lab4

0.0004

Generated by Doxygen 1.8.9.1

Thu Apr 16 2015 04:48:47

Contents

Chapter 1

Laboratorium 4

Author

Filip Malinowski (Conributor Szymon Furmańczyk)

Date

16.04.2015

Version

0.0004

Program sluzacy do uruchamiania algorytmow i badania ich szybkosci dzialania.
W programie zaimplementowane sa algorytmy:

- wczytywanie danej ilosci elementow do stosu
- wczytywanie danej ilosci elementow do kolejki
- wczytywanie danej ilosci elementow do listy
- wczytywanie danej ilosci elementow do listy tablicowej
- quicksort tablicy n elementowej

Do wykonania podanych powyzej trzech ostatnich algorytmow zostaly zaimplementowane potrzebne struktury danych (stos, kolejka, lista oraz lista tablicowa).

Ciala klas znajduja sie w folderze ./prj/inc
Definicje metod znajduja sie w folderze ./prj/src
Sprawozdanie znajduje sie w folderze ./prj/doc/sprawozdanie

Format wywolania:

```
./prj/make clean\n
./prj/make\n
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

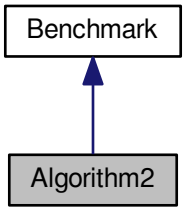
Class Documentation

5.1 Algorithm2 Class Reference

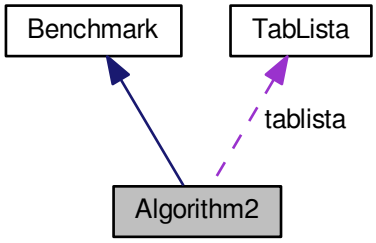
Klasa Algorithm2 modelujaca algorytm wczytywania do listy tablicowej. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do listy tablicowej.

`#include <algorithm2.hh>`

Inheritance diagram for Algorithm2:



Collaboration diagram for Algorithm2:



Public Member Functions

- Algorithm2 ()

  Konstruktor obiektu Algorithm2.
- Algorithm2 (int _tab[ ])

  Konstruktor parametryczny obiektu Algorithm2.
- ~Algorithm2 ()

  Destruktor obiektu Algorithm2.
- virtual void testAlgorithm (Benchmark *_algorithm)

  Metoda testowania algorytmu. Metoda sluzy do testowania szybkosci dzialania algorytmu. W klasie Algorithm2 nie ma konkretnego dzialania.
- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Wczytuje elementy do kolejki.
- virtual void runAlgorithmSzybkieSortowanie ()

  Metoda uruchamiania algorytmu szybkiego sortowania. Metoda sluzy to wykonywania danego algorytmu. Sortuje tablicę.

Private Attributes

- int tab [SIZE]

  Tablica elementow z danymi wejsciowymi.
- TabLista tablista

  Zmienna przechowujaca liste tablicowa.

5.1.1 Detailed Description

Definition at line 9 of file algorithm2.hh.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Algorithm2::Algorithm2 ( ) [inline]

Definition at line 24 of file algorithm2.hh.

5.1.2.2 Algorithm2::Algorithm2 (int *_tab[ ]*) [inline]

Parameters

in	*_tab*	- tablica przechowujaca dane wejsciowe.

Definition at line 30 of file algorithm2.hh.

5.1.2.3 Algorithm2::~Algorithm2 ( ) [inline]

Definition at line 35 of file algorithm2.hh.

5.1.3 Member Function Documentation

5.1.3.1 void Algorithm2::runAlgorithm (int *_border*) [virtual]

Parameters

in	*_border*	- ilosc elementow dla ktorych algorytm ma wykonac swoje dzialanie.

Reimplemented from Benchmark.

Definition at line 8 of file algorithm2.cpp.

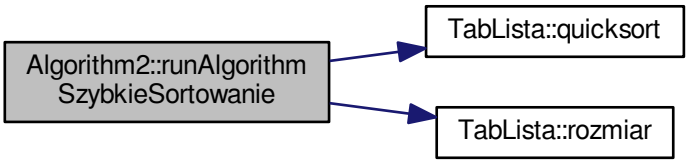
Here is the call graph for this function:

5.1.3.2 void Algorithm2::runAlgorithmSzybkieSortowanie ( ) [virtual]

Reimplemented from Benchmark.

Definition at line 17 of file algorithm2.cpp.

Here is the call graph for this function:



5.1.3.3 virtual void Algorithm2::testAlgorithm (Benchmark * *_algorithm*) [inline], [virtual]

Parameters

in	*_algorithm*	- testowany algorytm.

Definition at line 43 of file algorithm2.hh.

5.1.4 Member Data Documentation

5.1.4.1 int Algorithm2::tab[SIZE] [private]

Definition at line 14 of file algorithm2.hh.

5.1.4.2 TabLista Algorithm2::tablista [private]

Definition at line 18 of file algorithm2.hh.

The documentation for this class was generated from the following files:

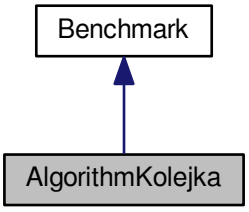
- algorithm2.hh
- algorithm2.cpp

5.2 AlgorithmKolejka Class Reference

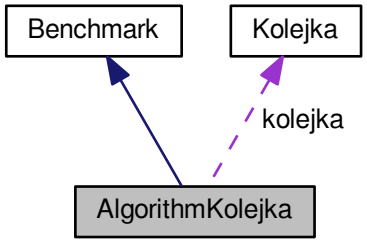
Klasa AlgorithmKolejka modelujaca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do kolejki.

`#include <algorithm_kolejka.hh>`

Inheritance diagram for AlgorithmKolejka:



Collaboration diagram for AlgorithmKolejka:



Public Member Functions

- AlgorithmKolejka ()

  Konstruktor obiektu AlgorithmKolejka.
- AlgorithmKolejka (int _tab[ ])

  Konstruktor parametryczny obiektu AlgorithmKolejka.
- ~AlgorithmKolejka ()

  Destruktor obiektu AlgorithmKolejka.

- virtual void testAlgorithm (Benchmark *_algorithm)

  Metoda testowania algorytmu. Metoda sluzy do testowania szybkosci dzialania algorytmu. W klasie AlgorithmKolejka nie ma konkretnego dzialania.

- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Wczytuje elementy do kolejki.

Private Attributes

- int tab [SIZE]

  Tablica elementow z danymi wejsciowymi.

- Kolejka kolejka

  Zmienna przechowujaca kolejke.

5.2.1 Detailed Description

Definition at line 9 of file algorithm_kolejka.hh.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AlgorithmKolejka::AlgorithmKolejka ( ) [inline]

Definition at line 24 of file algorithm_kolejka.hh.

5.2.2.2 AlgorithmKolejka::AlgorithmKolejka (int *_tab[ ]*) [inline]

Parameters

in	*_tab*	- tablica przechowujaca dane wejsciowe.

Definition at line 30 of file algorithm_kolejka.hh.

5.2.2.3 AlgorithmKolejka::~AlgorithmKolejka ( ) [inline]

Definition at line 35 of file algorithm_kolejka.hh.

5.2.3 Member Function Documentation

5.2.3.1 void AlgorithmKolejka::runAlgorithm (int *_border*) [virtual]

Parameters

in	*_border*	- ilosc elementow dla ktorych algorytm ma wykonac swoje dzialanie.

Reimplemented from Benchmark.

Definition at line 8 of file algorithm_kolejka.cpp.

Here is the call graph for this function:

5.2.3.2 virtual void AlgorithmKolejka::testAlgorithm (Benchmark * *_algorithm*) [inline], [virtual]

Parameters

in	*_algorithm*	- testowany algorytm.

Definition at line 43 of file algorithm_kolejka.hh.

5.2.4 Member Data Documentation

5.2.4.1 Kolejka AlgorithmKolejka::kolejka [private]

Definition at line 18 of file algorithm_kolejka.hh.

5.2.4.2 int AlgorithmKolejka::tab[SIZE] [private]

Definition at line 14 of file algorithm_kolejka.hh.

The documentation for this class was generated from the following files:

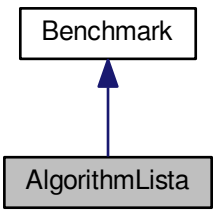
- algorithm_kolejka.hh
- algorithm_kolejka.cpp

5.3 AlgorithmLista Class Reference

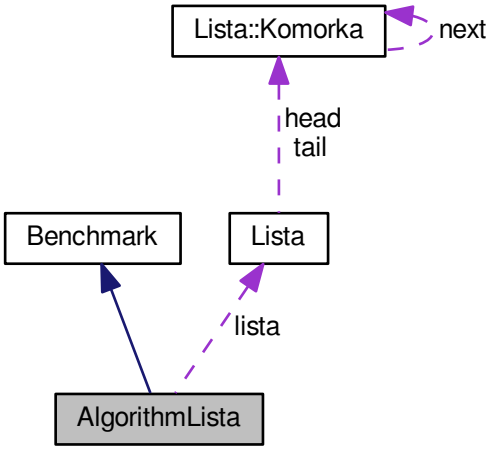
Klasa AlgorithmLista modelujaca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do listy.

`#include <algorithm_lista.hh>`

Inheritance diagram for AlgorithmLista:



Collaboration diagram for AlgorithmLista:



Public Member Functions

- AlgorithmLista ()

  Konstruktor obiektu AlgorithmLista.
- AlgorithmLista (int _tab[ ])

  Konstruktor parametryczny obiektu AlgorithmLista.
- ~AlgorithmLista ()

  Destruktor obiektu AlgorithmLista.
- virtual void testAlgorithm (Benchmark *_algorithm)

  Metoda testowania algorytmu. Metoda sluzy do testowania szybkosci dzialania algorytmu. W klasie AlgorithmLista nie ma konkretnego dzialania.
- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Wczytuje elementy do kolejki.

Private Attributes

- int tab [SIZE]

  Tablica elementow z danymi wejsciowymi.
- Lista lista

  Zmienna przechowujaca liste.

5.3.1 Detailed Description

Definition at line 9 of file algorithm_lista.hh.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 AlgorithmLista::AlgorithmLista ( ) [inline]

Definition at line 24 of file algorithm_lista.hh.

5.3.2.2 AlgorithmLista::AlgorithmLista (int *_tab[ ]*) [inline]

Parameters

in	*_tab*	- tablica przechowujaca dane wejsciowe.

Definition at line 30 of file algorithm_lista.hh.

5.3.2.3 AlgorithmLista::~AlgorithmLista ( ) [inline]

Definition at line 35 of file algorithm_lista.hh.

5.3.3 Member Function Documentation

5.3.3.1 void AlgorithmLista::runAlgorithm (int *_border*) [virtual]

Parameters

in	*_border*	- ilosc elementow dla ktorych algorytm ma wykonac swoje dzialanie.

Reimplemented from Benchmark.

Definition at line 8 of file algorithm_lista.cpp.

Here is the call graph for this function:

5.3.3.2 virtual void AlgorithmLista::testAlgorithm (Benchmark * *_algorithm*) [inline], [virtual]

Parameters

in	*_algorithm*	- testowany algorytm.

Definition at line 43 of file algorithm_lista.hh.

5.3.4 Member Data Documentation

5.3.4.1 Lista AlgorithmLista::lista [private]

Definition at line 18 of file algorithm_lista.hh.

5.3.4.2 int AlgorithmLista::tab[SIZE] [private]

Definition at line 14 of file algorithm_lista.hh.

The documentation for this class was generated from the following files:

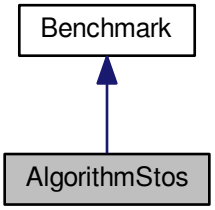
- algorithm_lista.hh
- algorithm_lista.cpp

5.4 AlgorithmStos Class Reference

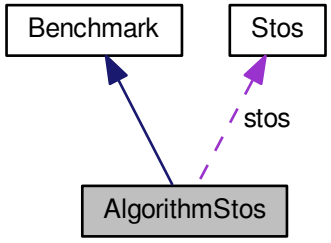
Klasa AlgorithmStos modelujaca algorytm wczytywania do stosu. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do stosu.

`#include <algorithm_stos.hh>`

Inheritance diagram for AlgorithmStos:



Collaboration diagram for AlgorithmStos:



Public Member Functions

- AlgorithmStos ()

  Konstruktor obiektu AlgorithmStos.
- AlgorithmStos (int _tab[ ])

  Konstruktor parametryczny obiektu AlgorithmStos.
- ~AlgorithmStos ()

  Destruktor obiektu AlgorithmStos.
- virtual void testAlgorithm (Benchmark *_algorithm)

  Metoda testowania algorytmu. Metoda sluzy do testowania szybkosci dzialania algorytmu. W klasie AlgorithmStos nie ma konkretnego dzialania.

- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Wczytuje elementy do stosu.

Private Attributes

- int tab [SIZE]

  Tablica elementow z danymi wejsciowymi.
- Stos stos

  Zmienna przechowujaca stos.

5.4.1 Detailed Description

Definition at line 9 of file algorithm_stos.hh.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 AlgorithmStos::AlgorithmStos ( ) [inline]

Definition at line 24 of file algorithm_stos.hh.

5.4.2.2 AlgorithmStos::AlgorithmStos (int *_tab[ ]*) [inline]

Parameters

in	*_tab*	- tablica przechowujaca dane wejsciowe.

Definition at line 30 of file algorithm_stos.hh.

5.4.2.3 AlgorithmStos::~AlgorithmStos ( ) [inline]

Definition at line 35 of file algorithm_stos.hh.

5.4.3 Member Function Documentation

5.4.3.1 void AlgorithmStos::runAlgorithm (int *_border*) [virtual]

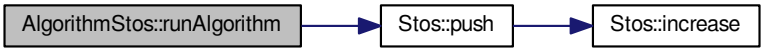
Parameters

in	*_border*	- ilosc elementow dla ktorych algorytm ma wykonac swoje dzialanie.

Reimplemented from Benchmark.

Definition at line 8 of file algorithm_stos.cpp.

Here is the call graph for this function:



5.4.3.2 virtual void AlgorithmStos::testAlgorithm (Benchmark * _algorithm) [inline], [virtual]

Parameters

in	_algorithm	- testowany algorytm.

Definition at line 43 of file algorithm_stos.hh.

5.4.4 Member Data Documentation

5.4.4.1 Stos AlgorithmStos::stos [private]

Definition at line 18 of file algorithm_stos.hh.

5.4.4.2 int AlgorithmStos::tab[SIZE] [private]

Definition at line 14 of file algorithm_stos.hh.

The documentation for this class was generated from the following files:

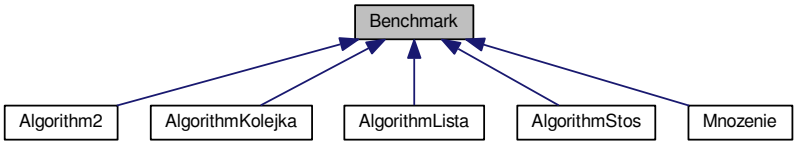
- algorithm_stos.hh
- algorithm_stos.cpp

5.5 Benchmark Class Reference

Klasa Benchmark modelujaca program benchmarkujacy. Obiekt tego typu reprezentuje program sprawdzajacy szybkosc wykonywania algorytmow.

`#include <benchmark.hh>`

Inheritance diagram for Benchmark:



Public Member Functions

- Benchmark ()

  Konstrukor obiektu Benchmark.
- ~Benchmark ()

  Destruktor obiektu Benchmark.
- virtual void testAlgorithm (Benchmark *_algorithm, int _n) const

  Metoda testowania algorytmu. Metoda sluzy to testowania szybkosci dzialania algorytmu. Wykonuje testowany algorytm dla 5 kolejnych ilosci elementow. Wykonanie algorytmu dla danego zestawu liczb powtarza dwa razy i usrednia wynik. Otrzymany czas wraz z iloscia testowanych danych zapisuje w pliku ret_data.txt.
- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy do wykonywania danego algorytmu. W klasie Benchmark nie ma konkretnego dzialania.

- virtual void runAlgorithmSzybkieSortowanie ()

  Metoda uruchamiania algorytmu szybkiego sortowania. Metoda sluzy do wykonywania danego algorytmu. W klasie Benchmark nie ma konkretnego dzialania.

Private Attributes

- std::string nazwy [4] = {"ret_data1.txt", "ret_data2.txt", "ret_data3.txt", "ret_data4.txt"}

  Tablica stringow przechowujaca nazwy plikow do zapisu.

5.5.1 Detailed Description

Definition at line 11 of file benchmark.hh.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Benchmark::Benchmark ( ) [inline]

Definition at line 23 of file benchmark.hh.

5.5.2.2 Benchmark::~Benchmark ( ) [inline]

Definition at line 28 of file benchmark.hh.

5.5.3 Member Function Documentation

5.5.3.1 virtual void Benchmark::runAlgorithm (int *_border*) [inline], [virtual]

Reimplemented in Algorithm2, AlgorithmKolejka, AlgorithmLista, AlgorithmStos, and Mnozenie.

Definition at line 47 of file benchmark.hh.

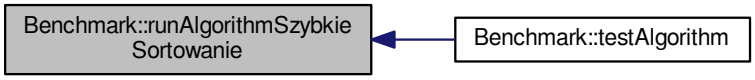
Here is the caller graph for this function:

5.5.3.2 virtual void Benchmark::runAlgorithmSzybkieSortowanie ( ) [inline], [virtual]

Reimplemented in Algorithm2.

Definition at line 54 of file benchmark.hh.

Here is the caller graph for this function:



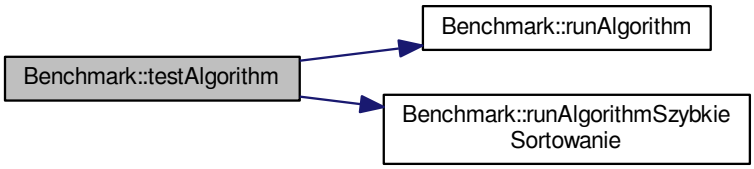
5.5.3.3 void Benchmark::testAlgorithm (Benchmark * *_algorithm*, int *_n*) const [virtual]

Parameters

in	*_algorithm*	- testowany algorytm.
in	*_n*	- indeks nazwy pliku

Definition at line 11 of file benchmark.cpp.

Here is the call graph for this function:



5.5.4 Member Data Documentation

5.5.4.1 std::string Benchmark::nazwy[4] = {"ret_data1.txt", "ret_data2.txt", "ret_data3.txt", "ret_data4.txt"} [private]

Definition at line 16 of file benchmark.hh.

The documentation for this class was generated from the following files:

- benchmark.hh
- benchmark.cpp

5.6 Kolejka Class Reference

Klasa Kolejka modelujaca strukture danych typu kolejka. Obiekt tego typu reprezentuje strukture danych typu kolejka wraz z operacjami mozliwymi do wykonania na tej strukturze.

`#include <kolejka.hh>`

Public Member Functions

- Kolejka ()

  Konstruktor obiektu Kolejka.
- Kolejka (long _size)

  Konstruktor parametryczny obiektu Kolejka.
- ~Kolejka ()

  Destruktor obiektu Kolejka.
- void enqueue (int _elem)

  Metoda dodawnia elementu. Metoda sluzy do dodawania elementu do kolejki.
- int dequeue ()

  Metoda usuwania elementu. Metoda sluzy do usuwania elementu z kolejki.

Private Member Functions

- void increase ()

  Metoda powiekszania kolejki. Metoda ta dodaje do kolejki 8 kolejnych wolnych pol pomiedzy pierwszym a ostatnim zapisanym elementem.

Private Attributes

- int * tab

  Wskaznik na tablice elementow kolejki.
- long size

  Rozmiar kolejki.
- int f

  Indeks pierwszego zapisanego elementu.
- int r

  Indeks ostatniego wolnego pola.

5.6.1 Detailed Description

Definition at line 9 of file kolejka.hh.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Kolejka::Kolejka ( )

Definition at line 5 of file kolejka.cpp.

5.6.2.2 Kolejka::Kolejka (long *_size*)

Parameters

in	*_size*	- rozmiar tworzonej kolejki.

Definition at line 14 of file kolejka.cpp.

5.6.2.3 Kolejka::~Kolejka ( )

Definition at line 23 of file kolejka.cpp.

5.6.3 Member Function Documentation

5.6.3.1 int Kolejka::dequeue ( )

Returns

- usuwany element.

Definition at line 48 of file kolejka.cpp.

5.6.3.2 void Kolejka::enqueue (int *_elem*)

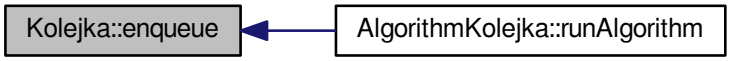
Parameters

in	*_elem*	- dodawany element.

Definition at line 40 of file kolejka.cpp.

Here is the call graph for this function:

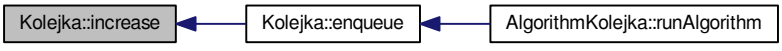
Here is the caller graph for this function:



5.6.3.3 void Kolejka::increase ( ) [private]

Definition at line 27 of file kolejka.cpp.

Here is the caller graph for this function:



5.6.4 Member Data Documentation

5.6.4.1 int Kolejka::f [private]

Definition at line 22 of file kolejka.hh.

5.6.4.2 int Kolejka::r [private]

Definition at line 26 of file kolejka.hh.

5.6.4.3 long Kolejka::size [private]

Definition at line 18 of file kolejka.hh.

5.6.4.4 int* Kolejka::tab [private]

Definition at line 14 of file kolejka.hh.

The documentation for this class was generated from the following files:

- kolejka.hh
- kolejka.cpp

5.7 Lista::Komorka Struct Reference

Struktura Komorka. Obiekt tego typu reprezentuje pojedyncza komorke wraz ze wskaznikiem na nastepna komorke listy.

Collaboration diagram for Lista::Komorka:

Public Member Functions

- Komorka (int _elem)

  Konstruktor paramteryczny obiektu Komorka.

Public Attributes

- int elem

  Wartosc elementu w pojedynczej komorce.
- Komorka * next

  Wskaznik na nastepna komorke listy.

5.7.1 Detailed Description

Definition at line 16 of file lista.hh.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Lista::Komorka::Komorka (int *_elem*) [inline]

Parameters

in	*_elem*	- wartosc przechowywanego elementu.

Definition at line 29 of file lista.hh.

5.7.3 Member Data Documentation

5.7.3.1 int Lista::Komorka::elem

Definition at line 20 of file lista.hh.

5.7.3.2 Komorka* Lista::Komorka::next

Definition at line 24 of file lista.hh.

The documentation for this struct was generated from the following file:

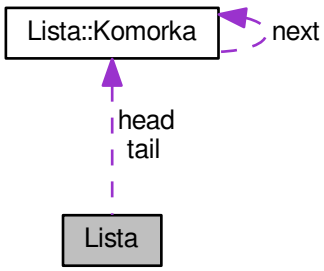
- lista.hh

5.8 Lista Class Reference

Klasa Lista modelujaca strukture danych typu lista. Obiekt tego typu reprezentuje strukture danych typu lista wraz z operacjami mozliwymi do wykonania na tej strukturze.

`#include <lista.hh>`

Collaboration diagram for Lista:



Classes

- struct Komorka

  Struktura Komorka. Obiekt tego typu reprezentuje pojedyncza komorke wraz ze wskaznikiem na nastepna komorke listy.

Public Member Functions

- Lista ()

  Konstruktor obiektu Lista.
- ∼Lista ()

  Destruktor obiektu Lista.
- void insert (int _elem)

  Metoda dodawnia elementu. Metoda sluzy do dodawania elementu do konca listy.
- int remove (int _f)

  Metoda usuwania elementu. Metoda sluzy do usuwania elementu o wskazanym indeksie.

Private Attributes

- Komorka ∗ head

  Wskaznik na poczatek listy.
- Komorka ∗ tail

  Wskaznik na koniec listy.

5.8.1 Detailed Description

Definition at line 9 of file lista.hh.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Lista::Lista ( ) [inline]

Definition at line 49 of file lista.hh.

5.8.2.2 Lista::∼Lista ( ) [inline]

Definition at line 54 of file lista.hh.

5.8.3 Member Function Documentation

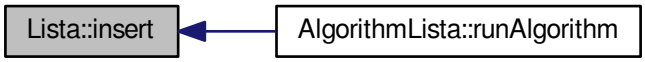
5.8.3.1 void Lista::insert (int *_elem*)

Parameters

in	*_elem*	- dodawany element.

Definition at line 5 of file lista.cpp.

Here is the caller graph for this function:



5.8.3.2 int Lista::remove (int _f)

Parameters

in	_f	- indeks elementu do usuniecia.

Returns

- usuwany element.

Definition at line 19 of file lista.cpp.

5.8.4 Member Data Documentation

5.8.4.1 Komorka* Lista::head [private]

Definition at line 38 of file lista.hh.

5.8.4.2 Komorka* Lista::tail [private]

Definition at line 42 of file lista.hh.

The documentation for this class was generated from the following files:

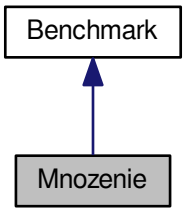
- lista.hh
- lista.cpp

5.9 Mnozenie Class Reference

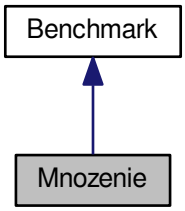
Klasa Mnozenie modelujaca algorytm mnozenia. Obiekt tego typu reprezentuje algorytm wykonujacy dzialanie mnozenia kazdego elementu tablicy tab przez 2.

`#include <algorithm1.hh>`

Inheritance diagram for Mnozenie:



Collaboration diagram for Mnozenie:



Public Member Functions

- Mnozenie ()

  Konstruktor obiektu Mnozenie.
- Mnozenie (int _tab[ ])

  Konstruktor parametryczny obiektu Mnozenie.
- ∼Mnozenie ()

  Destruktor obiektu Mnozenie.
- virtual void testAlgorithm (Benchmark ∗_algorithm)

  Metoda testowania algorytmu. Metoda sluzy do testowania szybkosci dzialania algorytmu. W klasie Mnozenie nie ma konkretnego dzialania.
- virtual void runAlgorithm (int _border)

  Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Mnozy kazdy element tablicy przez liczbe 2.

Private Attributes

- int tab [SIZE]

  Tablica elementow z danymi wejsciowymi.

5.9.1 Detailed Description

Definition at line 9 of file algorithm1.hh.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Mnozenie::Mnozenie ( ) [inline]

Definition at line 20 of file algorithm1.hh.

5.9.2.2 Mnozenie::Mnozenie (int *_tab[ ]*) [inline]

Parameters

in	*_tab*	- tablica przechowujaca dane wejsciowe.

Definition at line 26 of file algorithm1.hh.

5.9.2.3 Mnozenie::~Mnozenie ( ) [inline]

Definition at line 31 of file algorithm1.hh.

5.9.3 Member Function Documentation

5.9.3.1 void Mnozenie::runAlgorithm (int *_border*) [virtual]

Parameters

in	*_border*	- ilosc elementow dla ktorych algorytm ma wykonac swoje dzialanie.

Reimplemented from Benchmark.

Definition at line 7 of file algorithm1.cpp.

5.9.3.2 virtual void Mnozenie::testAlgorithm (**Benchmark** * *_algorithm*) [inline], [virtual]

Parameters

in	*_algorithm*	- testowany algorytm.

Definition at line 39 of file algorithm1.hh.

5.9.4 Member Data Documentation

5.9.4.1 int Mnozenie::tab[SIZE] [private]

Definition at line 14 of file algorithm1.hh.

The documentation for this class was generated from the following files:

- algorithm1.hh
- algorithm1.cpp

5.10 Stos Class Reference

Klasa Stos modelujaca strukture danych typu stos. Obiekt tego typu reprezentuje strukture danych typu stos wraz z operacjami mozliwymi do wykonania na tej strukturze.

`#include <stos.hh>`

Public Member Functions

- Stos ()

  Konstruktor obiektu Stos.
- Stos (long _size)

  Konstruktor parametryczny obiektu Stos.
- ~Stos ()

  Destruktor obiektu Stos.
- void push (int _elem)

  Metoda dodawnia elementu. Metoda sluzy do dodawania elementu do stosu.
- int pop ()

  Metoda usuwania elementu. Metoda sluzy do usuwania elementu ze stosu.

Private Member Functions

- void increase ()

  Metoda powiekszania stosu. Metoda ta dodaje do stosu 8 kolejnych wolnych pol.
- int decrease ()

  Metoda pomniejszania stosu. Metoda ta odejmuje od stosu jedno pole.

Private Attributes

- int * tab

  Wskaznik na tablice elementow stosu.
- long size

  Rozmiar stosu.
- long last

  Indeks ostatniego wolnego pola.

5.10.1 Detailed Description

Definition at line 9 of file stos.hh.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Stos::Stos ( )

Definition at line 5 of file stos.cpp.

5.10.2.2 Stos::Stos (long *_size*)

Parameters

in	*_size*	- rozmiar tworzonego stosu.

Definition at line 14 of file stos.cpp.

5.10.2.3 Stos::~Stos ( )

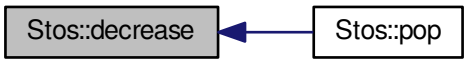
Definition at line 23 of file stos.cpp.

5.10.3 Member Function Documentation

5.10.3.1 int Stos::decrease ( ) [private]

Definition at line 53 of file stos.cpp.

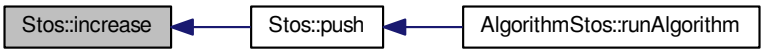
Here is the caller graph for this function:



5.10.3.2 void Stos::increase ( ) [private]

Definition at line 42 of file stos.cpp.

Here is the caller graph for this function:



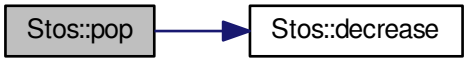
5.10.3.3 int Stos::pop ( )

Returns

- usuwany element.

Definition at line 35 of file stos.cpp.

Here is the call graph for this function:



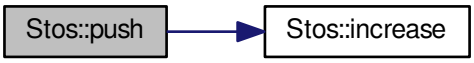
5.10.3.4 void Stos::push (int *_elem*)

Parameters

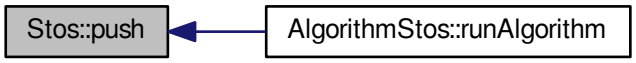
in	*_elem*	- dodawany element.

Definition at line 27 of file stos.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.4 Member Data Documentation

5.10.4.1 long Stos::last [private]

Definition at line 22 of file stos.hh.

5.10.4.2 long Stos::size [private]

Definition at line 18 of file stos.hh.

5.10.4.3 int* Stos::tab [private]

Definition at line 14 of file stos.hh.

The documentation for this class was generated from the following files:

- stos.hh
- stos.cpp

5.11 TabLista Class Reference

Klasa TabLista modelujaca strukture danych typu lista. Obiekt tego typu reprezentuje strukture danych typu lista zaimplementowana na tablicy dynamicznej. Obiekt zawiera rowniez podstowe metody listy.

`#include <tab_lista.hh>`

Public Member Functions

- TabLista ()

  Konstruktor obiektu TabLista.
- TabLista (long _size)

  Konstruktor parametryczny obiektu TabLista.
- ~TabLista ()

  Destruktor obiektu TabLista.
- void insert (int _elem)

  Metoda dodawnia elementu. Metoda sluzy do dodawania elementu do konca listy.
- int remove (int _f)

  Metoda usuwania elementu. Metoda sluzy do usuwania elementu o wskazanym indeksie.
- int rozmiar ()

  Metoda zwracania indeksu ostatniego elementu tablicy. Metoda sluzy do zwrócenia ilości elementow tablicy.
- void quicksort (int a, int b)

  Metoda sortująca. Metoda sluzy do sortowania przedziału tablicy.

Private Member Functions

- void increase ()

  Metoda powiekszania listy tablicowe. Metoda ta dodaje do listy 8 kolejnych wolnych pol.

Private Attributes

- int * tab

  Wskaznik na tablice elementow listy.
- long size

  Rozmiar listy.
- long last

  Wskaznik na ostatni wolny element.

5.11.1 Detailed Description

Definition at line 10 of file tab_lista.hh.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 TabLista::TabLista ( )

Definition at line 5 of file tab_lista.cpp.

5.11.2.2 TabLista::TabLista (long *_size*)

Definition at line 13 of file tab_lista.cpp.

5.11.2.3 TabLista::∼TabLista ( )

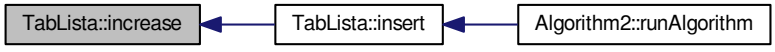
Definition at line 21 of file tab_lista.cpp.

5.11.3 Member Function Documentation

5.11.3.1 void TabLista::increase ( ) [private]

Definition at line 25 of file tab_lista.cpp.

Here is the caller graph for this function:



5.11.3.2 void TabLista::insert (int *_elem*)

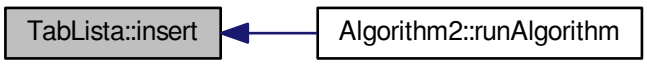
Parameters

in	*_elem*	- dodawany element.

Definition at line 35 of file tab_lista.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



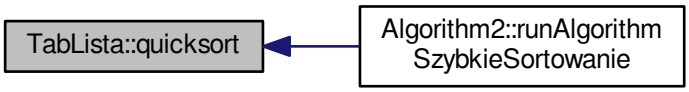
5.11.3.3 void TabLista::quicksort (int *a*, int *b*)

Parameters

in	*a*	- lewa granica przedziału sortowania.
in	*b*	- prawa granica przedziału sortowania.

Definition at line 57 of file tab_lista.cpp.

Here is the caller graph for this function:



5.11.3.4 int TabLista::remove (int *_f*)

Parameters

in	*_f*	- indeks elementu do usuniecia.

Returns

- usuwany element.

Definition at line 42 of file tab_lista.cpp.

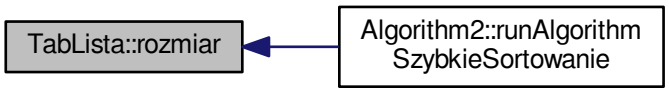
5.11.3.5 int TabLista::rozmiar ( )

Returns

- indeks ostatniego elementu.

Definition at line 133 of file tab_lista.cpp.

Here is the caller graph for this function:



5.11.4 Member Data Documentation

5.11.4.1 long TabLista::last [private]

Definition at line 23 of file tab_lista.hh.

5.11.4.2 long TabLista::size [private]

Definition at line 19 of file tab_lista.hh.

5.11.4.3 int* TabLista::tab [private]

Definition at line 15 of file tab_lista.hh.

The documentation for this class was generated from the following files:

- tab_lista.hh
- tab_lista.cpp

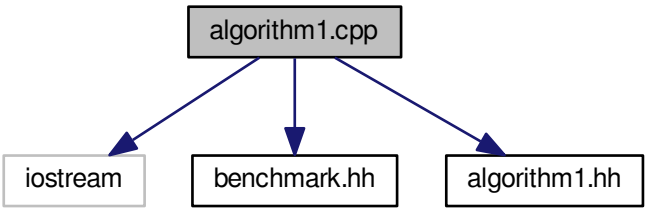
Chapter 6

File Documentation

6.1 algorithm1.cpp File Reference

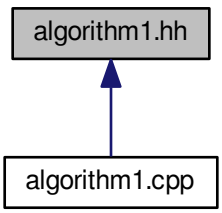
```
#include <iostream>
#include "benchmark.hh"
#include "algorithm1.hh"
```

Include dependency graph for algorithm1.cpp:



6.2 algorithm1.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

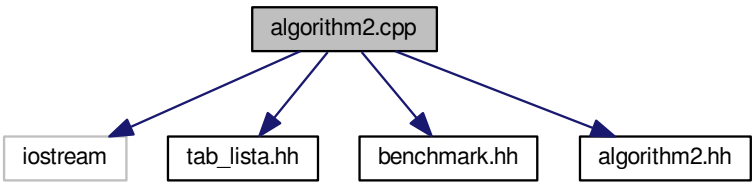
- class Mnozenie

  Klasa Mnozenie modelujaca algorytm mnozenia. Obiekt tego typu reprezentuje algorytm wykonujacy dzialanie mnozenia kazdego elementu tablicy tab przez 2.

6.3 algorithm2.cpp File Reference

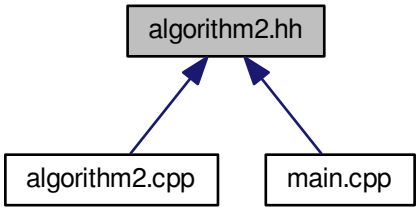
```
#include <iostream>
#include "tab_lista.hh"
#include "benchmark.hh"
#include "algorithm2.hh"
```

Include dependency graph for algorithm2.cpp:



6.4 algorithm2.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

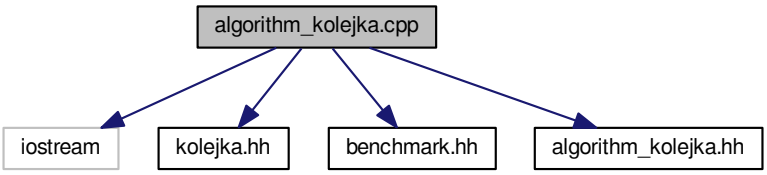
- class Algorithm2

  Klasa Algorithm2 modelujaca algorytm wczytywania do listy tablicowej. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do listy tablicowej.

6.5 algorithm_kolejka.cpp File Reference

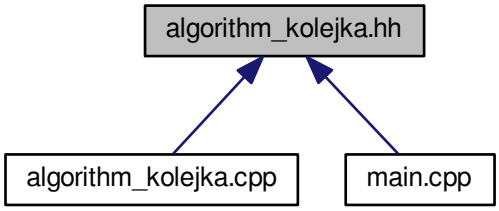
```
#include <iostream>
#include "kolejka.hh"
#include "benchmark.hh"
#include "algorithm_kolejka.hh"
```

Include dependency graph for algorithm_kolejka.cpp:



6.6 algorithm_kolejka.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

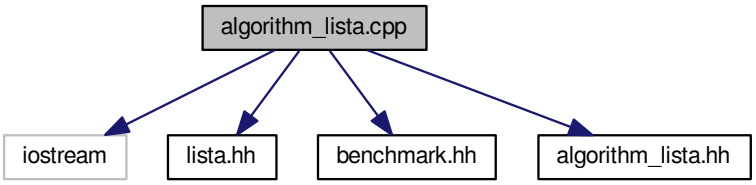
- class AlgorithmKolejka

  Klasa AlgorithmKolejka modelujaca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do kolejki.

6.7 algorithm_lista.cpp File Reference

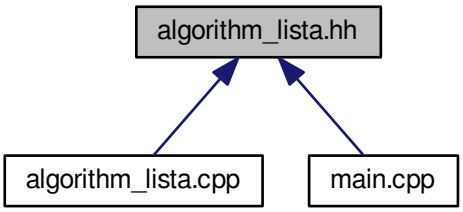
```
#include <iostream>
#include "lista.hh"
#include "benchmark.hh"
#include "algorithm_lista.hh"
```

Include dependency graph for algorithm_lista.cpp:



6.8 algorithm_lista.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

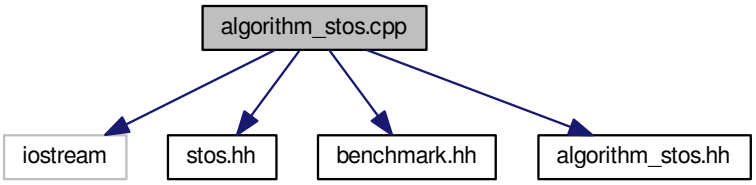
- class AlgorithmLista

  Klasa AlgorithmLista modelujaca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do listy.

6.9 algorithm_stos.cpp File Reference

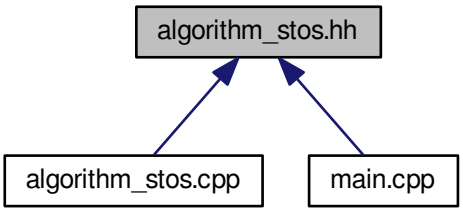
```
#include <iostream>
#include "stos.hh"
#include "benchmark.hh"
#include "algorithm_stos.hh"
```

Include dependency graph for algorithm_stos.cpp:



6.10 algorithm_stos.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

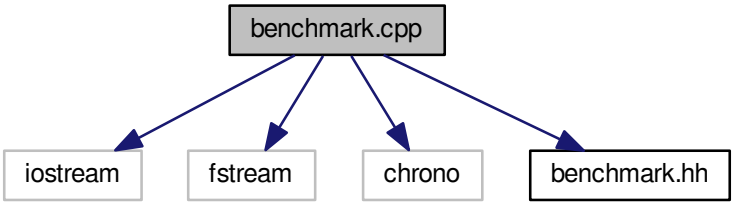
- class AlgorithmStos

  Klasa AlgorithmStos modelujaca algorytm wczytywania do stosu. Obiekt tego typu reprezentuje algorytm wykonujacy wykonujacy wczytywanie zadanej ilosci elementow do stosu.

6.11 benchmark.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <chrono>
#include "benchmark.hh"
```

Include dependency graph for benchmark.cpp:



Macros

- #define LENGTH 100
- #define REPEATS 4

6.11.1 Macro Definition Documentation

6.11.1.1 #define LENGTH 100

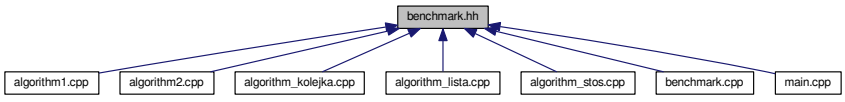
Definition at line 7 of file benchmark.cpp.

6.11.1.2 #define REPEATS 4

Definition at line 8 of file benchmark.cpp.

6.12 benchmark.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Benchmark

  Klasa Benchmark modelujaca program benchmarkujacy. Obiekt tego typu reprezentuje program sprawdzajacy szybkosc wykonywania algorytmow.

Macros

- #define SIZE 10000

6.12.1 Macro Definition Documentation

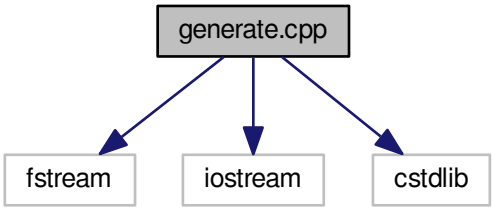
6.12.1.1 #define SIZE 10000

Definition at line 4 of file benchmark.hh.

6.13 generate.cpp File Reference

```
#include <fstream>
#include <iostream>
#include <cstdlib>
```

Include dependency graph for generate.cpp:



Macros

- #define SIZE 100000

Functions

- int main ()

  Funkcja generowania pliku z danymi wejsciowymi. Generuje liczby losowe od 1 do 51 i zapisuje je do pliku o nazwie data.txt.

6.13.1 Macro Definition Documentation

6.13.1.1 #define SIZE 100000

Definition at line 5 of file generate.cpp.

6.13.2 Function Documentation

6.13.2.1 int main ( )

Return values

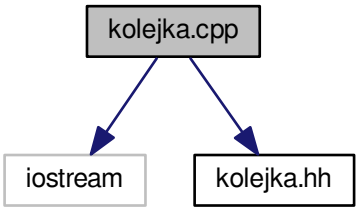
0	- gdy funkcja zadziala poprawnie.
1	- gdy wystapi blad otwarcia pliku do zapisu.

Definition at line 14 of file generate.cpp.

6.14 kolejka.cpp File Reference

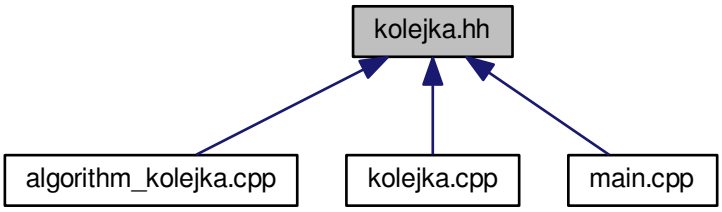
```
#include <iostream>
#include "kolejka.hh"
```

Include dependency graph for kolejka.cpp:



6.15 kolejka.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

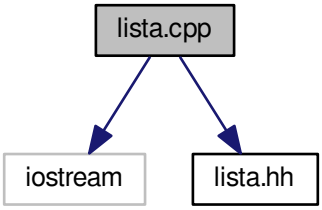
- class Kolejka

  Klasa Kolejka modelujaca strukture danych typu kolejka. Obiekt tego typu reprezentuje strukture danych typu kolejka wraz z operacjami mozliwymi do wykonania na tej strukturze.

6.16 lista.cpp File Reference

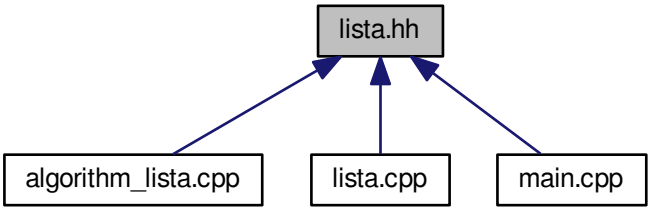
```
#include <iostream>
#include "lista.hh"
```

Include dependency graph for lista.cpp:



6.17 lista.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Lista

  Klasa Lista modelujaca strukture danych typu lista. Obiekt tego typu reprezentuje strukture danych typu lista wraz z operacjami mozliwymi do wykonania na tej strukturze.

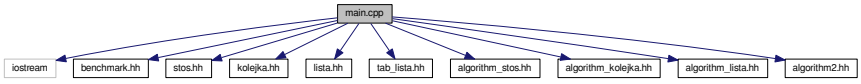
- struct Lista::Komorka

  Struktura Komorka. Obiekt tego typu reprezentuje pojedyncza komorke wraz ze wskaznikiem na nastepna komorke listy.

6.18 main.cpp File Reference

```
#include <iostream>
#include "benchmark.hh"
#include "stos.hh"
#include "kolejka.hh"
#include "lista.hh"
#include "tab_lista.hh"
#include "algorithm_stos.hh"
#include "algorithm_kolejka.hh"
#include "algorithm_lista.hh"
#include "algorithm2.hh"
```

Include dependency graph for main.cpp:



Functions

- int main ()

  Funkcja tworzaca i testujaca algorytm. Wczytuje dane otrzymane na strumien wejsciowy do tablicy data[]. Nastepnie tworzy obiekt Benchmark oraz obiekt Potegowanie. Pozniej uruchamia metode testujaca w obiekcie klasy Benchmark dla obiektu klasy Potegowanie.

6.18.1 Function Documentation

6.18.1.1 int main ( )

Return values

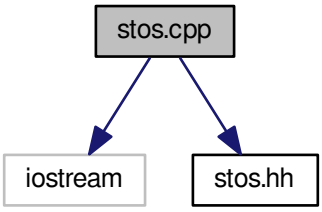
0	- domyslna wartosc zwracana przez funkcje.

Definition at line 21 of file main.cpp.

6.19 stos.cpp File Reference

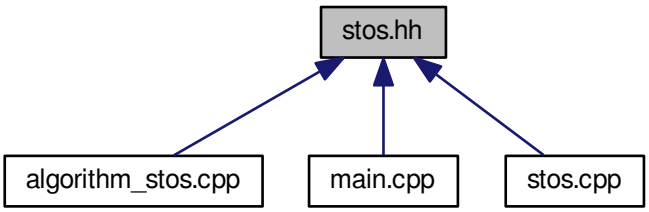
```
#include <iostream>
#include "stos.hh"
```

Include dependency graph for stos.cpp:



6.20 stos.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Stos

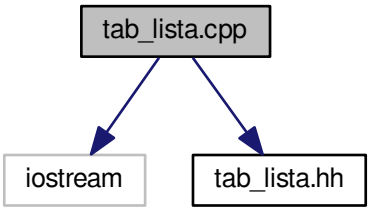
  Klasa Stos modelujaca strukture danych typu stos. Obiekt tego typu reprezentuje strukture danych typu stos wraz z operacjami mozliwymi do wykonania na tej strukturze.

6.21 strona-glowna.dox File Reference

6.22 tab_lista.cpp File Reference

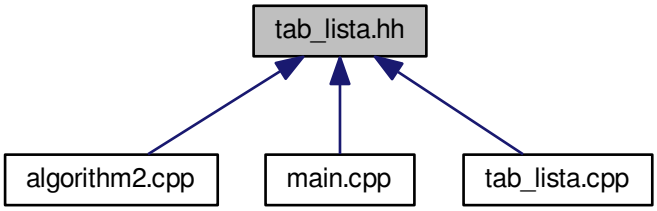
```
#include <iostream>
#include "tab_lista.hh"
```

Include dependency graph for tab_lista.cpp:



6.23 tab_lista.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class TabLista

  Klasa TabLista modelujaca strukture danych typu lista. Obiekt tego typu reprezentuje strukture danych typu lista zaimplementowana na tablicy dynamicznej. Obiekt zawiera rowniez podstowe metody listy.

Index