PAMSI - pwilkosz

1.0

Wygenerowano przez Doxygen 1.7.6.1

Sat Apr 5 2014 23:37:09

Spis treści

Rozdział 1

Indeks klas

1.1 Hierarchia klas

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

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

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

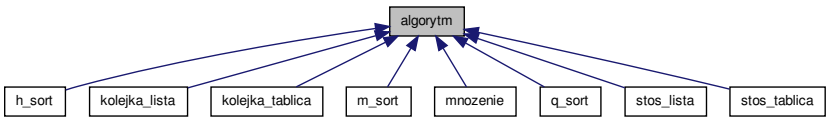
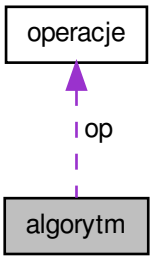


Diagram współpracy dla algorytm:



Metody publiczne

- algorytm (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor kopiujacy - 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 33 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 kopiujacy - przekazuje informacje o nazwach plikow, ktore zapisywane sa do pol klasy

Parametry

in	*plik1*	- plik wejsciowy
in	*plik2*	- plik wzorcowy
in	*N*	- ilosc danych wejsciowych
in	*M*	- ilosc powtorzen

Definicja w linii 76 pliku algorytm.hh.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 int algorytm::ile_danych ( )

Zwraca

ilosc liczb wejsciowych

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 98 pliku algorytm.cpp.

4.1.3.4 float algorytm::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym 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 wywoływań tej funkcji:

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

metoda ustawia wartosc n

Definicja w linii 90 pliku algorytm.hh.

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

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

Parametry

in	*plik*	- strumien pliku wejsciowego

Definicja w linii 10 pliku algorytm.cpp.

4.1.3.7 bool algorytm::wczytaj_wzor (ifstream & *plik*)

Metoda wczytuje plik wzorcowy do tablicy `dane_wz`.

Parametry

in	*plik*	- strumien pliku wejsciowego

Definicja w linii 21 pliku algorytm.cpp.

4.1.3.8 void algorytm::wlacz_zegar ( )

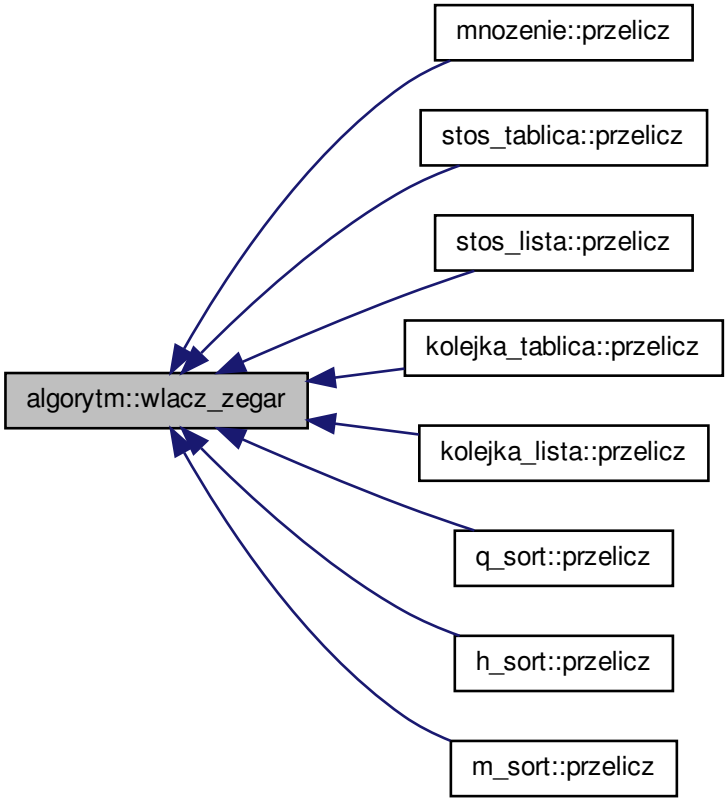
Metoda wlacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `start`.

Zwraca

start - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 38 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

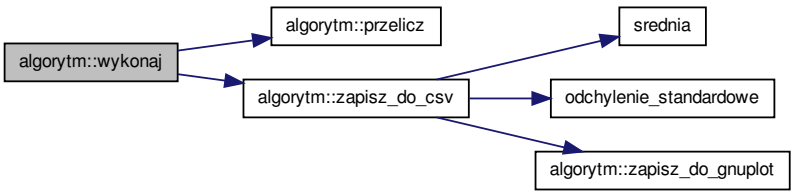


4.1.3.9 void **algorytm::wykonaj** (ofstream & *out*)

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

Definicja w linii 78 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



4.1.3.10 void algorytm::wylacz_zegar ( )

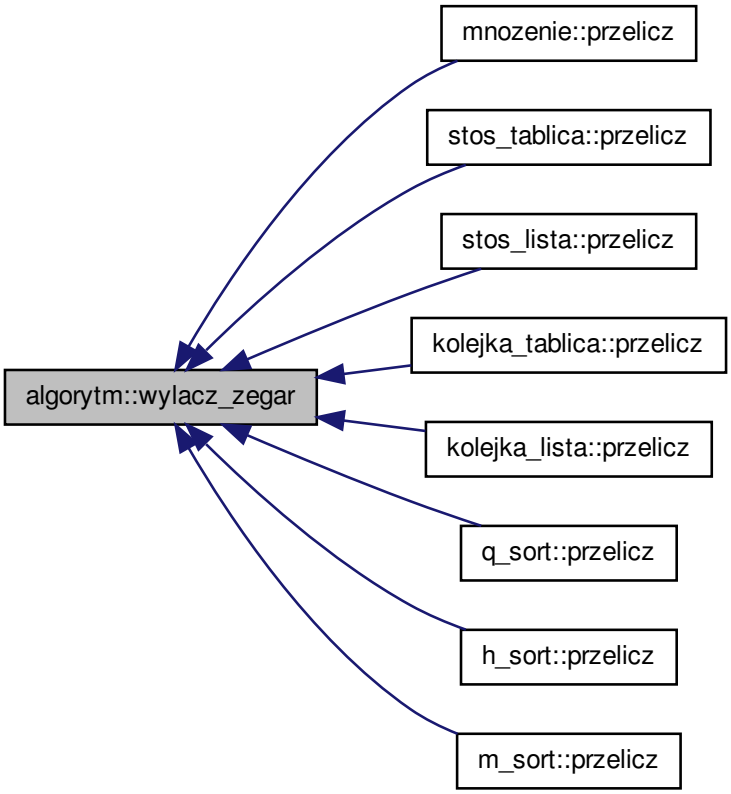
Metoda wyacza pomiar czasu poprzez wlaczenie funkcji `gettimeofday` i przechowanie czasu w zmiennej `end`.

Zwraca

end - zmienna pamietajaca czas poprzedzajacy wykonanie algorytmu

Definicja w linii 49 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

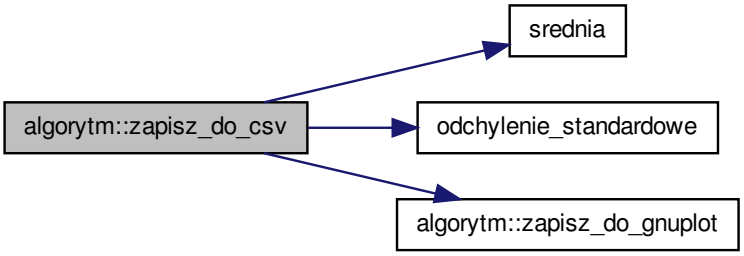


4.1.3.11 void algorytm::zapisz_do_csv (ofstream & *out*)

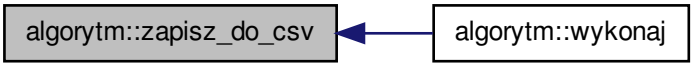
Metoda zapisuje tablice `czas` do pliku `wyjscie.csv`.

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 (ofstream & *out,* float *sr,* float *od*)

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

Definicja w linii 105 pliku algorytm.cpp.

Oto graf wywoływań tej funkcji:

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 float* algorytm::czas

zawiera wyniki dzialania algorytmu

Definicja w linii 66 pliku algorytm.hh.

4.1.4.2 double algorytm::czas1 [protected]

Definicja w linii 61 pliku algorytm.hh.

4.1.4.3 double algorytm::czas2 [protected]

Definicja w linii 61 pliku algorytm.hh.

4.1.4.4 float* algorytm::dane [protected]

Tablica liczb wczytana z pliku.

Definicja w linii 41 pliku algorytm.hh.

4.1.4.5 float* algorytm::dane_wz [protected]

tablica liczb zawartych w pliku wzorcowym

Definicja w linii 46 pliku algorytm.hh.

4.1.4.6 int algorytm::m [protected]

ilosc powtorzen

Definicja w linii 56 pliku algorytm.hh.

4.1.4.7 int algorytm::n [protected]

ilosc danych w pliku

Definicja w linii 52 pliku algorytm.hh.

4.1.4.8 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

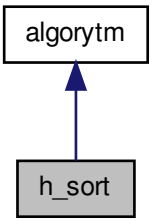
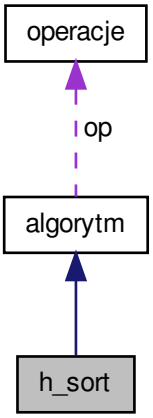


Diagram współpracy dla h_sort:



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 204 pliku algorytm.hh.

4.2.2 Dokumentacja konstruktora i destruktora

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

konstruktor klasy

Definicja w linii 207 pliku algorytm.hh.

4.2.3 Dokumentacja funkcji składowych

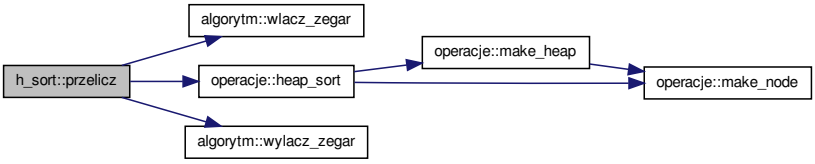
4.2.3.1 float h_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 179 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

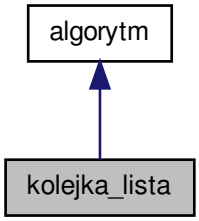
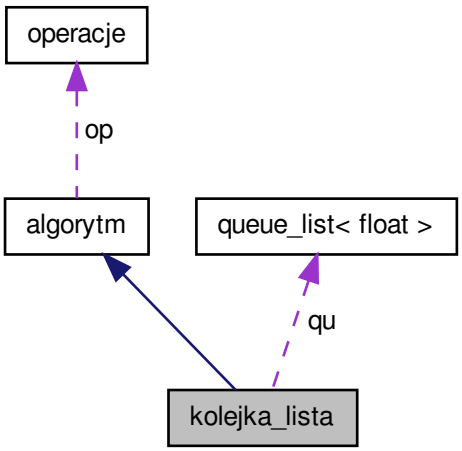


Diagram współpracy dla kolejka_lista:



Metody publiczne

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

  Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Atrybuty prywatne

- queue_list< float > qu

4.3.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 187 pliku algorytm.hh.

4.3.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 190 pliku algorytm.hh.

4.3.3 Dokumentacja funkcji składowych

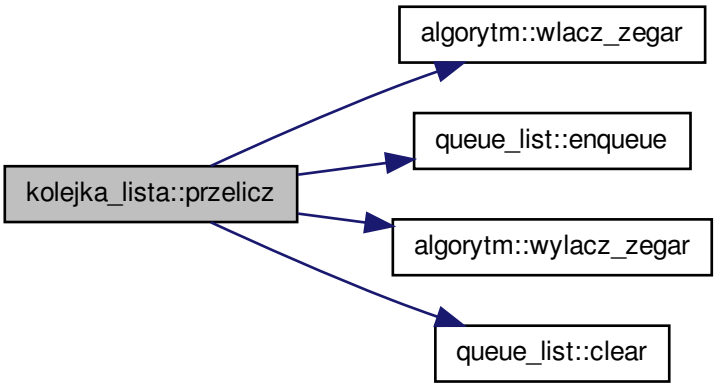
4.3.3.1 float kolejka_lista::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 159 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 188 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

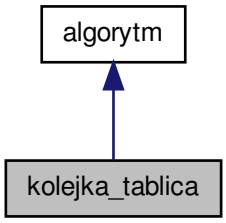
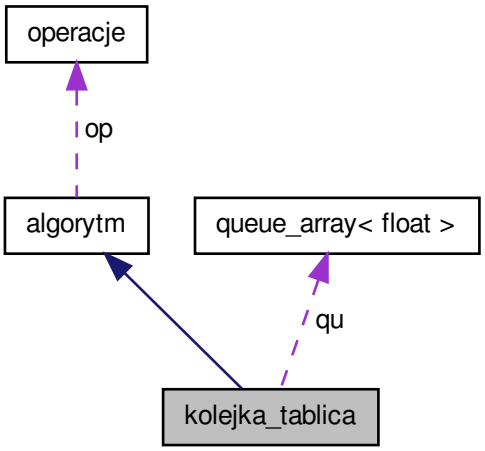


Diagram współpracy dla kolejka_tablica:



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 zadanym algorytmem.

Atrybuty prywatne

- queue_array< float > qu

4.4.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 175 pliku algorytm.hh.

4.4.2 Dokumentacja konstruktora i destruktora

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 181 pliku algorytm.hh.

4.4.3 Dokumentacja funkcji składowych

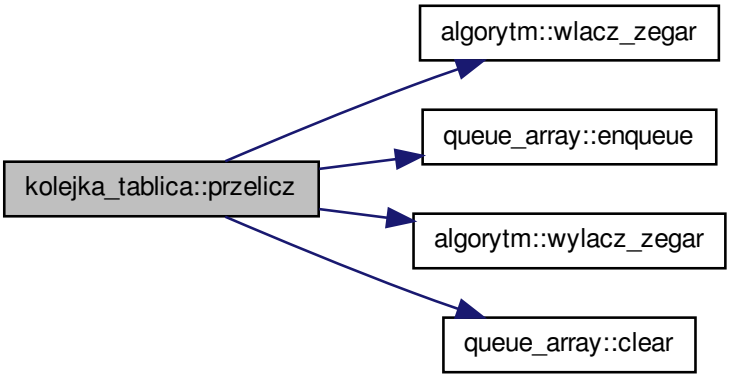
4.4.3.1 float kolejka_tablica::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 147 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 176 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

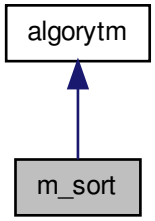
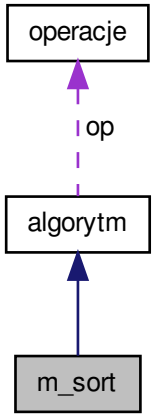


Diagram współpracy dla m_sort:



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 213 pliku algorytm.hh.

4.5.2 Dokumentacja konstruktora i destruktora

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

konstruktor

Definicja w linii 216 pliku algorytm.hh.

4.5.3 Dokumentacja funkcji składowych

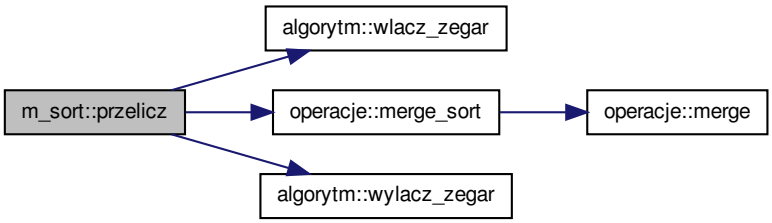
4.5.3.1 float m_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 188 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 wejsciowego przez 2

`#include <algorytm.hh>`

Diagram dziedziczenia dla mnozenie

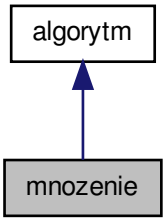
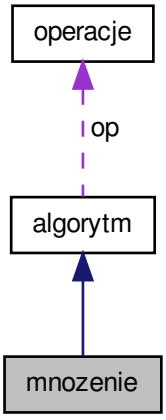


Diagram współpracy dla mnozenie:



Metody publiczne

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

  wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

4.6.1 Opis szczegółowy

modeluje algorytm dokonujacy mnozenia kazdego elementu pliku wejsciowego przez 2

Definicja w linii 135 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 wejsciowego i wzorcowego

Parametry

in	*plik1*	- plik wejsciowy
in	*plik2*	- plik wzorcowy
in	*N*	- ilosc danych wejsciowych
in	*M*	- ilosc powtorzen

Definicja w linii 144 pliku algorytm.hh.

4.6.3 Dokumentacja funkcji składowych

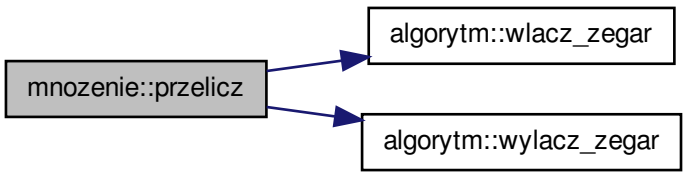
4.6.3.1 float mnozenie::przelicz ( ) [virtual]

wykonuje zalozony algorytm mnozenia elementow tablicy przez 2

Reimplementowana z algorytm.

Definicja w linii 113 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 bedacej polem klasy.*
- bool operator== (float ∗tab1)

  Przeciazenie operatora porownania; metoda porownuje zawartosci dwoch tablic.
- float & operator[] (int ind)

Atrybuty publiczne

- int n

  ilosc elementow w tablicy
- float * tab

  tablica z liczbami

4.7.1 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

in	*N*	- ilosc elementow w tablicy; parametr przypisywany do pola `n` 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

in	*e*	- 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, ktora nalezy dolaczyc
in	*rozm*	- rozmiar tablicy tab2

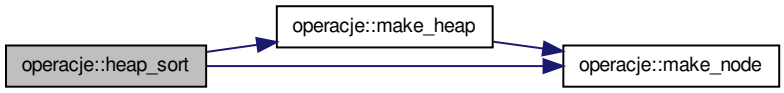
Definicja w linii 46 pliku operacje.cpp.

4.7.3.3 void operacje::heap_sort ( )

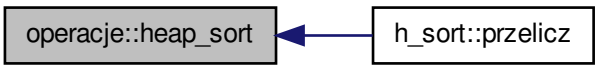
Metoda dokonuje sortowania po uprzednim utworzeniu kopca.

Definicja w linii 116 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



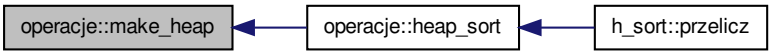
4.7.3.4 void operacje::make_heap ( )

Metoda tworzy kopiec binarny.

Definicja w linii 110 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*)

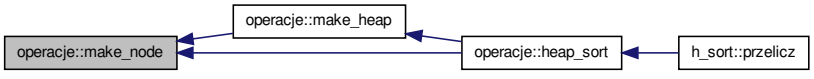
Metoda tworzy wezel drzewa, przypisujac mu 2 synow, ustawiajac ich w odpowiedniej kolejnosci (ojciec ma najwieksza wartosc)

Parametry

in	*rozmiar*	- rozmiar tablicy
in	*i*	- indeks elementu, do ktorego przypisujemy synow

Definicja w linii 95 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



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

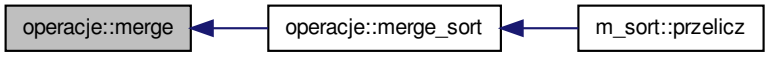
Metoda scala dwie czesci tablicy, jednoczesnie je porzadkujac.

Parametry

in	*poczatek*	- pierwszy indeks tablicy
in	*srodek*	- srodkowy indeks tablicy
	[ib]	koniec - ostatni indeks tablicy

Definicja w linii 130 pliku operacje.cpp.

Oto graf wywoływań tej funkcji:



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

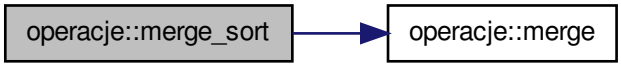
\ brief Metoda dokonuje sortowania poprzez rekurencyjne wywolanie dla obu polow tablic, nastepnie metoda dokonuje scalenia danych

Parametry

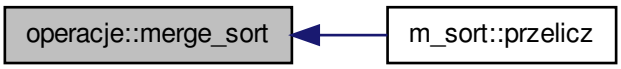
in	-	poczatek - pierwszy indeks tablicy
in	-	koniec - ostatni indeks tablicy

Definicja w linii 166 pliku operacje.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań 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 bedacej polem klasy.

Parametry

`in`	*tab1*	- 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 dwoch tablic.

Parametry

`in`	*tab1*	- tablica, ktorej wartosci porownujemy

Zwraca

true - gdy zawartsoc 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 *l*, int *p*)

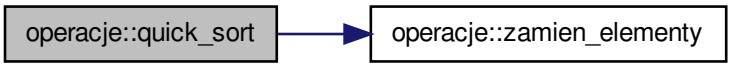
Metoda Dokonuje sortownaia szybkiego.

Parametry

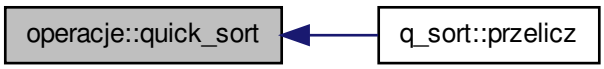
in	*l*	- pierwszy indeks tablicy
in	*p*	- ostatni indeks tablicy

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

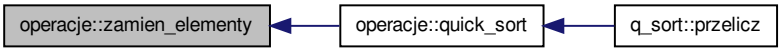
in	*i*	- element tablicy
in	*j*	- element tablicy

Zwraca

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

Definicja w linii 3 pliku operacje.cpp.

Oto graf wywoływań 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 szybkiemu

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

Diagram dziedziczenia dla q_sort

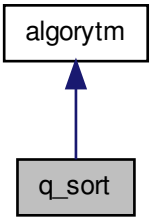
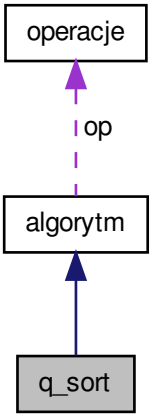


Diagram współpracy dla q_sort:



Metody publiczne

- q_sort (ifstream &plik1, ifstream &plik2, int N, int M)

  konstruktor klasy
- float przelicz ()

  metoda dokonujaca sortowania danych

4.8.1 Opis szczegółowy

klasa reprezentuje dane poddane sortowaniu szybkiemu

Definicja w linii 196 pliku algorytm.hh.

4.8.2 Dokumentacja konstruktora i destruktora

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

konstruktor klasy

Definicja w linii 199 pliku algorytm.hh.

4.8.3 Dokumentacja funkcji składowych

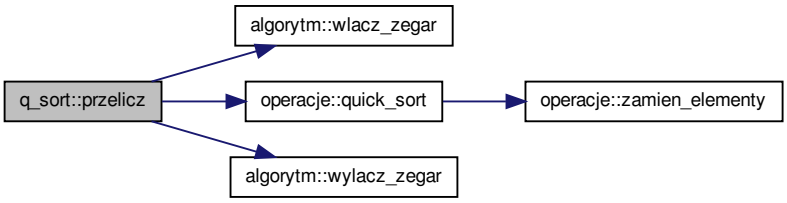
4.8.3.1 float q_sort::przelicz ( ) [virtual]

metoda dokonujaca sortowania danych

Reimplementowana z algorytm.

Definicja w linii 170 pliku algorytm.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

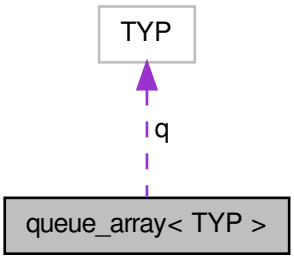
- algorytm.hh
- algorytm.cpp

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 destruktora

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.

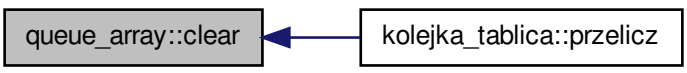
4.9.3 Dokumentacja funkcji składowych

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

czysci kolejke

Definicja w linii 173 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 129 pliku kolejka.hh.

4.9.3.3 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:

4.9.3.4 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 kolejke oparta na liscie STL.

Definicja w linii 19 pliku kolejka.hh.

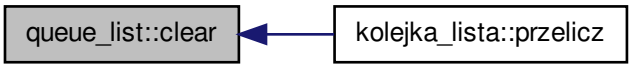
4.10.2 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 41 pliku kolejka.hh.

Oto graf wywoływań 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 wywoływań 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:

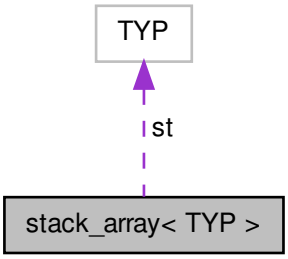
- kolejka.hh

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 destruktora

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.

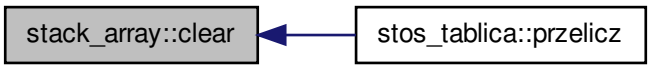
4.11.3 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 181 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

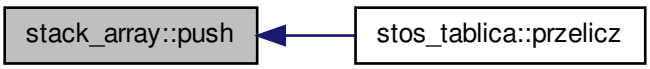
Definicja w linii 138 pliku stos.hh.

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 liscie STL.

Definicja w linii 22 pliku stos.hh.

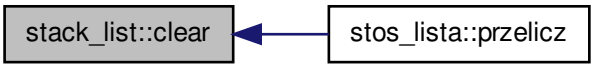
4.12.2 Dokumentacja funkcji składowych

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

czysci stos

Definicja w linii 50 pliku stos.hh.

Oto graf wywoływań 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 wywoływań 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

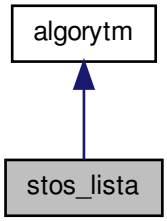
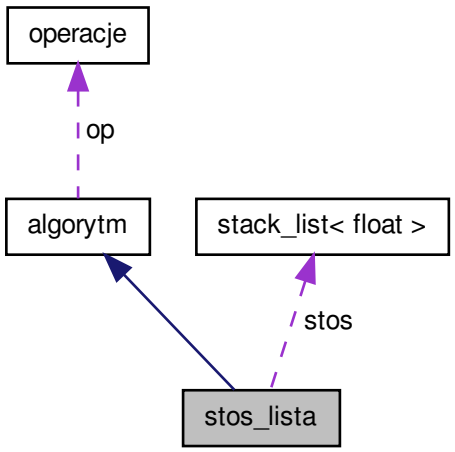


Diagram współpracy dla stos_lista:



Metody publiczne

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

  Metoda odpowiada za przetworzenie danych wejsciowych 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 166 pliku algorytm.hh.

4.13.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 169 pliku algorytm.hh.

4.13.3 Dokumentacja funkcji składowych

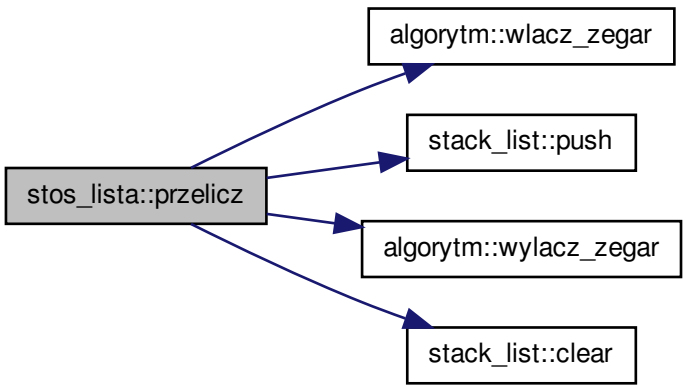
4.13.3.1 float stos_lista::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 135 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 167 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

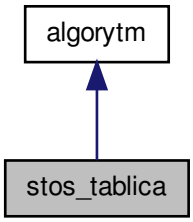
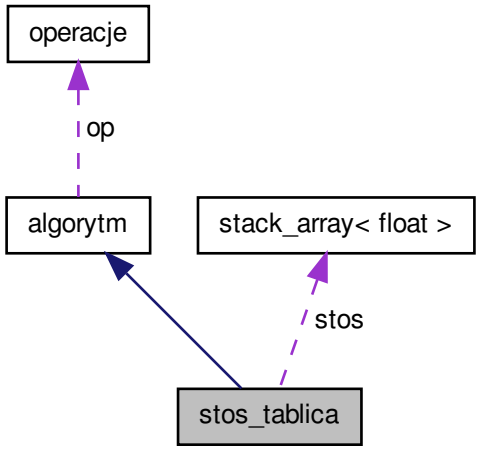


Diagram współpracy dla stos_tablica:



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 zadanym algorytmem.

Atrybuty prywatne

- stack_array< float > stos

4.14.1 Opis szczegółowy

klasa utworzona na potrzeby pomiaru czasu wypełnienia struktury

Definicja w linii 154 pliku algorytm.hh.

4.14.2 Dokumentacja konstruktora i destruktora

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 160 pliku algorytm.hh.

4.14.3 Dokumentacja funkcji składowych

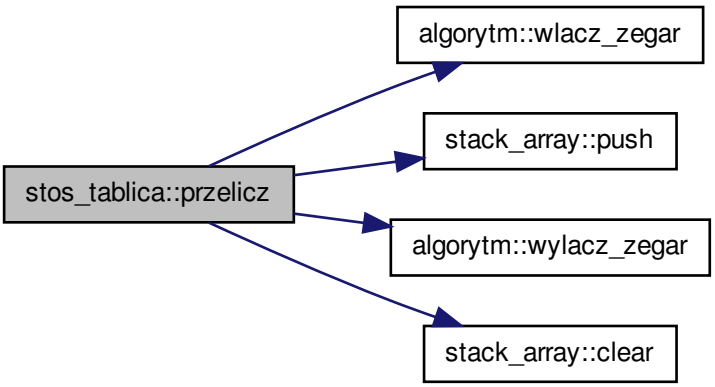
4.14.3.1 float stos_tablica::przelicz ( ) [virtual]

Metoda odpowiada za przetworzenie danych wejsciowych zgodnie z zadanym algorytmem.

Reimplementowana z algorytm.

Definicja w linii 124 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 155 pliku algorytm.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- algorytm.hh
- algorytm.cpp

4.15 Dokumentacja szablonu klasy tablica_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjna.

`#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 czy_pusta ()
- int zlicz_elementy ()

Metody prywatne

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

  Metoda ktora umieszcza wartosc oraz jej klucz w zadanym mejscu. Gdy wartosc z kluczem jest dodawana w srodek struktury, dane na prawo od niej przesuwane sa o jeden w prawo. Gdy istnieje potrzeba powiekszenia tablicy, stosuje sie znany juz radzaj gospodarowania pamiecia, gdzie rozmiar tablicy jest podwajany, co jest korzystne ze wzgledu na zlozonosc obliczeniowa.
- void wstaw (string k, TYP v, int ind_l, int ind_r)

  Metoda szuka pozycji, w ktora nalezy dodac element, aby tablica byla posortowana alfabetycznie.
- int znajdz (string k, int ind_l, int ind_r)

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

Atrybuty prywatne

- string ∗ key

  Tablica zawierajaca klucze poszukiwan.
- TYP ∗ value

  Tablica zawierajaca wartosci.
- int s

rozmiar tablicy

- int sp

  rozmiar danych zapelniajacych tablice

- bool found

  flaga informujaca o tym, czy dany klucz znaleziono w zbiorze

4.15.1 Opis szczegółowy

template<typename TYP>class tablica_asocjacyjna< TYP >

Klasa modeluje tablice asocjacyjna.

Definicja w linii 59 pliku tablica_asocjacyjna.hh.

4.15.2 Dokumentacja konstruktora i destruktora

4.15.2.1 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 151 pliku tablica_asocjacyjna.hh.

4.15.3 Dokumentacja funkcji składowych

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

Zwraca

true, gdy stos jest pusty, false w przeciwnym wypadku

Definicja w linii 207 pliku tablica_asocjacyjna.hh.

4.15.3.2 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 157 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:

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

Definicja w linii 75 pliku tablica_asocjacyjna.hh.

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

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

Zwraca

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

Definicja w linii 192 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:

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

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

Definicja w linii 172 pliku tablica_asocjacyjna.hh.

Oto graf wywoływań tej funkcji:

4.15.3.6 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 118 pliku tablica_asocjacyjna.hh.

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

Zwraca

ilosc elementow w strukturze

Definicja w linii 212 pliku tablica_asocjacyjna.hh.

4.15.3.8 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 130 pliku tablica_asocjacyjna.hh.

4.15.4 Dokumentacja atrybutów składowych

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

flaga informujaca o tym, czy dany klucz znaleziono w zbiorze

Definicja w linii 69 pliku tablica_asocjacyjna.hh.

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

Tablica zawierajaca klucze poszukiwan.

Definicja w linii 61 pliku tablica_asocjacyjna.hh.

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

rozmiar tablicy

Definicja w linii 65 pliku tablica_asocjacyjna.hh.

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

rozmiar danych zapelniajacych tablice

Definicja w linii 67 pliku tablica_asocjacyjna.hh.

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

Tablica zawierajaca wartosci.

Definicja w linii 63 pliku tablica_asocjacyjna.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- tablica_asocjacyjna.hh

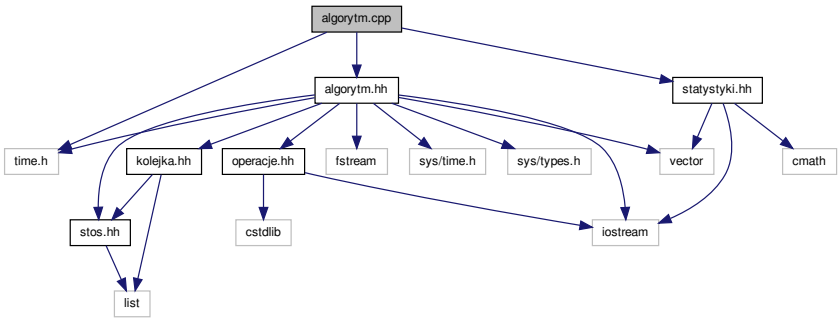
Rozdział 5

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

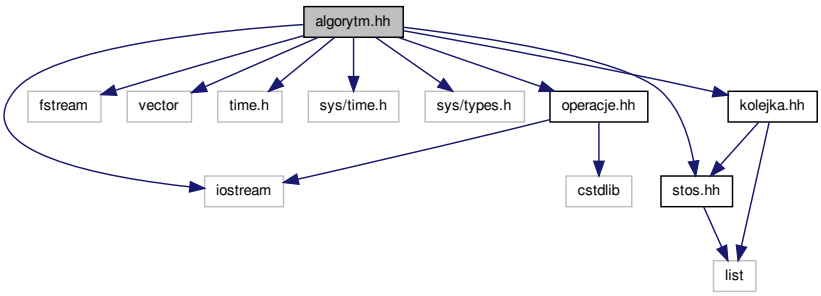
plik zawiera definicje metod klas zdefiniowanych w pliku algorytm.hh

Definicja w pliku algorytm.cpp.

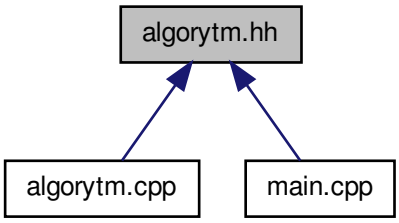
5.2 Dokumentacja pliku algorytm.hh

Definicja klas wykonujacych operacje na zestawie danych wejsciowych.

`#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"` 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, ktora ma za zadanie wczytac, przetworzyc i porownac 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

5.2.1 Opis szczegółowy

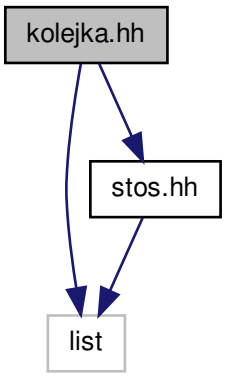
Definicja klas wykonujacych operacje na zestawie danych wejsciowych.

Definicja w pliku algorytm.hh.

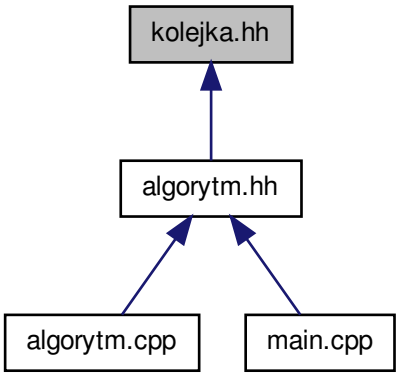
5.3 Dokumentacja pliku kolejka.hh

Plik zawiera definicje klasy Kolejka Zaimplementowanej na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiekszajacej swoj rozmiar b. powiekszajacej swoj rozmiar dwukrotnie, gdy kolejka sie 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 kolejke oparta na liscie STL.
- class queue_array< TYP >

  Modeluje kolejke 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. kazdorazowo powiekszajacej swoj rozmiar b. powiekszajacej swoj rozmiar dwukrotnie, gdy kolejka sie przepelni.

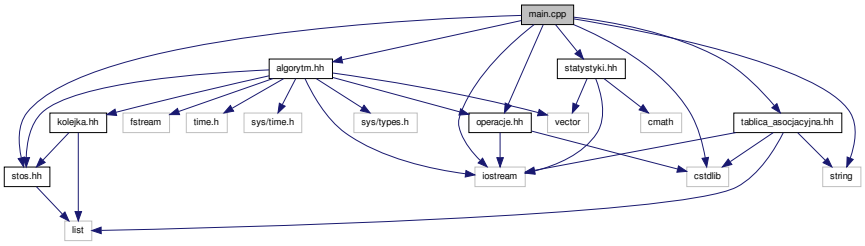
Definicja w pliku kolejka.hh.

5.4 Dokumentacja pliku main.cpp

plik glowny

```
#include <iostream> #include "algorytm.hh" #include "statystyki.-
hh"  #include "operacje.hh"  #include "stos.hh"  #include
"tablica_asocjacyjna.hh" #include <cstdlib> #include <string> ×
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main ()

5.4.1 Opis szczegółowy

plik glowny

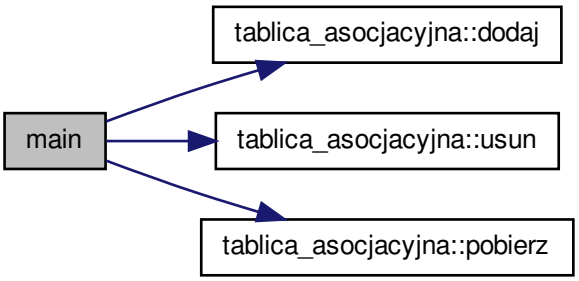
Definicja w pliku main.cpp.

5.4.2 Dokumentacja funkcji

5.4.2.1 int main ( )

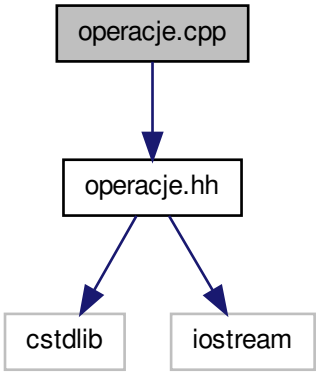
Definicja w linii 16 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



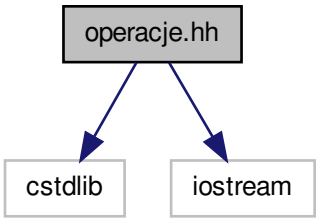
5.5 Dokumentacja pliku operacje.cpp

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

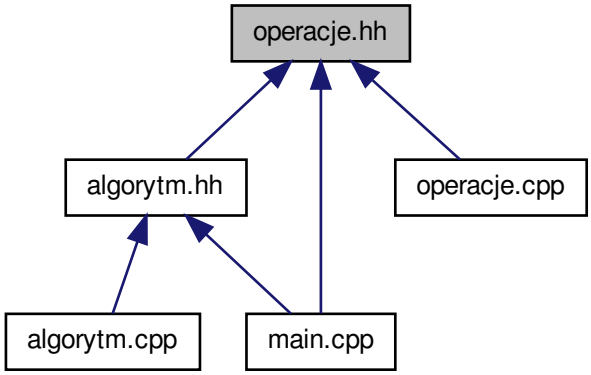


5.6 Dokumentacja pliku operacje.hh

`#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

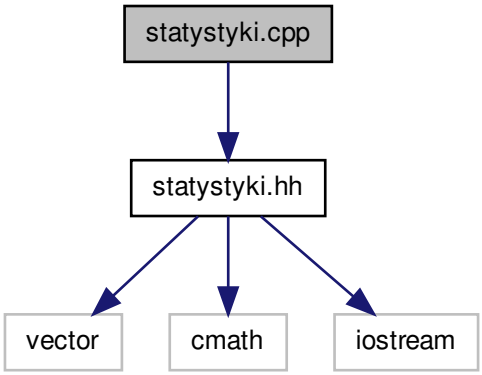
5.6.1 Dokumentacja definicji

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)

  funckja oblicza wartosc srednia
- float odchylenie_standardowe (float srednia, float *tab, int rozmiar)

  funckja oblicza odchylenie standardowe

5.7.1 Dokumentacja funkcji

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

funckja oblicza odchylenie standardowe

Parametry

tab	- kontener zawierajacy czasy wykonania algorytmu
srednia	- wartosc srednia
rozmiar	- rozmiar tablicy

Zwraca

odchylenie standardowe

Definicja w linii 16 pliku statystyki.cpp.

Oto graf wywoływań tej funkcji:

5.7.1.2 float srednia (float * *tab*, int *rozmiar*)

funckja oblicza wartosc srednia

Parametry

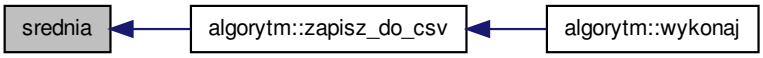
tab	- kontener zawierajacy czasy wykonania algorytmu
rozmiar	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

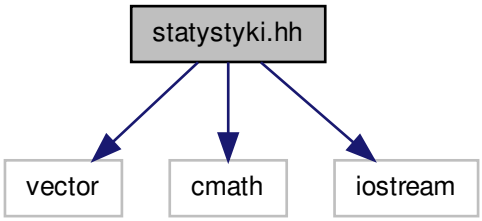
Oto graf wywoływań tej funkcji:



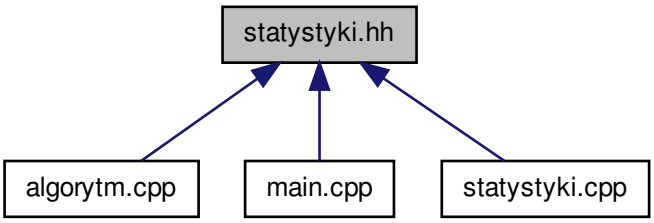
5.8 Dokumentacja pliku statystyki.hh

plik zawiera dekalracje funkcji odpowiedzialnych za przeprowadznaie statystyk

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



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



Funkcje

- float srednia (float ∗tab, int rozmiar)

  funckja oblicza wartosc srednia
- float odchylenie_standardowe (float srednia, float ∗tab, int rozmiar)

  funckja oblicza odchylenie standardowe

5.8.1 Opis szczegółowy

plik zawiera dekalracje funkcji odpowiedzialnych za przeprowadznaie 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

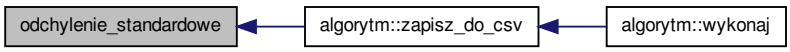
tab	- kontener zawierajacy czasy wykonania algorytmu
srednia	- wartosc srednia
rozmiar	- rozmiar tablicy

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

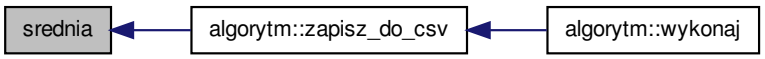
tab	- kontener zawierajacy czasy wykonania algorytmu
rozmiar	- rozmiar tablicy

Zwraca

wartosc srednia

Definicja w linii 3 pliku statystyki.cpp.

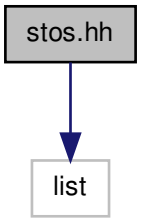
Oto graf wywoływań tej funkcji:



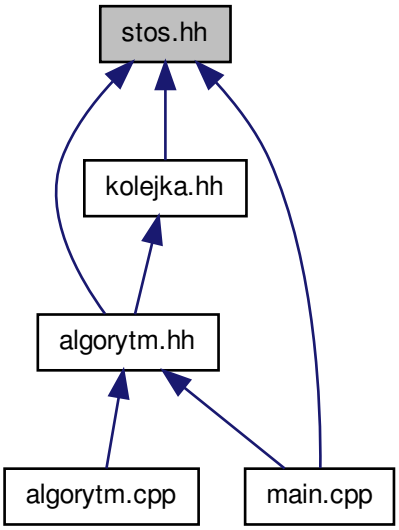
5.9 Dokumentacja pliku stos.hh

Plik zawiera definicje klasy `Stos` Zaimplementowana na 2 sposoby 1. Za pomocą listy. 2. Za pomocą tablicy a. kazdorazowo powiekszajacej swoj rozmiar b. powiekszajacej swoj 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 liscie STL.
- class stack_array< TYP >

  Modeluje stos w oparciu o tablice.

Wyliczenia

- enum flag { plus1, x2 }

  typ wyliczeniowy sluzacy do ustawienia sposobu zwiekszania pamieci

5.9.1 Opis szczegółowy

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

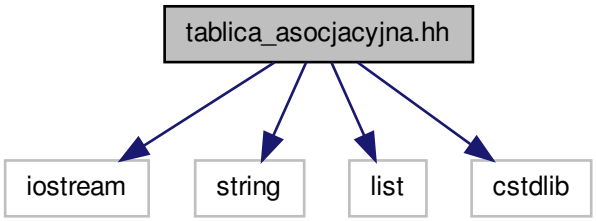
x2

Definicja w linii 17 pliku stos.hh.

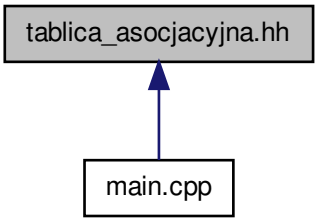
5.10 Dokumentacja pliku strona.dox

5.11 Dokumentacja pliku tablica_asocjacyjna.hh

`#include <iostream> #include <string> #include <list> ×
#include <cstdlib>` Wykres zależności załączania dla tablica_asocjacyjna.hh:



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



Komponenty

- class tablica_asocjacyjna< TYP >

  Klasa modeluje tablice asocjacyjna.

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)

5.11.1 Opis szczegółowy

Plik zawiera definicje klasy tablica_asocjacyjna, oraz definicje funkcji pomocniczych jako przeciazen operatorow porownania dla klasy typu string

Definicja w pliku tablica_asocjacyjna.hh.

5.11.2 Dokumentacja funkcji

5.11.2.1 bool operator< (string *s1*, string *s2*)

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

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

Definicja w linii 16 pliku tablica_asocjacyjna.hh.

5.11.2.2 bool operator<= (string *s1,* string *s2*)

Zwraca

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

Definicja w linii 40 pliku tablica_asocjacyjna.hh.

5.11.2.3 bool operator== (string *s1,* string *s2*)

Zwraca

true, gdy łańcuchy są identyczne

Definicja w linii 52 pliku tablica_asocjacyjna.hh.

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

funkcja sluzaca do alfabetycznego porzadkowania napisow

Zwraca

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

Definicja w linii 29 pliku tablica_asocjacyjna.hh.

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

Zwraca

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

Definicja w linii 46 pliku tablica_asocjacyjna.hh.