh.

4.16.3.4 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:



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

Zwroca

rozmiar ztosu

Definicja w linii 87 pliku stos.hh.

4.16.4 Dokumentacja atrybutów składowych

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

flaga trybu zwiększenia pamięci , przyjmuje wartość :

plus1 - dla trybu kazdorazowego powiększania pamięci

x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 68 pliku stos.hh.

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

Definicja w linii 61 pliku stos.hh.

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

Definicja w linii 61 pliku stos.hh.

4.16.4.4 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](#)

4.17 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](#)

4.17.1 Opis szczegółowy

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

Modeluje stos oparty na liscie STL.

Definicja w linii 22 pliku stos.hh.

4.17.2 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 50 pliku stos.hh.

Oto graf wywoływań tej funkcji:



4.17.2.2 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.

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

zdejmuje element z wierzchu stosu

Definicja w linii 42 pliku stos.hh.

4.17.2.4 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:



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

Zwraca

rozmiar ztosu

Definicja w linii 34 pliku `stos.hh`.

4.17.3 Dokumentacja atrybutów składowych

4.17.3.1 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](#)

4.18 Dokumentacja klasy `stos_list`

Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla stos_listy

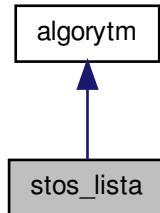
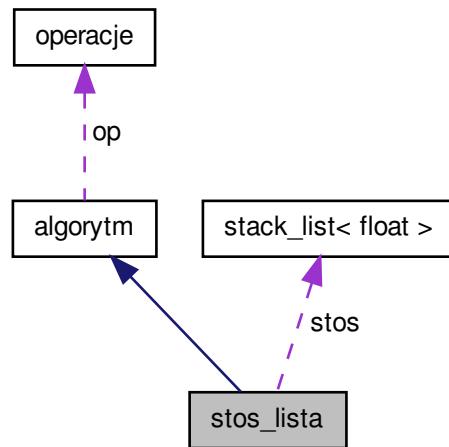


Diagram współpracy dla stos_listy:



Metody publiczne

- **stos_listy** (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`

4.18.1 Opis szczegółowy

Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 169 pliku algorytm.hh.

4.18.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 172 pliku algorytm.hh.

4.18.3 Dokumentacja funkcji składowych

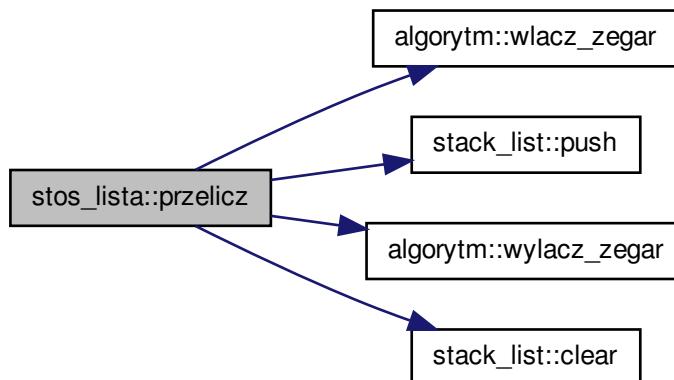
4.18.3.1 `float stos_lista::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 136 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.18.4 Dokumentacja atrybutów składowych

4.18.4.1 `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](#)

4.19 Dokumentacja klasy stos_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla stos_tablica

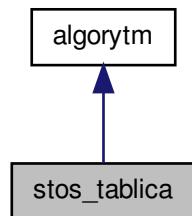
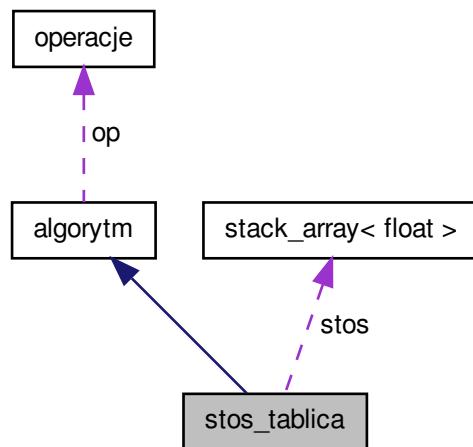


Diagram współpracy dla stos_tablica:



Metody publiczne

- **stos_tablica** (ifstream &plik1, ifstream &plik2, int N, int M, flag F)
konstruktor - ustawia flagę w zadanym stan
- **float przelicz ()**
Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorymem.

Atrybuty prywatne

- **stack_array< float > stos**

4.19.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 157 pliku algorytm.hh.

4.19.2 Dokumentacja konstruktora i destruktora

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

konstruktor - ustawia flagę w zadaną stan

Definicja w linii 163 pliku algorytm.hh.

4.19.3 Dokumentacja funkcji składowych

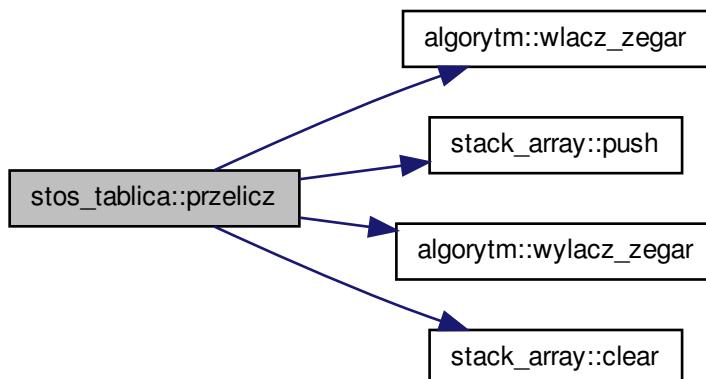
4.19.3.1 **float stos_tablica::przelicz() [virtual]**

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:



4.19.4 Dokumentacja atrybutów składowych

4.19.4.1 **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](#)

4.20 Dokumentacja klasy tab_aso

Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.

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

Diagram dziedziczenia dla tab_aso

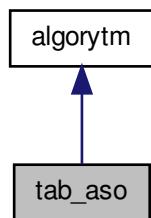
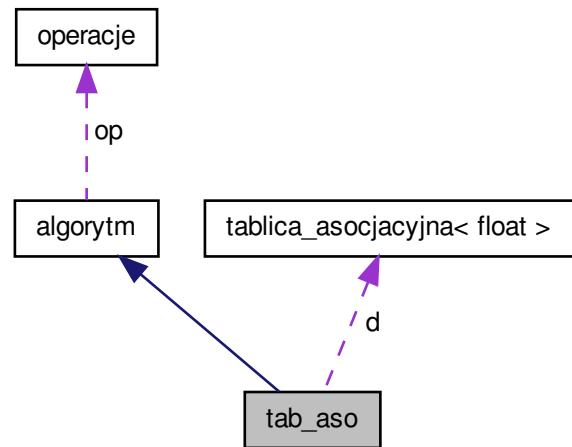


Diagram współpracy dla tab_aso:



Metody publiczne

- void [wczytaj_klucze](#) (ifstream &plik)
wczytywanie kluczy
- [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 kluczy

4.20.1 Opis szczegółowy

Modeluje tablice asocjacyjna przeznaczona do testowania szybkości wyszukiwania.

Definicja w linii 264 pliku algorytm.hh.

4.20.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Definicja w linii 277 pliku algorytm.hh.

4.20.3 Dokumentacja funkcji składowych

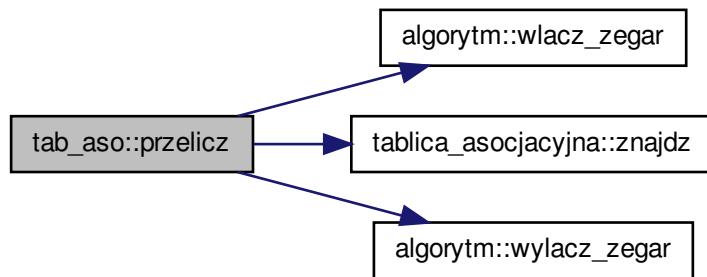
4.20.3.1 float [tab_aso::przelicz](#)() [virtual]

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:



4.20.3.2 void tab_aso::wczytaj_klucze (ifstream & plik)

wczytywanie klucz

Parametry

in	plik	- strumien z kluczmai uzytymi podczas testow
----	------	--

Definicja w linii 241 pliku algorytm.cpp.

4.20.4 Dokumentacja atrybutów składowych

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

tablica asocjacyjna

Definicja w linii 266 pliku algorytm.hh.

4.20.4.2 string* tab_aso::klucze [private]

wczytywanie klucz

Parametry

in	plik	- 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](#)

4.21 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 [wyapisz \(\)](#)

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 złożoność 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

4.21.1 Opis szczegółowy

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

Klasa modeluje tablice asocjacyjne.

Definicja w linii 18 pliku tablica_asocjacyjna.hh.

4.21.2 Dokumentacja konstruktora i destruktora

```
4.21.2.1 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 121 pliku tablica_asocjacyjna.hh.

4.21.3 Dokumentacja funkcji składowych

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

Zwraca

true, gdy stos jest pusty, false w przeciwnym wypadku

Definicja w linii 179 pliku tablica_asocjacyjna.hh.

4.21.3.2 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 127 pliku tablica_asocjacyjna.hh.

4.21.3.3 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 kluzem jest dodawana w srodek struktury, dane na prawo od niej przesuwane sa o jeden w prawo. Gdy istnieje potrzeba powiekszenia tablicy, stosuje sie znany juz radzaj gospodarowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.

Definicja w linii 34 pliku tablica_asocjacyjna.hh.

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

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

Zwrota

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

Definicja w linii 162 pliku tablica_asocjacyjna.hh.

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

Metoda usuwa zadany element, korzystajac z funkcji znajdz.

Definicja w linii 142 pliku tablica_asocjacyjna.hh.

4.21.3.6 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 82 pliku tablica_asocjacyjna.hh.

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

Definicja w linii 186 pliku tablica_asocjacyjna.hh.

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

Zwraca

ilosc elementow w strukturze

Definicja w linii 184 pliku tablica_asocjacyjna.hh.

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

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

Zwraca

indeks szukanego elementu

Definicja w linii 99 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:



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

Definicja w linii 172 pliku tablica_asocjacyjna.hh.

4.21.4 Dokumentacja atrybutów składowych

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

flaga informujaca o tym, czy dany klucz znaleziono w zbiorze

Definicja w linii 28 pliku tablica_asocjacyjna.hh.

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

Tablica zawierajaca klucze poszukiwan.

Definicja w linii 20 pliku tablica_asocjacyjna.hh.

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

rozmiar tablicy

Definicja w linii 24 pliku tablica_asocjacyjna.hh.

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

rozmiar danych zapelniajacych tablice

Definicja w linii 26 pliku tablica_asocjacyjna.hh.

4.21.4.5 **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:

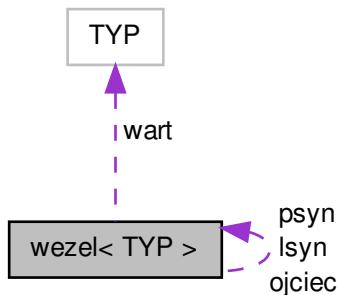
- [tablica_asocjacyjna.hh](#)

4.22 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

4.22.1 Opis szczegółowy

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

modeluje pojedynczy wezel drzewa

Definicja w linii 14 pliku drzewo.hh.

4.22.2 Dokumentacja konstruktora i destruktora

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

konstruktor bezparametryczny

Definicja w linii 29 pliku drzewo.hh.

4.22.2.2 `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.

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

destruktor

Definicja w linii 37 pliku drzewo.hh.

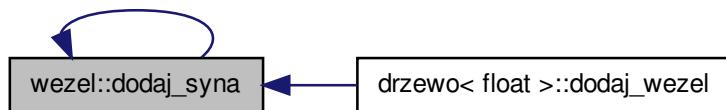
4.22.3 Dokumentacja funkcji składowych

4.22.3.1 `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:



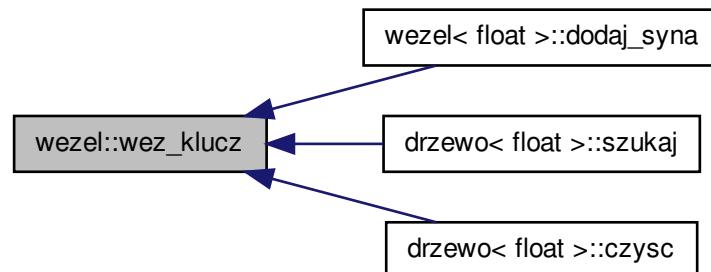
4.22.3.2 template<typename TYP> string wezel< TYP >::wez_klucz() [inline]

Zwraca

klucz wezla

Definicja w linii 41 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



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

Zwrota

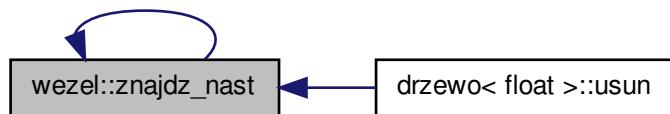
wartosc wezla

Definicja w linii 45 pliku drzewo.hh.

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

Definicja w linii 61 pliku drzewo.hh.

Oto graf wywoływań tej funkcji:



4.22.4 Dokumentacja atrybutów składowych

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

okresla, czyim synem jest wezel

Definicja w linii 21 pliku drzewo.hh.

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

klucz sluzacy do wyszukiwania

Definicja w linii 16 pliku drzewo.hh.

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

wskaznik na lewego syna wezla

Definicja w linii 25 pliku drzewo.hh.

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

wskaznik na ojca danego wezla

Definicja w linii 23 pliku drzewo.hh.

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

wskaznik na prawego syna wezla

Definicja w linii 27 pliku drzewo.hh.

4.22.4.6 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](#)

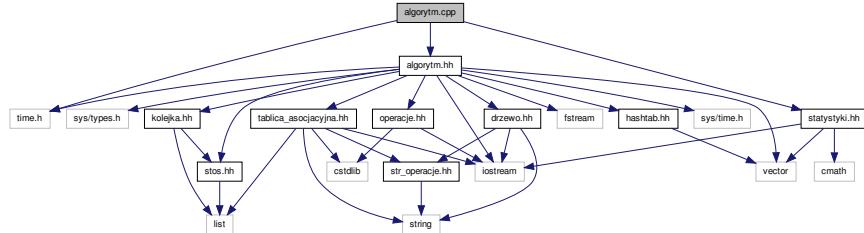
Rozdział 5

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](#)

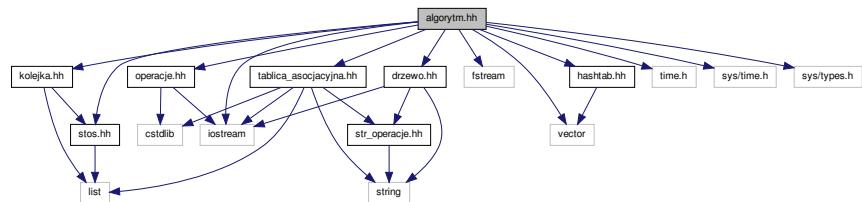
Definicja w pliku [algorytm.cpp](#).

5.2 Dokumentacja pliku algorytm.hh

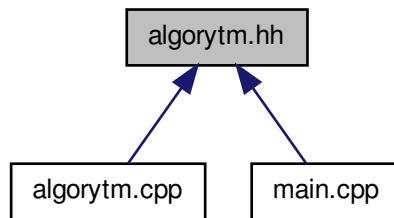
Definicja klas wykonujących operacje na zestawie danych wejściowych.

```
#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.

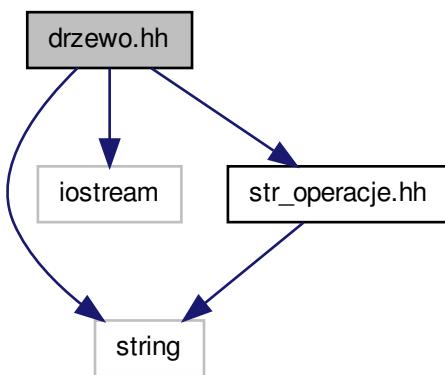
5.2.1 Opis szczegółowy

Definicja klas wykonujących operacje na zestawie danych wejściowych.

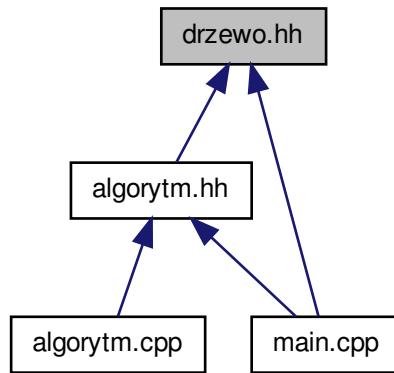
Definicja w pliku [algorytm.hh](#).

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

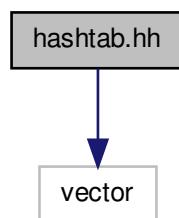
Wartości wyliczeń:

lewy
zaden
prawy

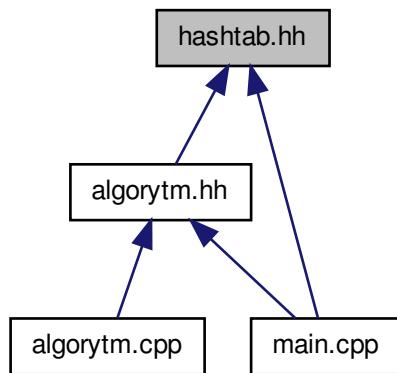
Definicja w linii 11 pliku drzewo.hh.

5.4 Dokumentacja pliku 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`

5.4.1 Opis szczegółowy

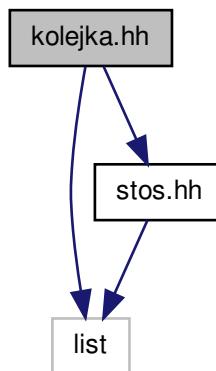
Plik zawiera definicje klasy reprezentujacej tablice haszujaca

Definicja w pliku `hashtab.hh`.

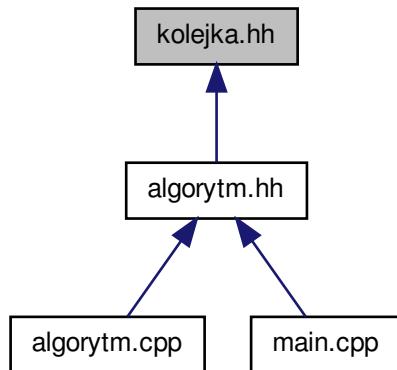
5.5 Dokumentacja pliku kolejka.hh

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

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



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



Komponenty

- class `queue_list< TYP >`
Modeluje kolejkę opartą na liscie STL.
- class `queue_array< TYP >`
Modeluje kolejkę w oparciu o tablice.

5.5.1 Opis szczegółowy

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

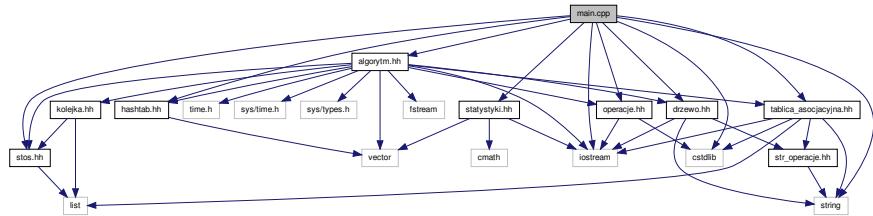
Definicja w pliku [kolejka.hh](#).

5.6 Dokumentacja pliku main.cpp

plik główny

```
#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 <cstdlib> #include <string> Wykres zależności załączania dla main.cpp:



Funkcje

- int [main \(\)](#)

5.6.1 Opis szczegółowy

plik główny

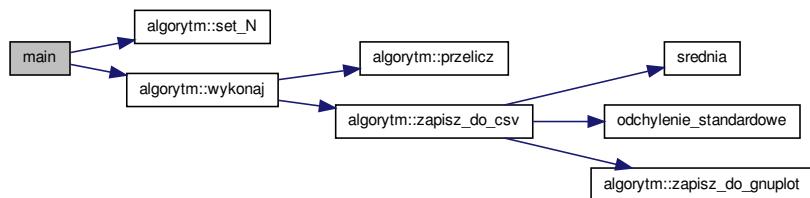
Definicja w pliku [main.cpp](#).

5.6.2 Dokumentacja funkcji

5.6.2.1 int [main \(\)](#)

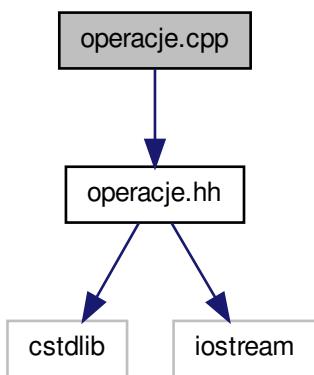
Definicja w linii 18 pliku [main.cpp](#).

Oto graf wywołań dla tej funkcji:



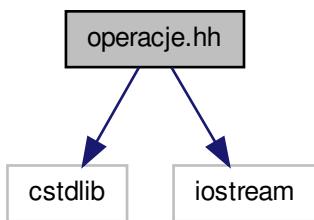
5.7 Dokumentacja pliku operacje.cpp

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

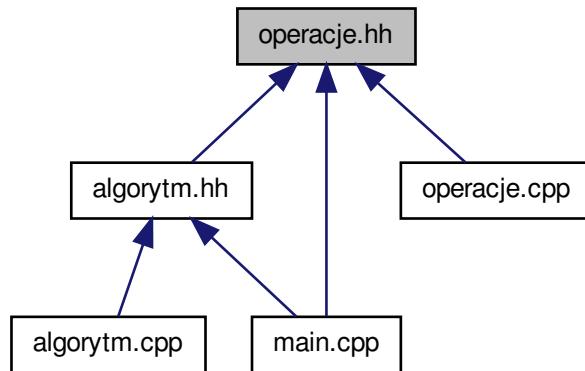


5.8 Dokumentacja pliku operacje.hh

#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

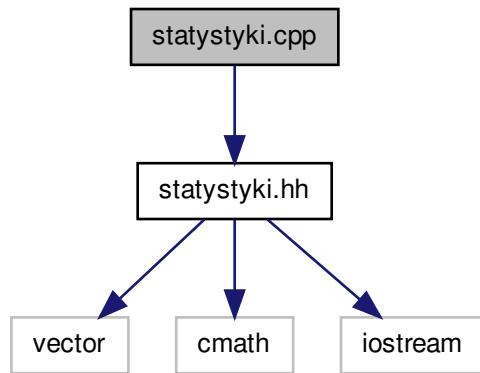
5.8.1 Dokumentacja definicji

5.8.1.1 #define ROZMIAR 9

Definicja w linii 3 pliku operacje.hh.

5.9 Dokumentacja pliku statystyki.cpp

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



Funkcje

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

5.9.1 Dokumentacja funkcji

5.9.1.1 float `odchylenie_standardowe` (float `srednia`, float * `tab`, int `rozmiar`)

funcja oblicza odchylenie standardowe

Parametry

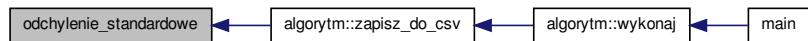
<code>tab</code>	- kontener zawierajacy czasy wykonania algorytmu
<code>srednia</code>	- wartosc srednia
<code>rozmiar</code>	- rozmiar tablicy

Zwroca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.9.1.2 float srednia (float * tab, int rozmiar)

funcja oblicza wartosc srednia

Parametry

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

Zwroca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

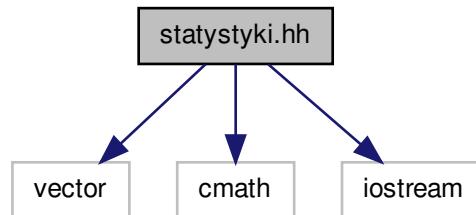
Oto graf wywoływań tej funkcji:



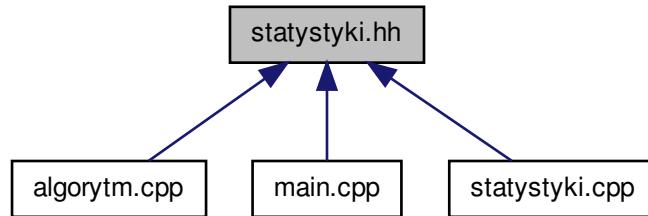
5.10 Dokumentacja pliku statystyki.hh

plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie 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

5.10.1 Opis szczegółowy

plik zawiera deklaracje funkcji odpowiedzialnych za prowadzenie statystyk

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

5.10.2 Dokumentacja funkcji

5.10.2.1 float odchylenie_standardowe (float srednia, float * tab, int rozmiar)

funckja oblicza odchylenie standardowe

Parametry

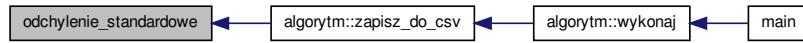
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.10.2.2 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.

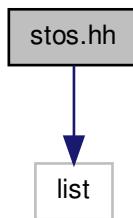
Oto graf wywoływań tej funkcji:



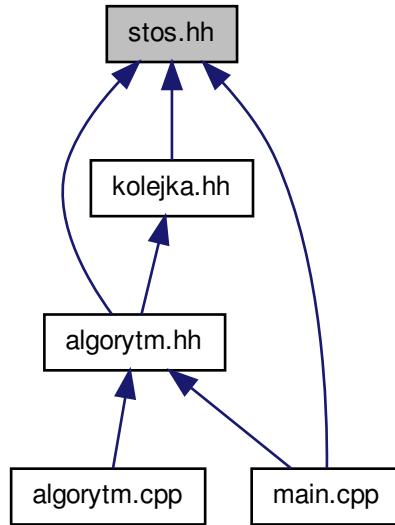
5.11 Dokumentacja pliku stos.hh

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

#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 zwiększenia pamięci

5.11.1 Opis szczegółowy

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

Definicja w pliku `stos.hh`.

5.11.2 Dokumentacja typów wyliczanych

5.11.2.1 enum flag

typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

Wartosci wyliczeń:

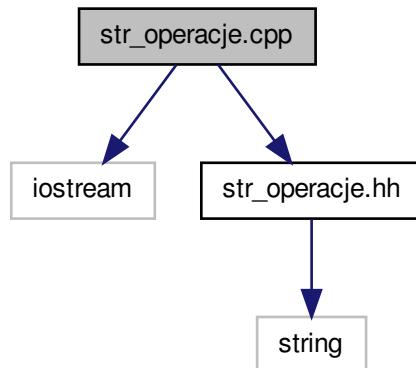
plus1

x2

Definicja w linii 17 pliku stos.hh.

5.12 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)

5.12.1 Dokumentacja funkcji

5.12.1.1 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.

5.12.1.2 bool operator<= (string s1, string s2)

Zwroca

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

Definicja w linii 26 pliku str_operacje.cpp.

5.12.1.3 bool operator== (string s1, string s2)

Zwroca

true, gdy lancuchy sa identyczne

Definicja w linii 34 pliku str_operacje.cpp.

5.12.1.4 bool operator> (string s1, string s2)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwroca

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

Definicja w linii 17 pliku str_operacje.cpp.

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

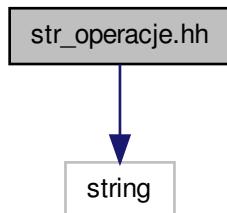
Zwroca

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

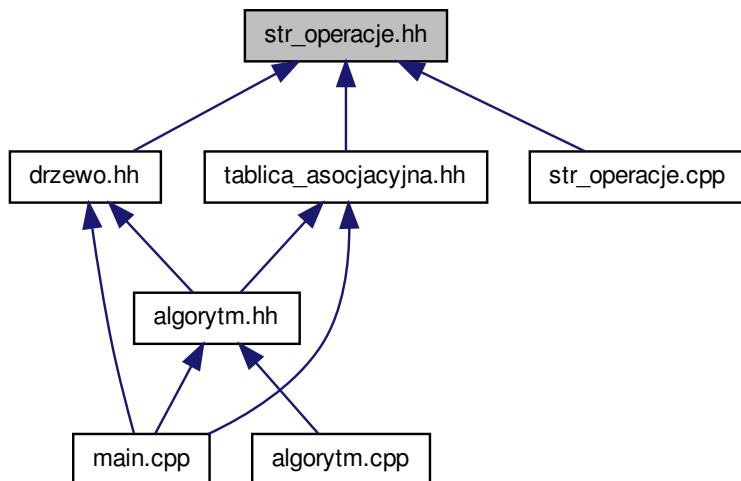
Definicja w linii 30 pliku str_operacje.cpp.

5.13 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)

5.13.1 Dokumentacja funkcji

5.13.1.1 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.

5.13.1.2 bool operator<= (string s1, string s2)

Zwroca

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

Definicja w linii 26 pliku str_operacje.cpp.

5.13.1.3 bool operator== (string s1, string s2)

Zwroca

true, gdy lancuchy sa identyczne

Definicja w linii 34 pliku str_operacje.cpp.

5.13.1.4 bool operator> (string s1, string s2)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwroca

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

Definicja w linii 17 pliku str_operacje.cpp.

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

Zwroca

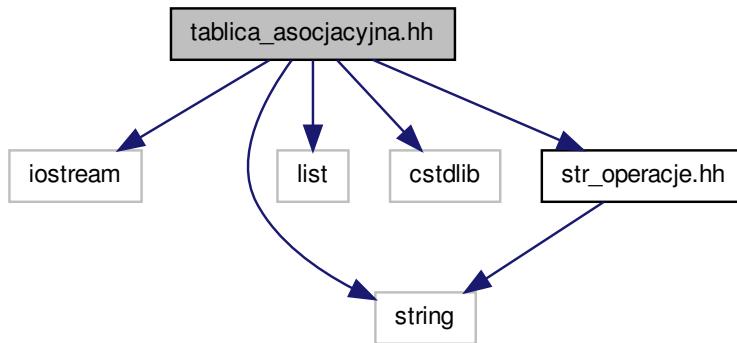
true, gdy s1 jest nizej w porzadku alfabetycznym niz s2 lub gdy oba stringi sa sobie rowne, false, gdy s1 jest wyzej w porzadku alfabetycznym niz s2

Definicja w linii 30 pliku str_operacje.cpp.

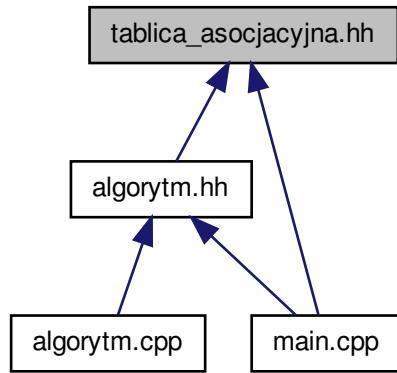
5.14 Dokumentacja pliku strona.dox

5.15 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.

5.15.1 Opis szczegółowy

Plik zawiera definicje klasy `tablica_asocjacyjna`, oraz definicje funkcji pomocniczych jako przeciażen operatorów porównania dla klasy typu string

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