

PAMSI - pwilkosz
1.0

Wygenerowano przez Doxygen 1.7.6.1

Sun Mar 23 2014 14:06:28

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	9
4.1.2 Dokumentacja konstruktora i destruktora	9
4.1.2.1 algorytm	9
4.1.3 Dokumentacja funkcji składowych	10
4.1.3.1 ile_danych	10
4.1.3.2 jaki_czas	10
4.1.3.3 porownaj	10
4.1.3.4 przelicz	10
4.1.3.5 set_N	11
4.1.3.6 wczytaj	11
4.1.3.7 wczytaj_wzor	11
4.1.3.8 wlacz zegar	11
4.1.3.9 wykonaj	12
4.1.3.10 wylacz zegar	13
4.1.3.11 zapisz_do_csv	14

4.1.3.12	zapisz_do_gnuplot	15
4.1.4	Dokumentacja atrybutów składowych	15
4.1.4.1	czas	16
4.1.4.2	dane	16
4.1.4.3	dane_wz	16
4.1.4.4	m	16
4.1.4.5	n	16
4.1.4.6	op	16
4.2	Dokumentacja klasy h_sort	16
4.2.1	Opis szczegółowy	18
4.2.2	Dokumentacja konstruktora i destruktor	18
4.2.2.1	h_sort	18
4.2.3	Dokumentacja funkcji składowych	18
4.2.3.1	przelicz	18
4.3	Dokumentacja klasy kolejka_lista	18
4.3.1	Opis szczegółowy	20
4.3.2	Dokumentacja konstruktora i destruktor	20
4.3.2.1	kolejka_lista	20
4.3.3	Dokumentacja funkcji składowych	20
4.3.3.1	przelicz	20
4.3.4	Dokumentacja atrybutów składowych	21
4.3.4.1	qu	21
4.4	Dokumentacja klasy kolejka_tablica	21
4.4.1	Opis szczegółowy	23
4.4.2	Dokumentacja konstruktora i destruktor	23
4.4.2.1	kolejka_tablica	23
4.4.3	Dokumentacja funkcji składowych	23
4.4.3.1	przelicz	23
4.4.4	Dokumentacja atrybutów składowych	24
4.4.4.1	qu	24
4.5	Dokumentacja klasy m_sort	24
4.5.1	Opis szczegółowy	26
4.5.2	Dokumentacja konstruktora i destruktor	26
4.5.2.1	m_sort	26

4.5.3	Dokumentacja funkcji składowych	26
4.5.3.1	przelicz	26
4.6	Dokumentacja klasy mnozenie	26
4.6.1	Opis szczegółowy	28
4.6.2	Dokumentacja konstruktora i destruktora	28
4.6.2.1	mnozenie	28
4.6.3	Dokumentacja funkcji składowych	28
4.6.3.1	przelicz	28
4.7	Dokumentacja klasy operacje	29
4.7.1	Opis szczegółowy	30
4.7.2	Dokumentacja konstruktora i destruktora	30
4.7.2.1	operacje	30
4.7.2.2	operacje	30
4.7.3	Dokumentacja funkcji składowych	30
4.7.3.1	dodaj_element	30
4.7.3.2	dodaj_elementy	30
4.7.3.3	heap_sort	31
4.7.3.4	make_heap	31
4.7.3.5	make_node	32
4.7.3.6	merge	33
4.7.3.7	merge_sort	33
4.7.3.8	odwroc_tablice	34
4.7.3.9	operator=	34
4.7.3.10	operator==	34
4.7.3.11	operator[]	34
4.7.3.12	quick_sort	35
4.7.3.13	zamien_elementy	35
4.7.4	Dokumentacja atrybutów składowych	36
4.7.4.1	n	36
4.7.4.2	tab	36
4.8	Dokumentacja klasy q_sort	36
4.8.1	Opis szczegółowy	38
4.8.2	Dokumentacja konstruktora i destruktora	38
4.8.2.1	q_sort	38

4.8.3	Dokumentacja funkcji składowych	38
4.8.3.1	przelicz	38
4.9	Dokumentacja szablonu klasy <code>queue_array< TYP ></code>	38
4.9.1	Opis szczegółowy	40
4.9.2	Dokumentacja konstruktora i destruktor	40
4.9.2.1	<code>queue_array</code>	40
4.9.2.2	<code>queue_array</code>	40
4.9.3	Dokumentacja funkcji składowych	40
4.9.3.1	<code>clear</code>	40
4.9.3.2	<code>dequeue</code>	41
4.9.3.3	<code>enqueue</code>	41
4.9.3.4	<code>is_empty</code>	41
4.9.3.5	<code>size</code>	41
4.9.4	Dokumentacja atrybutów składowych	41
4.9.4.1	<code>f</code>	42
4.9.4.2	<code>q</code>	42
4.9.4.3	<code>s</code>	42
4.9.4.4	<code>sp</code>	42
4.10	Dokumentacja szablonu klasy <code>queue_list< TYP ></code>	42
4.10.1	Opis szczegółowy	43
4.10.2	Dokumentacja funkcji składowych	43
4.10.2.1	<code>clear</code>	43
4.10.2.2	<code>dequeue</code>	43
4.10.2.3	<code>enqueue</code>	43
4.10.2.4	<code>is_empty</code>	44
4.10.2.5	<code>size</code>	44
4.10.3	Dokumentacja atrybutów składowych	44
4.10.3.1	<code>q</code>	44
4.11	Dokumentacja szablonu klasy <code>stack_array< TYP ></code>	44
4.11.1	Opis szczegółowy	46
4.11.2	Dokumentacja konstruktora i destruktor	46
4.11.2.1	<code>stack_array</code>	46
4.11.2.2	<code>stack_array</code>	46
4.11.3	Dokumentacja funkcji składowych	46

4.11.3.1	clear	46
4.11.3.2	is_empty	46
4.11.3.3	pop	47
4.11.3.4	push	47
4.11.3.5	size	47
4.11.4	Dokumentacja atrybutów składowych	47
4.11.4.1	f	47
4.11.4.2	s	48
4.11.4.3	sp	48
4.11.4.4	st	48
4.12	Dokumentacja szablonu klasy <code>stack_list< TYP ></code>	48
4.12.1	Opis szczegółowy	49
4.12.2	Dokumentacja funkcji składowych	49
4.12.2.1	clear	49
4.12.2.2	is_empty	49
4.12.2.3	pop	49
4.12.2.4	push	49
4.12.2.5	size	50
4.12.3	Dokumentacja atrybutów składowych	50
4.12.3.1	st	50
4.13	Dokumentacja klasy <code>stos_lista</code>	50
4.13.1	Opis szczegółowy	52
4.13.2	Dokumentacja konstruktora i destruktora	52
4.13.2.1	<code>stos_lista</code>	52
4.13.3	Dokumentacja funkcji składowych	52
4.13.3.1	przelicz	52
4.13.4	Dokumentacja atrybutów składowych	53
4.13.4.1	<code>stos</code>	53
4.14	Dokumentacja klasy <code>stos_tablica</code>	53
4.14.1	Opis szczegółowy	54
4.14.2	Dokumentacja konstruktora i destruktora	54
4.14.2.1	<code>stos_tablica</code>	55
4.14.3	Dokumentacja funkcji składowych	55
4.14.3.1	przelicz	55

4.14.4	Dokumentacja atrybutów składowych	55
4.14.4.1	stos	55
5	Dokumentacja plików	57
5.1	Dokumentacja pliku algorytm.cpp	57
5.1.1	Opis szczegółowy	57
5.2	Dokumentacja pliku algorytm.hh	57
5.2.1	Opis szczegółowy	59
5.3	Dokumentacja pliku kolejka.hh	59
5.3.1	Opis szczegółowy	60
5.4	Dokumentacja pliku main.cpp	60
5.4.1	Opis szczegółowy	61
5.4.2	Dokumentacja funkcji	61
5.4.2.1	main	61
5.5	Dokumentacja pliku operacje.cpp	62
5.6	Dokumentacja pliku operacje.hh	62
5.6.1	Dokumentacja definicji	63
5.6.1.1	ROZMIAR	63
5.7	Dokumentacja pliku statystyki.cpp	64
5.7.1	Dokumentacja funkcji	64
5.7.1.1	odchylenie_standardowe	64
5.7.1.2	srednia	65
5.8	Dokumentacja pliku statystyki.hh	65
5.8.1	Opis szczegółowy	66
5.8.2	Dokumentacja funkcji	67
5.8.2.1	odchylenie_standardowe	67
5.8.2.2	srednia	67
5.9	Dokumentacja pliku stos.hh	68
5.9.1	Opis szczegółowy	69
5.9.2	Dokumentacja typów wyliczanych	70
5.9.2.1	flag	70
5.10	Dokumentacja pliku strona.dox	70

Rozdział 1

Indeks klas

1.1 Hierarchia klas

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

algorytm	7
h_sort	16
kolejka_lista	18
kolejka_tablica	21
m_sort	24
mnozenie	26
q_sort	36
stos_lista	50
stos_tablica	53
operacje	29
queue_array< TYP >	38
queue_list< TYP >	42
stack_array< TYP >	44
stack_list< TYP >	48

Rozdział 2

Indeks klas

2.1 Lista klas

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

algorytm	Definicja klasy algorytm Jest to klasa bazowa, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym	7
h_sort	Klasa reprezentuje dane poddane sortowaniu przez kopcowanie . .	16
kolejka_lista	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	18
kolejka_tablica	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	21
m_sort	Klasa reprezentuje dane poddane sortowaniu przez scalanie	24
mnozenie	Modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2	26
operacje	Klasa modeluje tablice z danymi i metody służące do operacji na niej	29
q_sort	Klasa reprezentuje dane poddane sortowaniu szybkemu	36
queue_array< TYP >	Modeluje kolejkę w oparciu o tablice	38
queue_list< TYP >	Modeluje kolejkę opartą na liście STL	42
stack_array< TYP >	Modeluje stos w oparciu o tablice	44
stack_list< TYP >	Modeluje stos oparty na liście STL	48
stos_lista	Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury .	50

[stos_tablica](#)

Klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury . [53](#)

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

algorytm.cpp	Plik zawiera definicje metod klas zdefiniowanych w pliku algorytm.hh	57
algorytm.hh	Definicja klas wykonujących operacje na zestawie danych wejściowych	57
kolejka.hh	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	59
main.cpp	Plik główny	60
operacje.cpp		62
operacje.hh		62
statystyki.cpp		64
statystyki.hh	Plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk	65
stos.hh	Plik zawiera definicje klasy Stos Zaimplementowanej na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. każdorazowo powiększającej swój rozmiar b. powiększającej swój rozmiar dwukrotnie, gdy stos się przepelni	68

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy algorytm

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

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

Diagram dziedziczenia dla algorytm

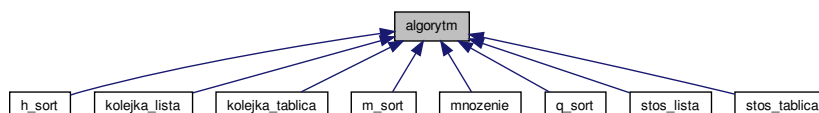
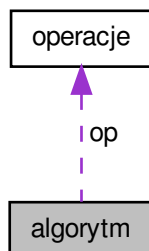


Diagram współpracy dla algorytm:



Metody publiczne

- **algorytm** (ifstream &plik1, ifstream &plik2, int N, int M)
konstruktor kopiujący - przekazuje informacje o nazwach plików, które zapisywane są do pol klasy
- void **wykonaj** (ofstream &out)
funkcja dokonuje operacji na pliku wejściowym, wywołuje metody odpowiedzialne za pomiar czasu oraz za porównanie wyniku operacji z plikiem wzorcowym
- bool **wczytaj** (ifstream &plik)
Metoda wczytuje plik wejściowy do tablicy dane oraz do obiektu op klasy operacje.
- void **set_N** (int wart)
metoda ustawia wartość n
- bool **wczytaj_wzor** (ifstream &plik)
Metoda wczytuje plik wzorcowy do tablicy dane_wz.
- virtual float **przelicz** ()
Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytmem.
- bool **porownaj** ()
porównuje przetworzone dane z danymi wzorcowymi
- int **ile_danych** ()
- float * **jaki_czas** ()
- float **wlacz zegar** ()
Metoda włącza pomiar czasu poprzez włączenie funkcji gettimeofday i przechowanie czasu w zmiennej start.
- float **wylacz zegar** ()
Metoda wyłącza pomiar czasu poprzez włączenie funkcji gettimeofday i przechowanie czasu w zmiennej end.

- void `zapisz_do_csv` (ofstream &out)
Metoda zapisuje tablice `czas` do pliku `wyjście.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 działania 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

4.1.1 Opis szczegółowy

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

Definicja w linii 33 pliku `algorytm.hh`.

4.1.2 Dokumentacja konstruktora i destruktoru

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

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

Parametry

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

Definicja w linii 75 pliku algorytm.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 `int algorytm::ile_danych ()`

Zwraca

ilosc liczb wejscowych

Definicja w linii 31 pliku algorytm.cpp.

4.1.3.2 `float * algorytm::jaki_czas ()`

Zwraca

tablica `czas` z danymi pomiarowymi czasu wykonywania algorytmu

Definicja w linii 34 pliku algorytm.cpp.

4.1.3.3 `bool algorytm::porownaj ()`

porownuje przetworzony dane z danymi wzorcowymi

Zwraca

true - gdy pliki zgodne false - w przeciwnym przypadku

Definicja w linii 89 pliku algorytm.cpp.

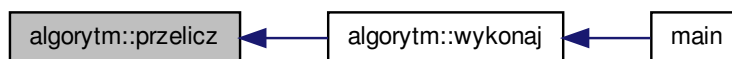
4.1.3.4 `float algorytm::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejscowych zgodnie z zadany algorytmem.

Reimplementowana w [m_sort](#), [h_sort](#), [q_sort](#), [kolejka_lista](#), [kolejka_tablica](#), [stos_lista](#), [stos_tablica](#) i [mnozenie](#).

Definicja w linii 9 pliku algorytm.cpp.

Oto graf wywołań tej funkcji:

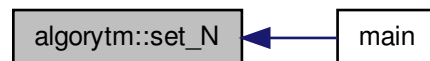


4.1.3.5 void algorytm::set_N (int wart) [inline]

metoda ustawia wartosc n

Definicja w linii 89 pliku algorytm.hh.

Oto graf wywoływań tej funkcji:



4.1.3.6 bool algorytm::wczytaj (ifstream &plik)

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

Parametry

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

Definicja w linii 10 pliku algorytm.cpp.

4.1.3.7 bool algorytm::wczytaj_wzor (ifstream &plik)

Metoda wczytuje plik wzorcowy do tablicy dane_wz.

Parametry

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

Definicja w linii 21 pliku algorytm.cpp.

4.1.3.8 float algorytm::wlacz zegar ()

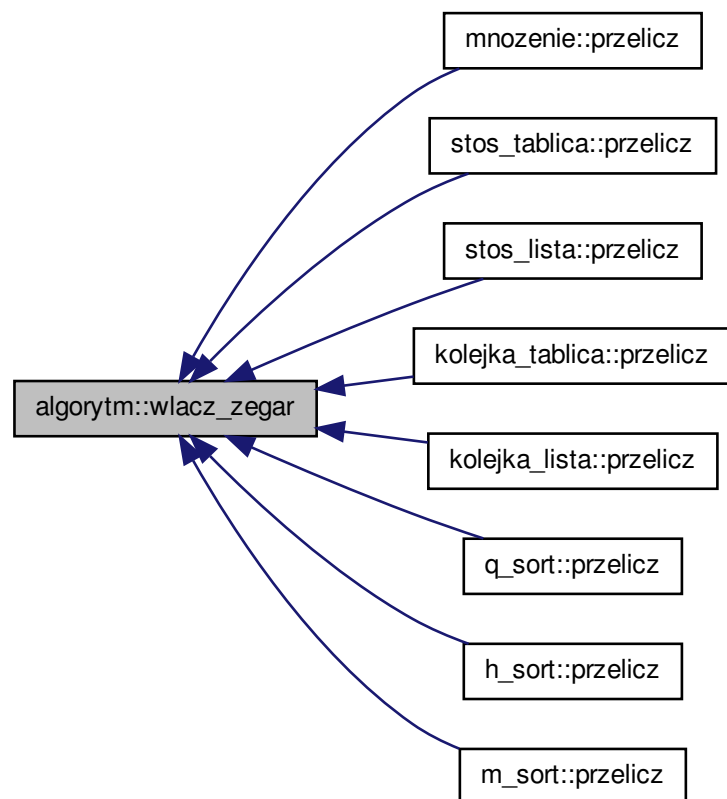
Metoda wlacza pomiar czasu poprzez wlaczenie funkcji gettimeofday i przechowanie czasu w zmiennej start.

Zwraca

start - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

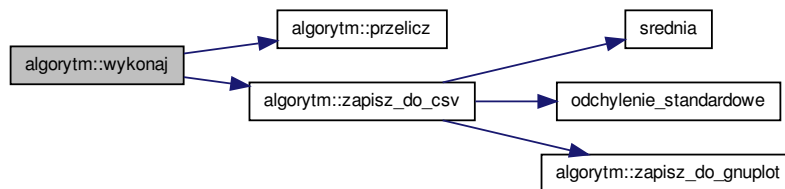
Oto graf wywoływania tej funkcji:

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

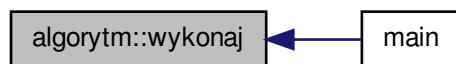
funkcja dokonuje operacji na pliku wejscowym, wywołuje metody odpowiedzialne za pomiar czasu oraz za porownanie wyniku operacji z plikiem wzorcowym

Definicja w linii 69 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.1.3.10 float algorytm::wylacz_zegar ()

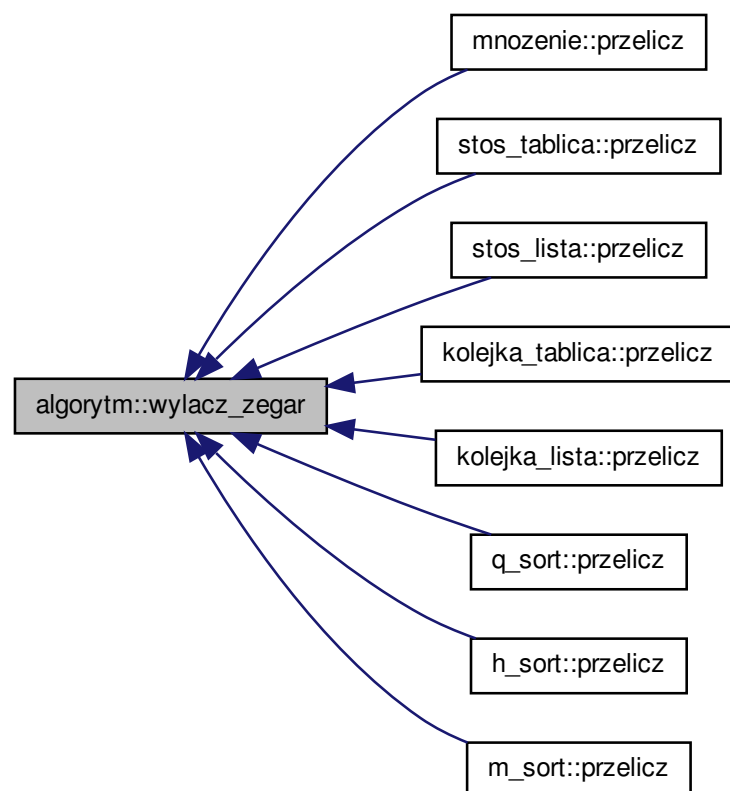
Metoda wyacza pomiar czasu poprzez włączenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `end`.

Zwraca

end - zmienna pamiętająca czas poprzedzający wykonanie algorytmu

Definicja w linii 46 pliku algorytm.cpp.

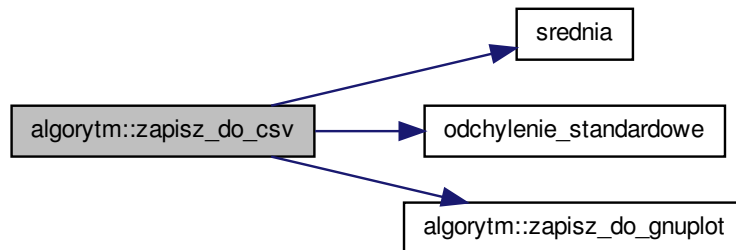
Oto graf wywołań tej funkcji:

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

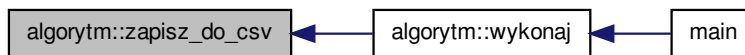
Metoda zapisuje tablice `czas` do pliku `wyjście.csv`.

Definicja w linii 54 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

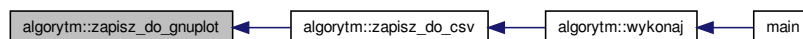


4.1.3.12 void algorytm::zapisz_do_gnuplot (ofstream & out, float sr, float od)

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

Definicja w linii 96 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 65 pliku `algorytm.hh`.

4.1.4.2 `float* algorytm::dane` `[protected]`

Tablica liczb wczytana z pliku.

Definicja w linii 41 pliku `algorytm.hh`.

4.1.4.3 `float* algorytm::dane_wz` `[protected]`

tablica liczb zawartych w pliku wzorcowym

Definicja w linii 46 pliku `algorytm.hh`.

4.1.4.4 `int algorytm::m` `[protected]`

ilosc powtorzen

Definicja w linii 56 pliku `algorytm.hh`.

4.1.4.5 `int algorytm::n` `[protected]`

ilosc danych w pliku

Definicja w linii 52 pliku `algorytm.hh`.

4.1.4.6 `operacje algorytm::op` `[protected]`

klasa zawierajaca tablice i metody do operacji na niej

Definicja w linii 60 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.2 Dokumentacja klasy `h_sort`

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

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


Diagram dziedziczenia dla h_sort

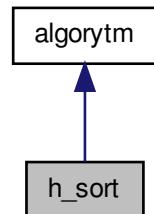
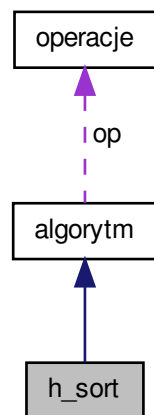


Diagram współpracy dla h_sort:



Metody publiczne

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

metoda dokonujaca sortowania danych

4.2.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez kopcowanie

Definicja w linii 203 pliku algorytm.hh.

4.2.2 Dokumentacja konstruktora i destruktor

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

konstruktor klasy

Definicja w linii 206 pliku algorytm.hh.

4.2.3 Dokumentacja funkcji składowych

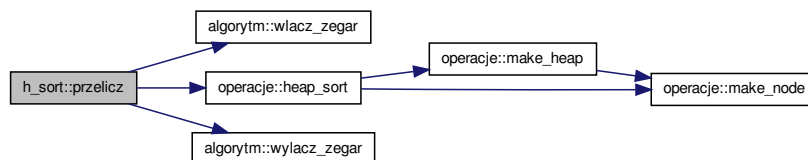
4.2.3.1 `float h_sort::przelicz ()` [virtual]

metoda dokonująca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 173 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.3 Dokumentacja klasy kolejka_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka_lista

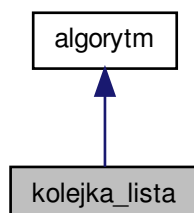
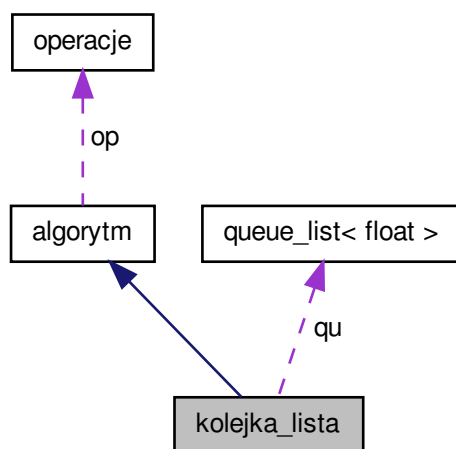


Diagram współpracy dla kolejka_lista:



Metody publiczne

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

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytm.

Atrybuty prywatne

- `queue_list< float > qu`

4.3.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 186 pliku algorytm.hh.

4.3.2 Dokumentacja konstruktora i destruktor

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

Definicja w linii 189 pliku algorytm.hh.

4.3.3 Dokumentacja funkcji składowych

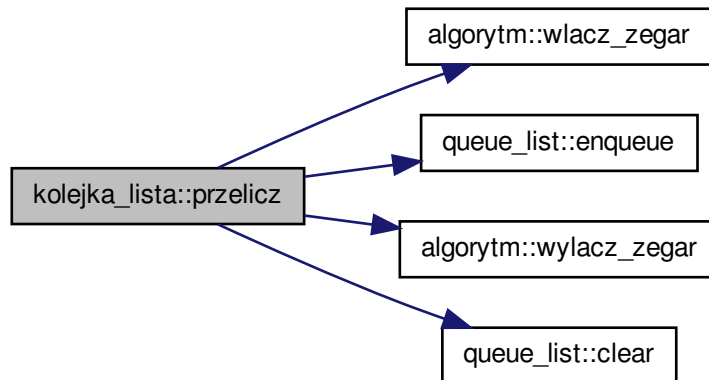
4.3.3.1 `float kolejka_lista::przelicz ()` [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadaniem algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 151 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.3.4 Dokumentacja atrybutów składowych

4.3.4.1 `queue_list<float> kolejka_lista::qu` [private]

Definicja w linii 187 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.4 Dokumentacja klasy kolejka_tablica

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla kolejka_tablica

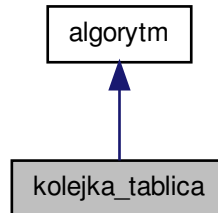
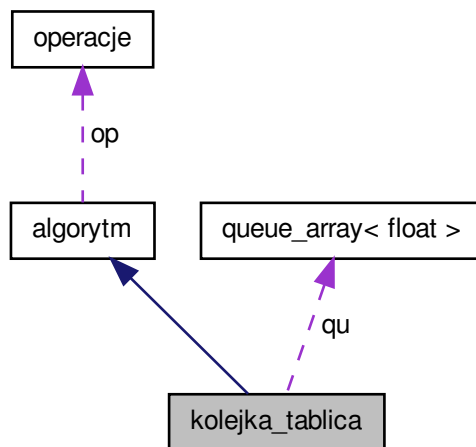


Diagram współpracy dla kolejka_tablica:



Metody publiczne

- `kolejka_tablica` (ifstream &plik1, ifstream &plik2, int N, int M, `flag` F)
konstruktor - ustawia flage w zadany stan
- float `przelicz` ()
Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadany algorytmem.

Atrybuty prywatne

- `queue_array< float > qu`

4.4.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 174 pliku algorytm.hh.

4.4.2 Dokumentacja konstruktora i destruktor

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

konstruktor - ustawia flage w zadany stan

Definicja w linii 180 pliku algorytm.hh.

4.4.3 Dokumentacja funkcji składowych

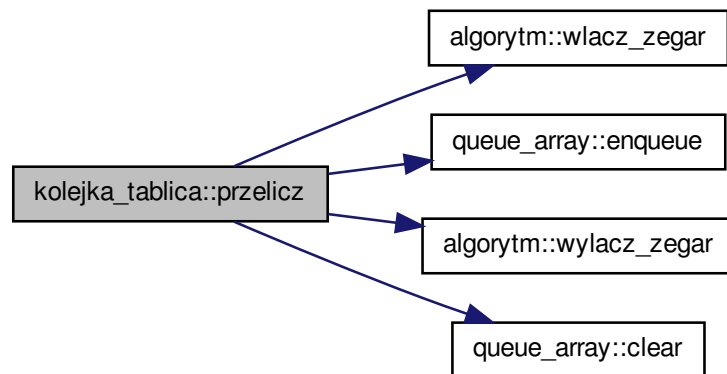
4.4.3.1 `float kolejka_tablica::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 138 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.4.4 Dokumentacja atrybutów składowych

4.4.4.1 `queue_array<float> kolejka_tablica::qu` `[private]`

Definicja w linii 175 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.5 Dokumentacja klasy `m_sort`

klasa reprezentuje dane poddane sortowaniu przez scalanie

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


Diagram dziedziczenia dla m_sort

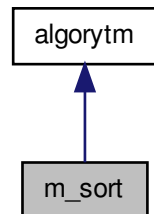
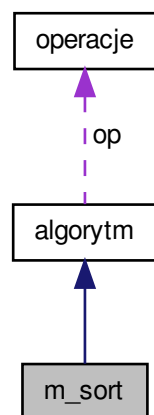


Diagram współpracy dla m_sort:



Metody publiczne

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

4.5.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu przez scalanie

Definicja w linii 212 pliku algorytm.hh.

4.5.2 Dokumentacja konstruktora i destruktor

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

konstruktor

Definicja w linii 215 pliku algorytm.hh.

4.5.3 Dokumentacja funkcji składowych

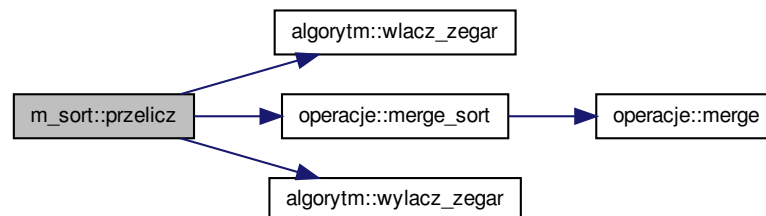
4.5.3.1 `float m_sort::przelicz () [virtual]`

metoda dokonujaca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 183 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 mnozenie

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejscowego przez 2

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

Diagram dziedziczenia dla mnozenie

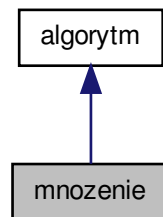
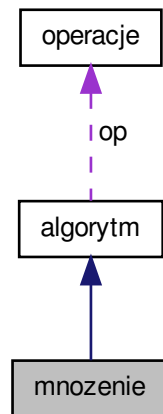


Diagram współpracy dla mnozenie:



Metody publiczne

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

wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

4.6.1 Opis szczegółowy

modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2
Definicja w linii 134 pliku algorytm.hh.

4.6.2 Dokumentacja konstruktora i destruktora

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

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

Parametry

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

Definicja w linii 143 pliku algorytm.hh.

4.6.3 Dokumentacja funkcji składowych

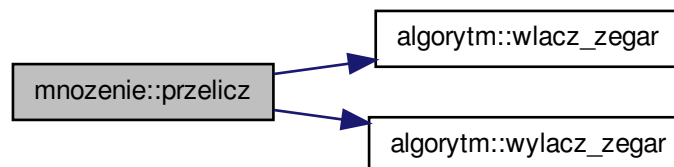
4.6.3.1 **float mnozenie::przelicz** () [virtual]

wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

Reimplementowana z [algorytm](#).

Definicja w linii 103 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.7 Dokumentacja klasy operacje

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

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

Metody publiczne

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

Atrybuty publiczne

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

4.7.1 Opis szczegółowy

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

Definicja w linii 9 pliku `operacje.hh`.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 `operacje::operacje ()`

konstruktor bezparametryczny

4.7.2.2 `operacje::operacje (int N) [inline]`

konstruktor parametryczny - alokuje pamiec w dynamicznej tablicy `tab`

Parametry

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

Definicja w linii 26 pliku `operacje.hh`.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 `void operacje::dodaj_element (float e)`

metoda dodaje element do tablicy, alokujac dodatkowa pamiec

Parametry

<code>in</code>	<code>e</code>	- element, ktory nalezy dolaczyc do tablicy
-----------------	----------------	---

Definicja w linii 27 pliku `operacje.cpp`.

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

metoda dodaje elementy do tablicy

Parametry

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

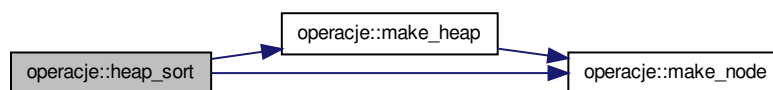
Definicja w linii 46 pliku operacje.cpp.

4.7.3.3 void operacje::heap_sort ()

Metoda dokonuje sortowania po uprzednim utworzeniu kopca.

Definicja w linii 112 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



4.7.3.4 void operacje::make_heap ()

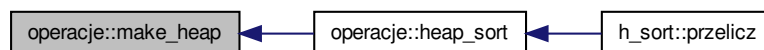
Metoda tworzy kopiec binarny.

Definicja w linii 106 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



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

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

Parametry

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

Definicja w linii 91 pliku `operacje.cpp`.

Oto graf wywoływań tej funkcji:



4.7.3.6 void operacje::merge (int poczatek, int srodek, int koniec)

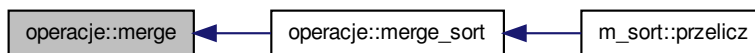
Metoda scala dwie czesci tablicy, jednocześnie je porzadkujac.

Parametry

in	<i>poczatek</i>	- pierwszy indeks tablicy
in	<i>srodek</i>	- srodkowy indeks tablicy
	<i>[ib]</i>	koniec - ostatni indeks tablicy

Definicja w linii 126 pliku operacje.cpp.

Oto graf wywoływania tej funkcji:



4.7.3.7 void operacje::merge_sort (int poczatek, int koniec)

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

Parametry

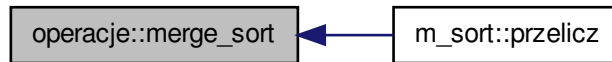
in	-	poczatek - pierwszy indeks tablicy
in	-	koniec - ostatni indeks tablicy

Definicja w linii 162 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywołań tej funkcji:



4.7.3.8 void operacje::odwroc_tablice ()

metoda odwraca wszystkie elementy tablicy

Definicja w linii 12 pliku operacje.cpp.

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

Przeciazenie operatora przypisania; przypisuje elementy tablicy *tab1* do tablicy będącej polem klasy.

Parametry

<i>in</i>	<i>tab1</i>	- tablica, ktorej zawartosc przypisujemy
-----------	-------------	--

Definicja w linii 63 pliku operacje.cpp.

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

Przeciazenie operatora porownania; metoda porownuje zawartosci dwuch tablic.

Parametry

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

Zwraca

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

Definicja w linii 69 pliku operacje.cpp.

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

Definicja w linii 86 pliku operacje.hh.

4.7.3.12 void operacje::quick_sort (int *i*, int *p*)

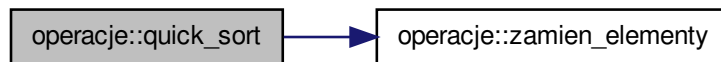
Metoda Dokonuje sortowania szybkiego.

Parametry

in	<i>i</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:

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

Metoda zamienia 2 elementy tablicy.

Parametry

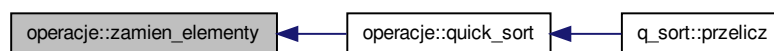
in	<i>i</i>	- element tablicy
in	<i>j</i>	- element tablicy

Zwraca

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

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywołań tej funkcji:



4.7.4 Dokumentacja atrybutów składowych

4.7.4.1 int operacje::n

ilosc elementow w tablicy

Definicja w linii 14 pliku operacje.hh.

4.7.4.2 float* operacje::tab

tablica z liczbami

Definicja w linii 17 pliku operacje.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.8 Dokumentacja klasy q_sort

klasa reprezentuje dane poddane sortowaniu szybkemu

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

Diagram dziedziczenia dla q_sort

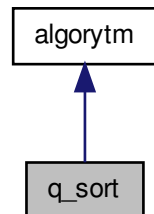
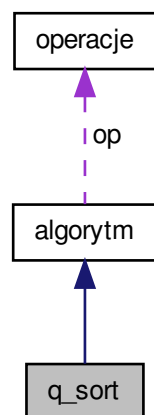


Diagram współpracy dla q_sort:



Metody publiczne

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

4.8.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkemu

Definicja w linii 195 pliku algorytm.hh.

4.8.2 Dokumentacja konstruktora i destruktor

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

konstruktor klasy

Definicja w linii 198 pliku algorytm.hh.

4.8.3 Dokumentacja funkcji składowych

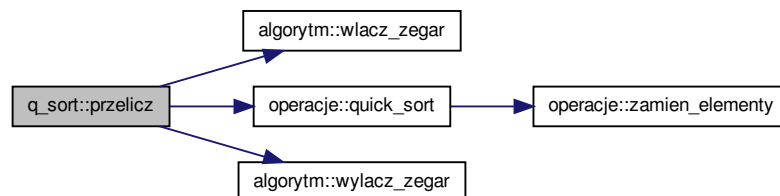
4.8.3.1 `float q_sort::przelicz ()` [virtual]

metoda dokonująca sortowania danych

Reimplementowana z [algorytm](#).

Definicja w linii 163 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

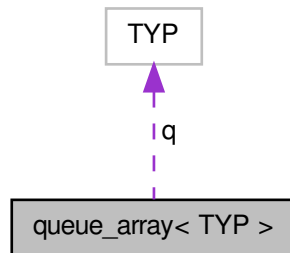
- [algorytm.hh](#)
- [algorytm.cpp](#)

4.9 Dokumentacja szablonu klasy `queue_array< TYP >`

Modeluje kolejke w oparciu o tablice.

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

Diagram współpracy dla queue_array< TYP >:



Metody publiczne

- `queue_array ()`
konstruktor bezparametryczny
- `queue_array (flag F)`
konstruktor parametryczny - ustawia flage na zadana pozycje
- `int size ()`
- `bool is_empty ()`
- `void enqueue (TYP &element)`
Dodaje element na poczatek kolejki w zaleznosci od wybranego trybu powiekszania tablicy.
- `TYP dequeue ()`
usuwa element z konca kolejki
- `void clear ()`
czysci kolejke

Atrybuty publiczne

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

Atrybuty prywatne

- `TYP * q`
- `int s`
- `int sp`

4.9.1 Opis szczegółowy

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

Modeluje kolejke w oparciu o tablice.

Definicja w linii 50 pliku kolejka.hh.

4.9.2 Dokumentacja konstruktora i destruktor

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

konstruktor bezparametryczny

Definicja w linii 63 pliku kolejka.hh.

```
4.9.2.2 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.

4.9.3 Dokumentacja funkcji składowych

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

czysci kolejke

Definicja w linii 170 pliku kolejka.hh.

Oto graf wywoływań tej funkcji:



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

usuwa element z konca kolejki

Definicja w linii 128 pliku `kolejka.hh`.

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

Dodaje element na początek kolejki w zależności od wybranego trybu powiększania tablicy.

Definicja w linii 82 pliku `kolejka.hh`.

Oto graf wywołań tej funkcji:



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

Zwraca

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

Definicja w linii 75 pliku `kolejka.hh`.

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

Zwraca

rozmiar kolejki

Definicja w linii 70 pliku `kolejka.hh`.

4.9.4 Dokumentacja atrybutów składowych

4.9.4.1 `template<typename TYP> flag queue_array< TYP >::f`

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

Definicja w linii 59 pliku kolejka.hh.

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

Definicja w linii 51 pliku kolejka.hh.

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

Definicja w linii 52 pliku kolejka.hh.

4.9.4.4 `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](#)

4.10 Dokumentacja szablonu klasy `queue_list< TYP >`

Modeluje kolejke oparta na liscie STL.

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

Metody publiczne

- bool `is_empty` ()
- int `size` ()
- void `enqueue` (TYP &element)
dodaje element
- TYP `dequeue` ()
usuwa element
- void `clear` ()
czysci stos

Atrybuty prywatne

- list< TYP > `q`

4.10.1 Opis szczegółowy

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

Modeluje kolejkę opartą na liście STL.

Definicja w linii 19 pliku `kolejka.hh`.

4.10.2 Dokumentacja funkcji składowych

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

czyszczy stos

Definicja w linii 41 pliku `kolejka.hh`.

Oto graf wywołań tej funkcji:



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

usuwa element

Definicja w linii 35 pliku `kolejka.hh`.

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

dodaje element

Definicja w linii 33 pliku `kolejka.hh`.

Oto graf wywołań tej funkcji:



4.10.2.4 `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.

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

Zwraca

rozmiar kolejki

Definicja w linii 31 pliku kolejka.hh.

4.10.3 Dokumentacja atrybutów składowych

4.10.3.1 `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:

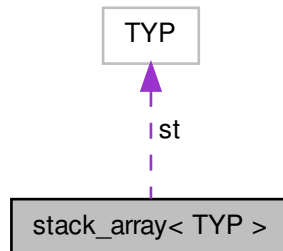
- [kolejka.hh](#)

4.11 Dokumentacja szablonu klasy `stack_array< TYP >`

Modeluje stos w oparciu o tablice.

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

Diagram współpracy dla `stack_array< TYP >`:



Metody publiczne

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

Atrybuty publiczne

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

Atrybuty prywatne

- `TYP * st`
- `int s`
- `int sp`

4.11.1 Opis szczegółowy

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

Modeluje stos w oparciu o tablice.

Definicja w linii 59 pliku stos.hh.

4.11.2 Dokumentacja konstruktora i destruktor

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

konstruktor bezparametryczny

Definicja w linii 72 pliku stos.hh.

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

konstruktor parametryczny - ustawia flage na zadana pozycje

Definicja w linii 74 pliku stos.hh.

4.11.3 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 178 pliku stos.hh.

Oto graf wywoływań tej funkcji:



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

Zwraca

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

Definicja w linii 79 pliku `stos.hh`.

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

zdejmuje element ze stosu

Definicja w linii 135 pliku `stos.hh`.

4.11.3.4 `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:



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

Zwraca

rozmiar ztosu

Definicja w linii 87 pliku `stos.hh`.

4.11.4 Dokumentacja atrybutów składowych

4.11.4.1 `template<typename TYP> flag stack_array< TYP >::f`

flaga trybu zwiekszania pamieci , przyjmuje wartosc :

plus1 - dla trybu kazdorazowego powiekszania pamieci

x2 - dla trybu podwajania rozmiaru struktury

Definicja w linii 68 pliku stos.hh.

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

Definicja w linii 61 pliku stos.hh.

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

Definicja w linii 61 pliku stos.hh.

4.11.4.4 `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](#)

4.12 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`

4.12.1 Opis szczegółowy

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

Modeluje stos oparty na liście STL.

Definicja w linii 22 pliku `stos.hh`.

4.12.2 Dokumentacja funkcji składowych

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

czyszczy stos

Definicja w linii 50 pliku `stos.hh`.

Oto graf wywołań tej funkcji:



4.12.2.2 `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`.

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

zdejmuje element z wierzchu stosu

Definicja w linii 42 pliku `stos.hh`.

4.12.2.4 `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 wywołań tej funkcji:



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

Zwraca

rozmiar ztosu

Definicja w linii 34 pliku stos.hh.

4.12.3 Dokumentacja atrybutów składowych

4.12.3.1 `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](#)

4.13 Dokumentacja klasy stos_lista

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla stos_lista

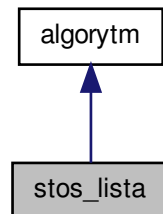
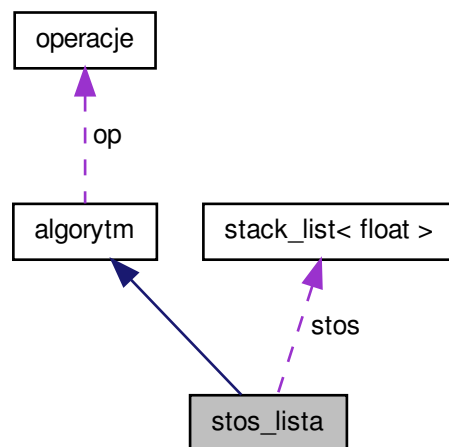


Diagram współpracy dla stos_lista:



Metody publiczne

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

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- `stack_list` < float > `stos`

4.13.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 165 pliku `algorytm.hh`.

4.13.2 Dokumentacja konstruktora i destruktor

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

Definicja w linii 168 pliku `algorytm.hh`.

4.13.3 Dokumentacja funkcji składowych

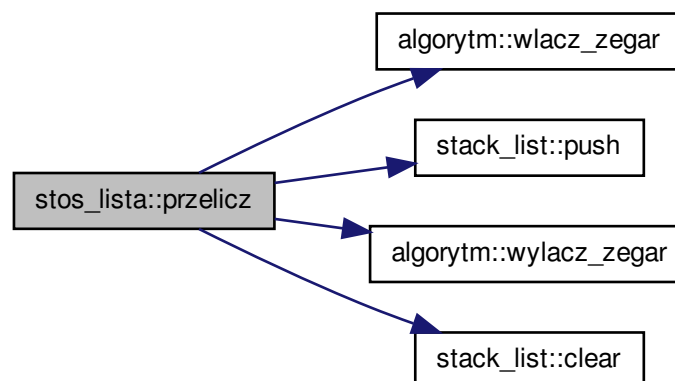
4.13.3.1 `float stos_lista::przelicz ()` [virtual]

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadaniem algorytmem.

Reimplementowana z `algorytm`.

Definicja w linii 126 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:



4.13.4 Dokumentacja atrybutów składowych

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

Definicja w linii 166 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

4.14 Dokumentacja klasy `stos_tablica`

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

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

Diagram dziedziczenia dla `stos_tablica`

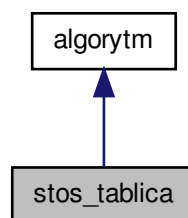
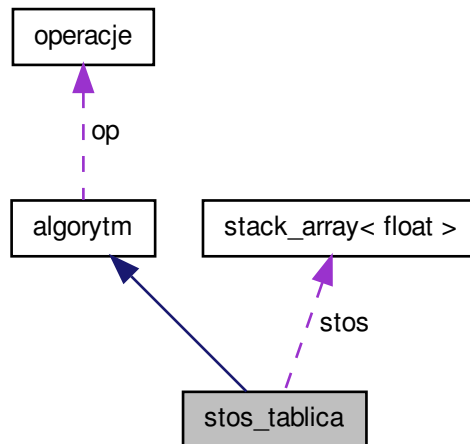


Diagram współpracy dla `stos_tablica`:



Metody publiczne

- `stos_tablica` (`ifstream &plik1`, `ifstream &plik2`, `int N`, `int M`, `flag F`)
konstruktor - ustawia flage w zadany stan
- `float przelicz ()`
Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadany algorytmem.

Atrybuty prywatne

- `stack_array< float > stos`

4.14.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 153 pliku `algorytm.hh`.

4.14.2 Dokumentacja konstruktora i destruktor

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

konstruktor - ustawia flage w zadany stan

Definicja w linii 159 pliku `algorytm.hh`.

4.14.3 Dokumentacja funkcji składowych

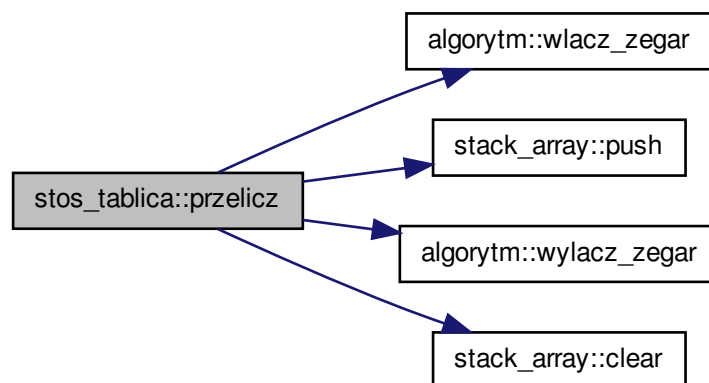
4.14.3.1 `float stos_tablica::przelicz () [virtual]`

Metoda odpowiada za przetworzenie danych wejściowych zgodnie z zadany algorytmem.

Reimplementowana z [algorytm](#).

Definicja w linii 114 pliku `algorytm.cpp`.

Oto graf wywołań dla tej funkcji:



4.14.4 Dokumentacja atrybutów składowych

4.14.4.1 `stack_array<float> stos_tablica::stos [private]`

Definicja w linii 154 pliku `algorytm.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [algorytm.hh](#)

- [algorytm.cpp](#)

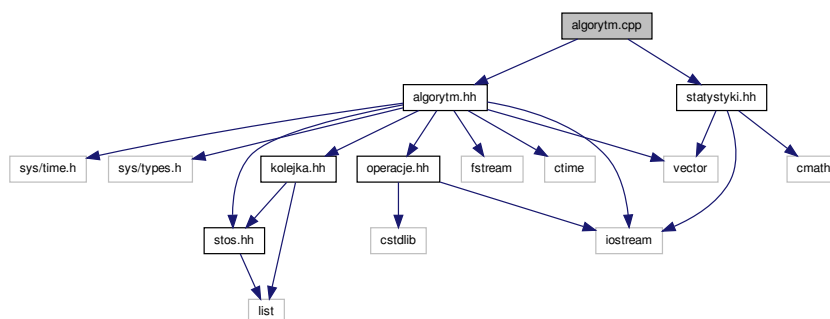
Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku algorytm.cpp

plik zawiera definicje metod klas zdefiniowanych w pliku [algorytm.hh](#)

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



5.1.1 Opis szczegółowy

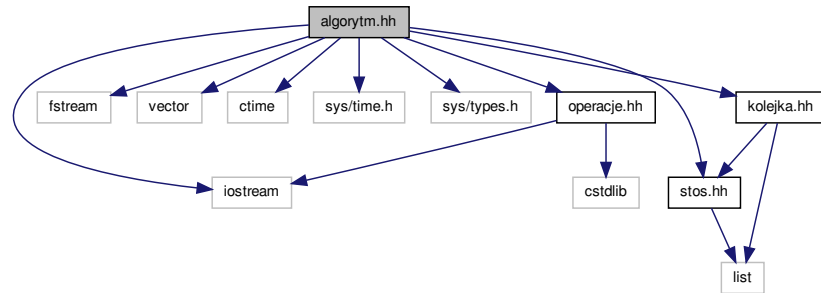
plik zawiera definicje metod klas zdefiniowanych w pliku [algorytm.hh](#)

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

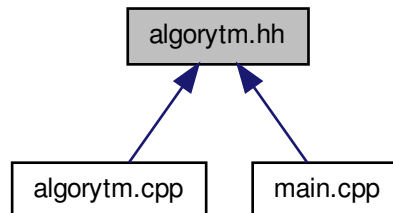
5.2 Dokumentacja pliku algorytm.hh

Definicja klas wykonujących operacje na zestawie danych wejściowych.

```
#include <iostream> #include <fstream> #include <vector> ×
#include <ctime> #include <sys/time.h> #include <sys/types.h>
#include "operacje.hh" #include "stos.hh" #include
"kolejka.hh" Wykres zależności załączania dla algorytm.hh:
```



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



Komponenty

- class [algorytm](#)
Definicja klasy algorytm Jest to klasa bazowa, która ma za zadanie wczytać, przetworzyć i porównać dane z plikiem wzorcowym.
- class [mnozenie](#)
modeluje algorytm dokonujący mnożenia każdego elementu pliku wejściowego przez 2
- class [stos_tablica](#)
klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

- class `stos_lista`
klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury
- class `kolejka_tablica`
klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury
- class `kolejka_lista`
klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury
- class `q_sort`
klasa reprezentuje dane poddane sortowaniu szybkemu
- class `h_sort`
klasa reprezentuje dane poddane sortowaniu przez kopcowanie
- class `m_sort`
klasa reprezentuje dane poddane sortowaniu przez scalanie

5.2.1 Opis szczegółowy

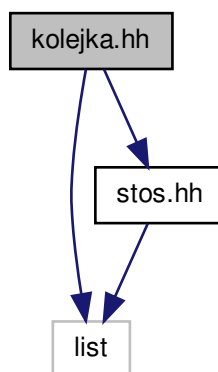
Definicja klas wykonujących operacje na zestawie danych wejściowych.

Definicja w pliku `algorytm.hh`.

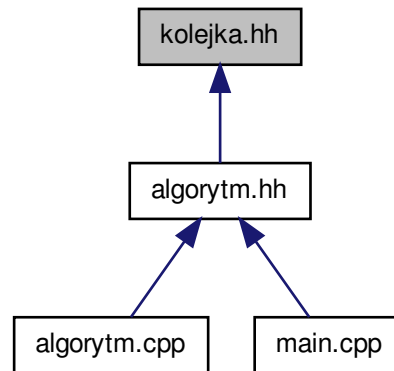
5.3 Dokumentacja pliku kolejka.hh

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.

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



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



Komponenty

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

5.3.1 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.

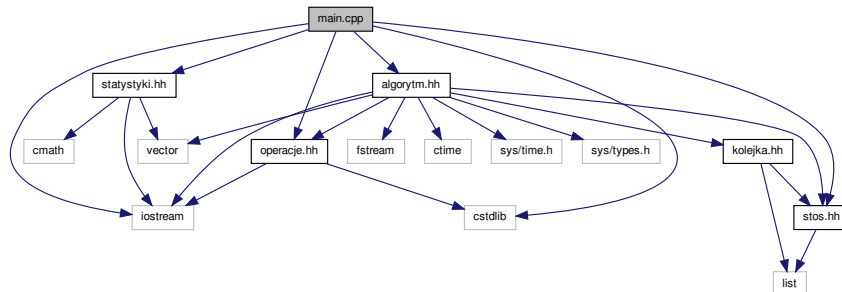
Definicja w pliku [kolejka.hh](#).

5.4 Dokumentacja pliku main.cpp

plik glowny

```
#include <iostream> #include "algorytm.hh" #include "statystyki.-  
hh" #include "operacje.hh" #include "stos.hh" #include
```

<cstdlib> Wykres zależności załączania dla main.cpp:



Funkcje

- int [main](#) ()

5.4.1 Opis szczegółowy

plik glowny

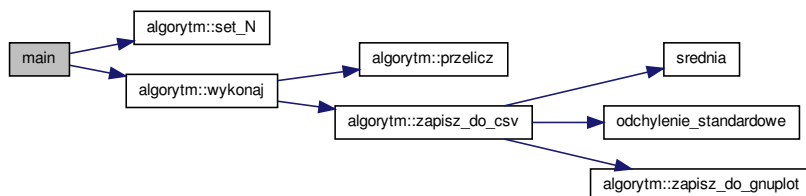
Definicja w pliku [main.cpp](#).

5.4.2 Dokumentacja funkcji

5.4.2.1 int main ()

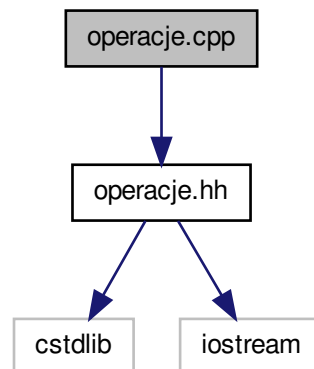
Definicja w linii 13 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



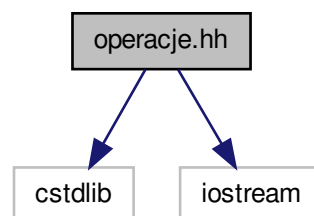
5.5 Dokumentacja pliku operacje.cpp

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

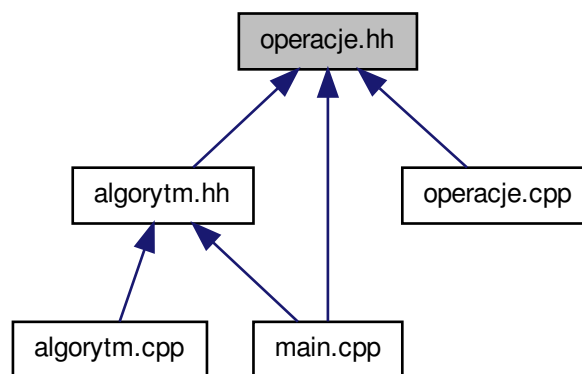


5.6 Dokumentacja pliku operacje.hh

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



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



Komponenty

- class `operacje`

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

Definicje

- #define `ROZMIAR` 9

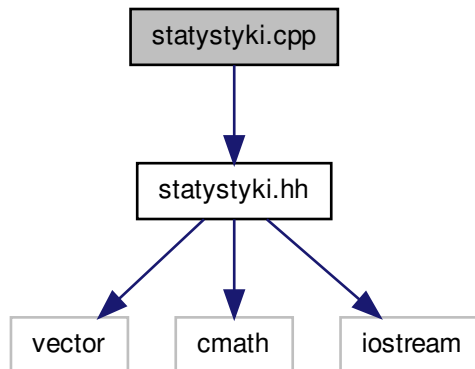
5.6.1 Dokumentacja definicji

5.6.1.1 #define `ROZMIAR` 9

Definicja w linii 3 pliku operacje.hh.

5.7 Dokumentacja pliku statystyki.cpp

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



Funkcje

- float [srednia](#) (float *tab, int rozmiar)
funkcja oblicza wartosc srednia
- float [odchylenie_standardowe](#) (float [srednia](#), float *tab, int rozmiar)
funkcja oblicza odchylenie standardowe

5.7.1 Dokumentacja funkcji

5.7.1.1 float [odchylenie_standardowe](#) (float [srednia](#), float * [tab](#), int [rozmiar](#))

funkcja oblicza odchylenie standardowe

Parametry

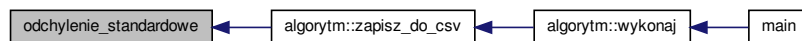
<i>tab</i>	- kontener zawierający czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywołań tej funkcji:



5.7.1.2 float srednia (float * tab, int rozmiar)

funkcja oblicza wartosc srednia

Parametry

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

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

Oto graf wywołań tej funkcji:

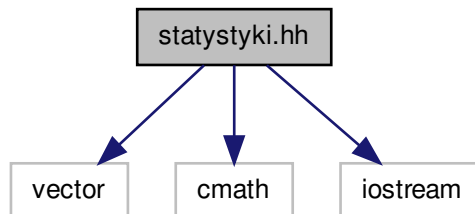


5.8 Dokumentacja pliku statystyki.hh

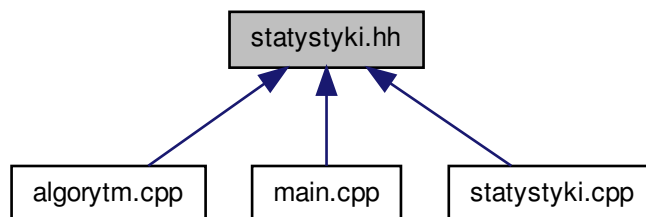
plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk

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

Wykres zależności załączania dla statystyki.hh:



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



Funkcje

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

5.8.1 Opis szczegółowy

plik zawiera deklaracje funkcji odpowiedzialnych za przeprowadzanie statystyk

Definicja w pliku `statystyki.hh`.

5.8.2 Dokumentacja funkcji

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

funckja oblicza odchylenie standardowe

Parametry

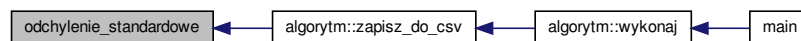
<i>tab</i>	- kontener zawierajacy czasy wykonania algorytmu
<i>srednia</i>	- wartosc srednia
<i>rozmiar</i>	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:



5.8.2.2 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.

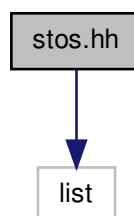
Oto graf wywołań tej funkcji:



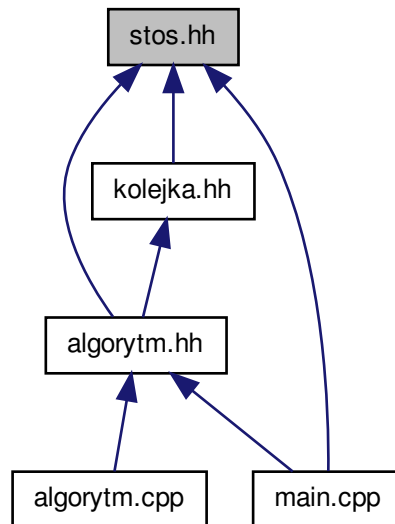
5.9 Dokumentacja pliku stos.hh

Plik zawiera definicje klasy `Stos`. Zaimplementowana na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiekszajacej svoj rozmiar b. powiekszajacej svoj rozmiar dwukrotnie, gdy stos sie 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 liście STL.
- class `stack_array< TYP >`
Modeluje stos w oparciu o tablice.

Wyliczenia

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

5.9.1 Opis szczegółowy

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

Definicja w pliku `stos.hh`.

5.9.2 Dokumentacja typów wyliczanych

5.9.2.1 enum flag

typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

Wartości wyliczeń:

plus1

x2

Definicja w linii 17 pliku stos.hh.

5.10 Dokumentacja pliku strona.dox