

lab4
0.0004

Generated by Doxygen 1.8.9.1

Thu Apr 16 2015 04:48:47

Contents

1	Laboratorium 4	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	Algorithm2 Class Reference	9
5.1.1	Detailed Description	10
5.1.2	Constructor & Destructor Documentation	10
5.1.2.1	Algorithm2	10
5.1.2.2	Algorithm2	10
5.1.2.3	~Algorithm2	10
5.1.3	Member Function Documentation	10
5.1.3.1	runAlgorithm	10
5.1.3.2	runAlgorithmSzybkieSortowanie	11
5.1.3.3	testAlgorithm	11
5.1.4	Member Data Documentation	11
5.1.4.1	tab	11
5.1.4.2	tablista	11
5.2	AlgorithmKolejka Class Reference	12
5.2.1	Detailed Description	13
5.2.2	Constructor & Destructor Documentation	13
5.2.2.1	AlgorithmKolejka	13
5.2.2.2	AlgorithmKolejka	13
5.2.2.3	~AlgorithmKolejka	13
5.2.3	Member Function Documentation	13
5.2.3.1	runAlgorithm	13

5.2.3.2	testAlgorithm	14
5.2.4	Member Data Documentation	14
5.2.4.1	kolejka	14
5.2.4.2	tab	14
5.3	AlgorithmLista Class Reference	14
5.3.1	Detailed Description	15
5.3.2	Constructor & Destructor Documentation	15
5.3.2.1	AlgorithmLista	15
5.3.2.2	AlgorithmLista	16
5.3.2.3	~AlgorithmLista	16
5.3.3	Member Function Documentation	16
5.3.3.1	runAlgorithm	16
5.3.3.2	testAlgorithm	16
5.3.4	Member Data Documentation	16
5.3.4.1	lista	16
5.3.4.2	tab	16
5.4	AlgorithmStos Class Reference	17
5.4.1	Detailed Description	18
5.4.2	Constructor & Destructor Documentation	18
5.4.2.1	AlgorithmStos	18
5.4.2.2	AlgorithmStos	18
5.4.2.3	~AlgorithmStos	18
5.4.3	Member Function Documentation	18
5.4.3.1	runAlgorithm	18
5.4.3.2	testAlgorithm	19
5.4.4	Member Data Documentation	19
5.4.4.1	stos	19
5.4.4.2	tab	19
5.5	Benchmark Class Reference	19
5.5.1	Detailed Description	20
5.5.2	Constructor & Destructor Documentation	20
5.5.2.1	Benchmark	20
5.5.2.2	~Benchmark	20
5.5.3	Member Function Documentation	20
5.5.3.1	runAlgorithm	20
5.5.3.2	runAlgorithmSzybkieSortowanie	20
5.5.3.3	testAlgorithm	21
5.5.4	Member Data Documentation	21
5.5.4.1	nazwy	21
5.6	Kolejka Class Reference	21

5.6.1	Detailed Description	22
5.6.2	Constructor & Destructor Documentation	22
5.6.2.1	Kolejka	22
5.6.2.2	Kolejka	22
5.6.2.3	~Kolejka	22
5.6.3	Member Function Documentation	23
5.6.3.1	dequeue	23
5.6.3.2	enqueue	23
5.6.3.3	increase	23
5.6.4	Member Data Documentation	24
5.6.4.1	f	24
5.6.4.2	r	24
5.6.4.3	size	24
5.6.4.4	tab	24
5.7	Lista::Komorka Struct Reference	24
5.7.1	Detailed Description	25
5.7.2	Constructor & Destructor Documentation	25
5.7.2.1	Komorka	25
5.7.3	Member Data Documentation	25
5.7.3.1	elem	25
5.7.3.2	next	25
5.8	Lista Class Reference	25
5.8.1	Detailed Description	26
5.8.2	Constructor & Destructor Documentation	26
5.8.2.1	Lista	26
5.8.2.2	~Lista	26
5.8.3	Member Function Documentation	26
5.8.3.1	insert	26
5.8.3.2	remove	27
5.8.4	Member Data Documentation	27
5.8.4.1	head	27
5.8.4.2	tail	27
5.9	Mnozenie Class Reference	27
5.9.1	Detailed Description	29
5.9.2	Constructor & Destructor Documentation	29
5.9.2.1	Mnozenie	29
5.9.2.2	Mnozenie	29
5.9.2.3	~Mnozenie	29
5.9.3	Member Function Documentation	29
5.9.3.1	runAlgorithm	29

5.9.3.2	testAlgorithm	29
5.9.4	Member Data Documentation	29
5.9.4.1	tab	29
5.10	Stos Class Reference	30
5.10.1	Detailed Description	30
5.10.2	Constructor & Destructor Documentation	30
5.10.2.1	Stos	30
5.10.2.2	Stos	30
5.10.2.3	~Stos	31
5.10.3	Member Function Documentation	31
5.10.3.1	decrease	31
5.10.3.2	increase	31
5.10.3.3	pop	31
5.10.3.4	push	32
5.10.4	Member Data Documentation	32
5.10.4.1	last	32
5.10.4.2	size	33
5.10.4.3	tab	33
5.11	TabLista Class Reference	33
5.11.1	Detailed Description	34
5.11.2	Constructor & Destructor Documentation	34
5.11.2.1	TabLista	34
5.11.2.2	TabLista	34
5.11.2.3	~TabLista	34
5.11.3	Member Function Documentation	34
5.11.3.1	increase	34
5.11.3.2	insert	34
5.11.3.3	quicksort	35
5.11.3.4	remove	35
5.11.3.5	rozmiar	35
5.11.4	Member Data Documentation	36
5.11.4.1	last	36
5.11.4.2	size	36
5.11.4.3	tab	36
6	File Documentation	37
6.1	algorithm1.cpp File Reference	37
6.2	algorithm1.hh File Reference	37
6.3	algorithm2.cpp File Reference	38
6.4	algorithm2.hh File Reference	38

6.5	algorithm_kolejka.cpp File Reference	39
6.6	algorithm_kolejka.hh File Reference	39
6.7	algorithm_lista.cpp File Reference	39
6.8	algorithm_lista.hh File Reference	40
6.9	algorithm_stos.cpp File Reference	40
6.10	algorithm_stos.hh File Reference	41
6.11	benchmark.cpp File Reference	41
6.11.1	Macro Definition Documentation	42
6.11.1.1	LENGTH	42
6.11.1.2	REPEATS	42
6.12	benchmark.hh File Reference	42
6.12.1	Macro Definition Documentation	43
6.12.1.1	SIZE	43
6.13	generate.cpp File Reference	43
6.13.1	Macro Definition Documentation	43
6.13.1.1	SIZE	43
6.13.2	Function Documentation	43
6.13.2.1	main	43
6.14	kolejka.cpp File Reference	44
6.15	kolejka.hh File Reference	44
6.16	lista.cpp File Reference	45
6.17	lista.hh File Reference	45
6.18	main.cpp File Reference	46
6.18.1	Function Documentation	46
6.18.1.1	main	46
6.19	stos.cpp File Reference	46
6.20	stos.hh File Reference	47
6.21	strona-glowna.dox File Reference	47
6.22	tab_lista.cpp File Reference	47
6.23	tab_lista.hh File Reference	48
	Index	49

Chapter 1

Laboratorium 4

Author

Filip Malinowski (Contributor 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:

Benchmark	19
Algorithm2	9
AlgorithmKolejka	12
AlgorithmLista	14
AlgorithmStos	17
Mnozenie	27
Kolejka	21
Lista::Komorka	24
Lista	25
Stos	30
TabLista	33

Chapter 3

Class Index

3.1 Class List

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

Algorithm2	Klasa Algorithm2 modelująca algorytm wczytywania do listy tablicowej. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do listy tablicowej	9
AlgorithmKolejka	Klasa AlgorithmKolejka modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do kolejki	12
AlgorithmLista	Klasa AlgorithmLista modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do listy	14
AlgorithmStos	Klasa AlgorithmStos modelująca algorytm wczytywania do stosu. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do stosu	17
Benchmark	Klasa Benchmark modelująca program benchmarkujący. Obiekt tego typu reprezentuje program sprawdzający szybkość wykonywania algorytmów	19
Kolejka	Klasa Kolejka modelująca strukturę danych typu kolejka. Obiekt tego typu reprezentuje strukturę danych typu kolejka wraz z operacjami możliwymi do wykonania na tej strukturze	21
Lista::Komorka	Struktura Komorka . Obiekt tego typu reprezentuje pojedynczą komórkę wraz ze wskaźnikiem na następną komórkę listy	24
Lista	Klasa Lista modelująca strukturę danych typu lista. Obiekt tego typu reprezentuje strukturę danych typu lista wraz z operacjami możliwymi do wykonania na tej strukturze	25
Mnozenie	Klasa Mnozenie modelująca algorytm mnożenia. Obiekt tego typu reprezentuje algorytm wykonujący działanie mnożenia każdego elementu tablicy tab przez 2	27
Stos	Klasa Stos modelująca strukturę danych typu stos. Obiekt tego typu reprezentuje strukturę danych typu stos wraz z operacjami możliwymi do wykonania na tej strukturze	30
TabLista	Klasa TabLista modelująca strukturę danych typu lista. Obiekt tego typu reprezentuje strukturę danych typu lista zaimplementowaną na tablicy dynamicznej. Obiekt zawiera również podstawowe metody listy	33

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

algorithm1.cpp	37
algorithm1.hh	37
algorithm2.cpp	38
algorithm2.hh	38
algorithm_kolejka.cpp	39
algorithm_kolejka.hh	39
algorithm_lista.cpp	39
algorithm_lista.hh	40
algorithm_stos.cpp	40
algorithm_stos.hh	41
benchmark.cpp	41
benchmark.hh	42
generate.cpp	43
kolejka.cpp	44
kolejka.hh	44
lista.cpp	45
lista.hh	45
main.cpp	46
stos.cpp	46
stos.hh	47
tab_lista.cpp	47
tab_lista.hh	48

Chapter 5

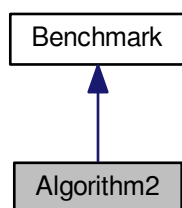
Class Documentation

5.1 Algorithm2 Class Reference

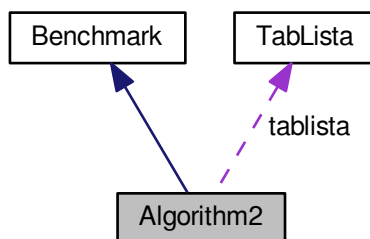
Klasa [Algorithm2](#) modelująca algorytm wczytywania do listy tablicowej. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do listy tablicowej.

```
#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 sluzi do testowania szybkoosci dzialania algorytmu. W klasie [Algorithm2](#) nie ma konkretnego dzialania.
- virtual void [runAlgorithm](#) (int _border)
Metoda uruchamiania algorytmu. Metoda sluzi to wykonywania danego algorytmu. Wczytuje elementy do kolejki.
- virtual void [runAlgorithmSzybkieSortowanie](#) ()
Metoda uruchamiania algorytmu szybkiego sortowania. Metoda sluzi to wykonywania danego algorytmu. Sortuje tablice.

Private Attributes

- int [tab](#) [[SIZE](#)]
Tablica elementow z danymi wejscowymi.
- [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 wejscowe.
----	----------------------	--

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](#)]

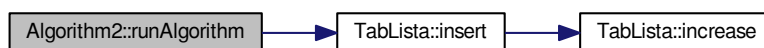
Parameters

<code>in</code>	<code>_border</code>	- ilosc elementow dla ktorzych algorytm ma wykonac swoje dzialanie.
-----------------	----------------------	---

Reimplemented from [Benchmark](#).

Definition at line 8 of file algorithm2.cpp.

Here is the call graph for this function:

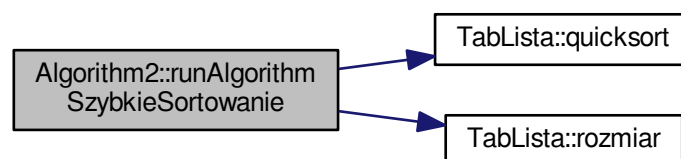


5.1.3.2 void Algorithm2::runAlgorithmSzybkieSortowanie () [virtual]

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

<code>in</code>	<code>_algorithm</code>	- 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:

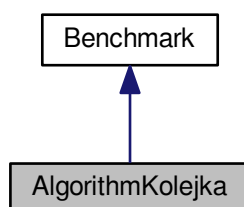
- [algorithm2.hh](#)
- [algorithm2.cpp](#)

5.2 AlgorithmKolejka Class Reference

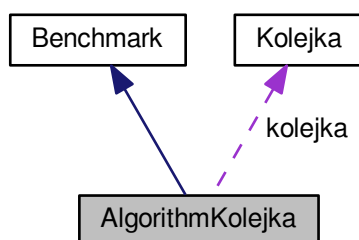
Klasa [AlgorithmKolejka](#) modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do kolejki.

```
#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 szybkości działania algorytmu. W klasie [AlgorithmKolejka](#) nie ma konkretnego działania.

- 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

<code>in</code>	<code>_tab</code>	- 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

<code>in</code>	<code>_border</code>	- 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

<code>in</code>	<code>_algorithm</code>	- 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:

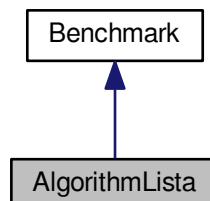
- [algorithm_kolejka.hh](#)
- [algorithm_kolejka.cpp](#)

5.3 AlgorithmLista Class Reference

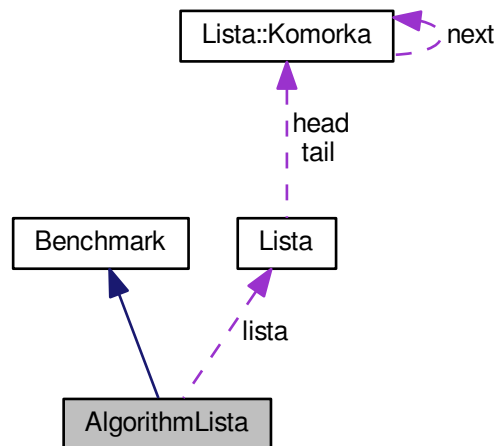
Klasa [AlgorithmLista](#) modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do listy.

```
#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 sluzi do testowania szybkoosci dzialania algorytmu. W klasie [AlgorithmLista](#) nie ma konkretnego dzialania.
- virtual void [runAlgorithm](#) (int _border)
Metoda uruchamiania algorytmu. Metoda sluzi 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

<i>in</i>	<i>_tab</i>	- tablica przechowująca dane wejściowe.
-----------	-------------	---

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]

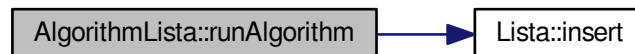
Parameters

<i>in</i>	<i>_border</i>	- ilość elementów dla których algorytm ma wykonać swoje działanie.
-----------	----------------	--

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

<i>in</i>	<i>_algorithm</i>	- 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:

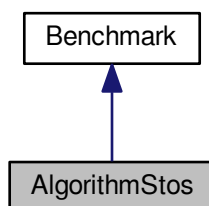
- [algorithm_lista.hh](#)
- [algorithm_lista.cpp](#)

5.4 AlgorithmStos Class Reference

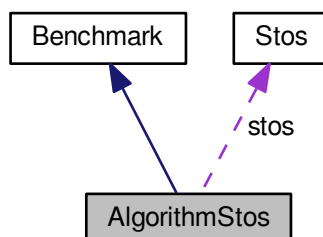
Klasa [AlgorithmStos](#) modelująca algorytm wczytywania do stosu. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do stosu.

```
#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 szybkości działania algorytmu. W klasie [AlgorithmStos](#) nie ma konkretnego działania.

- virtual void [runAlgorithm](#) (int `_border`)

Metoda uruchamiania algorytmu. Metoda sluzi 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

<code>in</code>	<code>_tab</code>	- 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

<code>in</code>	<code>_border</code>	- 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	<code>_algorithm</code>	- 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:

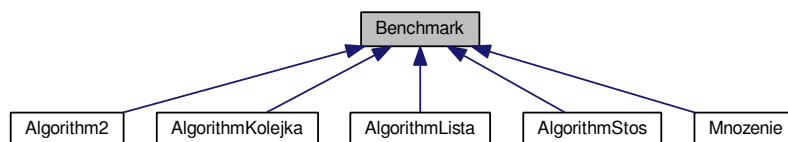
- [algorithm_stos.hh](#)
- [algorithm_stos.cpp](#)

5.5 Benchmark Class Reference

Klasa [Benchmark](#) modelująca program benchmarkujący. Obiekt tego typu reprezentuje program sprawdzający szybkość wykonywania algorytmów.

```
#include <benchmark.hh>
```

Inheritance diagram for [Benchmark](#):



Public Member Functions

- [Benchmark](#) ()
Konstruktor obiektu [Benchmark](#).
- [~Benchmark](#) ()
Destruktor obiektu [Benchmark](#).
- `virtual void testAlgorithm (Benchmark *_algorithm, int _n) const`
Metoda testowania algorytmu. Metoda służy do testowania szybkości działania algorytmu. Wykonuje testowany algorytm dla 5 kolejnych ilości elementów. Wykonanie algorytmu dla danego zestawu liczb powtarza dwa razy i uśrednia wynik. Otrzymany czas wraz z ilością testowanych danych zapisuje w pliku `ret_data.txt`.
- `virtual void runAlgorithm (int _border)`
Metoda uruchamiania algorytmu. Metoda służy do wykonywania danego algorytmu. W klasie [Benchmark](#) nie ma konkretnego działania.

- 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:

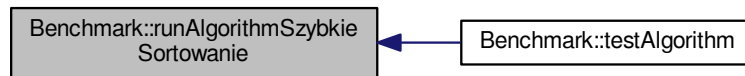


5.5.3.2 [virtual void Benchmark::runAlgorithmSzybkieSortowanie](#) () `[inline], [virtual]`

Reimplemented in [Algorithm2](#).

Definition at line 54 of file benchmark.hh.

Here is the caller graph for this function:



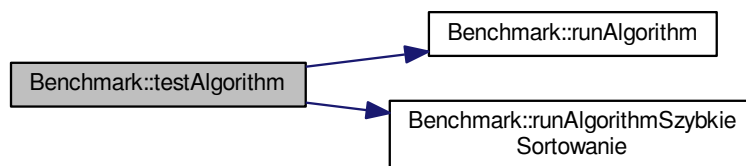
5.5.3.3 `void Benchmark::testAlgorithm (Benchmark * _algorithm, int _n) const` `[virtual]`

Parameters

in	<code>_algorithm</code>	- testowany algorytm.
in	<code>_n</code>	- 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](#) modelująca strukturę danych typu kolejka. Obiekt tego typu reprezentuje strukturę danych typu kolejka wraz z operacjami możliwymi 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 dodawania 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](#)
Wskaźnik 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

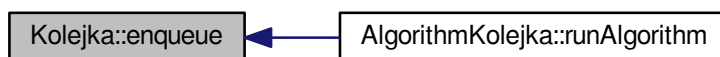
in	<i>_elem</i>	- dodawany element.
----	--------------	---------------------

Definition at line 40 of file kolejka.cpp.

Here is the call graph for this function:



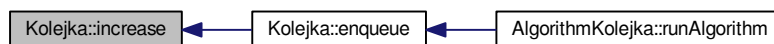
Here is the caller graph for this function:



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.

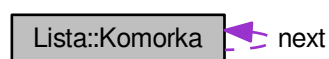
The documentation for this class was generated from the following files:

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

5.7 Lista::Komorka Struct Reference

Struktura [Komorka](#). Obiekt tego typu reprezentuje pojedyncza komorkę wraz ze wskaźnikiem na następną komorkę listy.

Collaboration diagram for Lista::Komorka:



Public Member Functions

- [Komorka](#) (`int _elem`)
Konstruktor parametryczny obiektu [Komorka](#).

Public Attributes

- `int elem`
Wartość elementu w pojedynczej komorce.
- [Komorka](#) * `next`
Wskaźnik na następną komorkę 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

<i>in</i>	<i>_elem</i>	- 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:

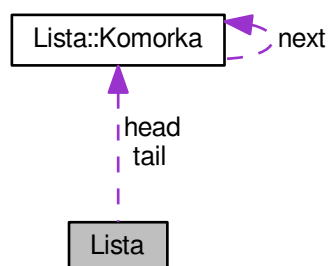
- [lista.hh](#)

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 komorkę wraz ze wskaźnikiem na następną komorkę listy.

Public Member Functions

- [Lista](#) ()

Konstruktor obiektu [Lista](#).

- [~Lista](#) ()

Destruktor obiektu [Lista](#).

- void [insert](#) (int _elem)

Metoda dodawania elementu. Metoda służy do dodawania elementu do końca listy.

- int [remove](#) (int _f)

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

Private Attributes

- [Komorka](#) * [head](#)

Wskaźnik na początek listy.

- [Komorka](#) * [tail](#)

Wskaźnik 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

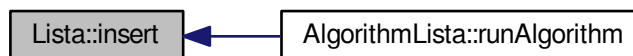
5.8.3.1 void [Lista::insert](#) (int *_elem*)

Parameters

<i>in</i>	<i>_elem</i>	- 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

<code>in</code>	<code>_f</code>	- 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:

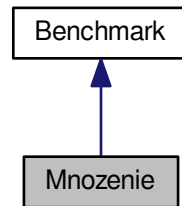
- [lista.hh](#)
- [lista.cpp](#)

5.9 Mnozenie Class Reference

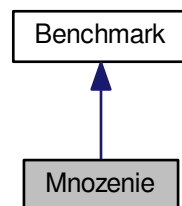
Klasa [Mnozenie](#) modelujaca algorytm mnozenia. Obiekt tego typu reprezentuje algorytm wykonujacy dzialanie mnozenia kazdego elementu tablicy tab przez 2.

```
#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 szybkości działania algorytmu. W klasie `Mnozenie` nie ma konkretnego działania.
- `virtual void runAlgorithm (int _border)`
Metoda uruchamiania algorytmu. Metoda sluzy to wykonywania danego algorytmu. Mnozy każdy element tablicy przez liczbę 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 przechowująca dane wejściowe.
----	------	---

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	- ilość elementów dla których algorytm ma wykonać swoje działanie.
----	---------	--

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](#) modelująca strukture danych typu stos. Obiekt tego typu reprezentuje strukture danych typu stos wraz z operacjami możliwymi 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 dodawania 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](#)
Wskaźnik 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

<code>in</code>	<code>_size</code>	- 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**

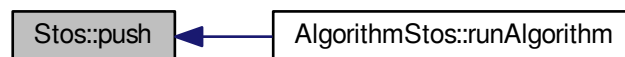
<code>in</code>	<code>_elem</code>	- dodawany element.
-----------------	--------------------	---------------------

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](#) modelująca strukturę danych typu lista. Obiekt tego typu reprezentuje strukturę danych typu lista zaimplementowana na tablicy dynamicznej. Obiekt zawiera również podstawowe 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 dodawania elementu. Metoda służy do dodawania elementu do końca listy.
- int [remove](#) (int _f)
Metoda usuwania elementu. Metoda służy do usuwania elementu o wskazanym indeksie.
- int [rozmiar](#) ()
Metoda zwracania indeksu ostatniego elementu tablicy. Metoda służy do zwrócenia ilości elementów tablicy.
- void [quicksort](#) (int a, int b)
Metoda sortująca. Metoda służy do sortowania przedziału tablicy.

Private Member Functions

- void [increase](#) ()
Metoda powiększania listy tablicowe. Metoda ta dodaje do listy 8 kolejnych wolnych pól.

Private Attributes

- int * [tab](#)
Wskaźnik na tablicę elementów listy.
- long [size](#)
Rozmiar listy.
- long [last](#)
Wskaźnik 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 ()

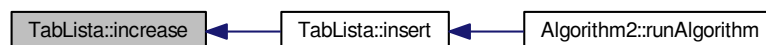
Definition at line 21 of file tab_lista.cpp.

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

<code>in</code>	<code>_elem</code>	- 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:



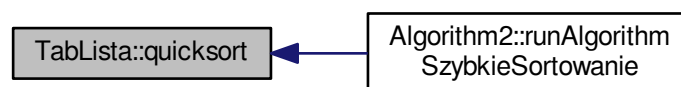
5.11.3.3 void TabLista::quicksort (int *a*, int *b*)

Parameters

in	<i>a</i>	- lewa granica przedziału sortowania.
in	<i>b</i>	- 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	<i>_f</i>	- 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:

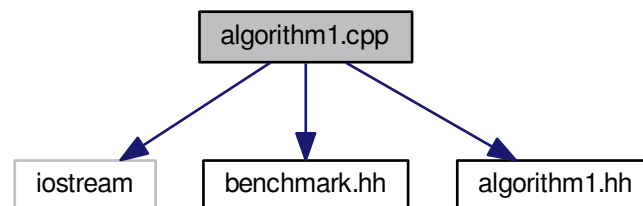
- [tab_lista.hh](#)
- [tab_lista.cpp](#)

Chapter 6

File Documentation

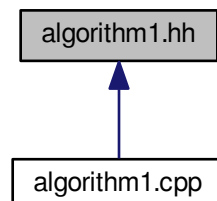
6.1 algorithm1.cpp File Reference

```
#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:



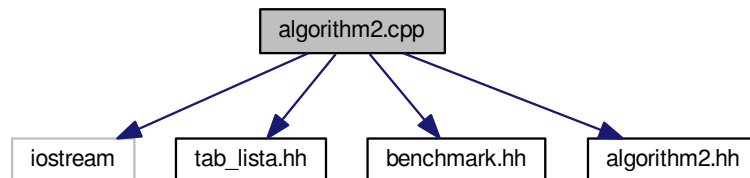
Classes

- class [Mnozenie](#)

Klasa [Mnozenie](#) modelująca algorytm mnożenia. Obiekt tego typu reprezentuje algorytm wykonujący działanie mnożenia każdego elementu tablicy `tab` przez 2.

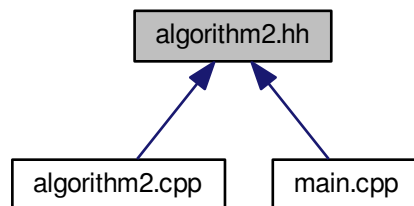
6.3 algorithm2.cpp File Reference

```
#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:



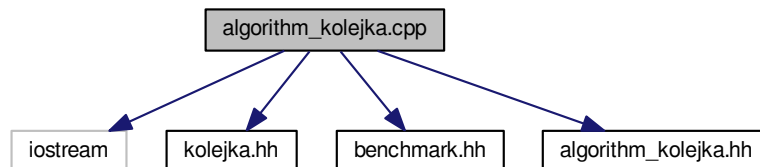
Classes

- class [Algorithm2](#)

Klasa [Algorithm2](#) modelująca algorytm wczytywania do listy tablicowej. Obiekt tego typu reprezentuje algorytm wykonujący wczytywanie zadanej ilości elementów do listy tablicowej.

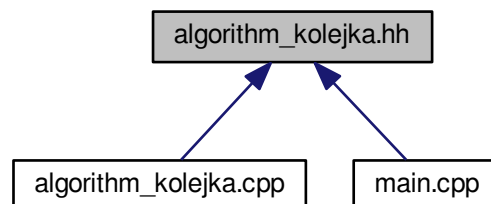
6.5 algorithm_kolejka.cpp File Reference

```
#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

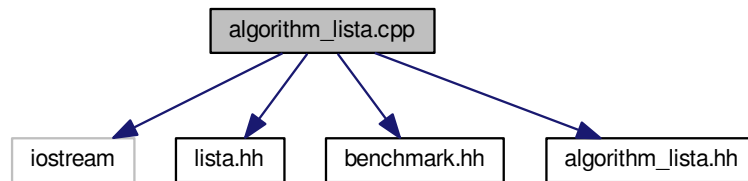
- class [AlgorithmKolejka](#)

Klasa [AlgorithmKolejka](#) modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wykonujący wczytywanie zadanej ilości elementów do kolejki.

6.7 algorithm_lista.cpp File Reference

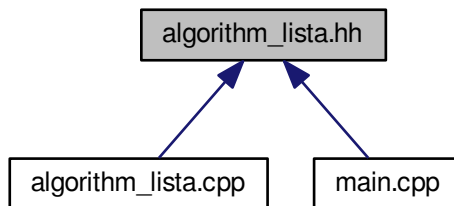
```
#include <iostream>
#include "lista.hh"
#include "benchmark.hh"
#include "algorithm_lista.hh"
```

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

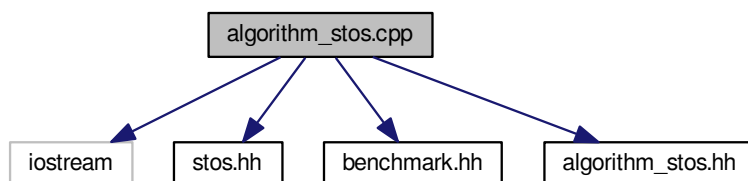
- class [AlgorithmLista](#)

Klasa [AlgorithmLista](#) modelująca algorytm wczytywania do kolejki. Obiekt tego typu reprezentuje algorytm wykonujący wykonujący wczytywanie zadanej ilości elementów do listy.

6.9 `algorithm_stos.cpp` File Reference

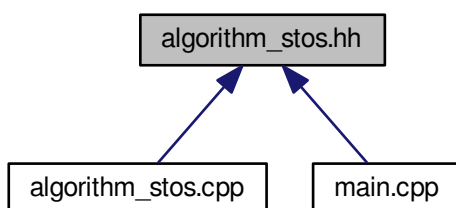
```
#include <iostream>
#include "stos.hh"
#include "benchmark.hh"
#include "algorithm_stos.hh"
```


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

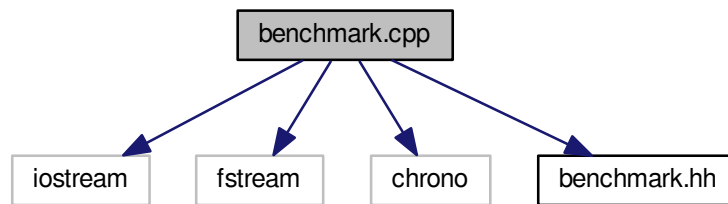
- class [AlgorithmStos](#)

Klasa [AlgorithmStos](#) modelująca algorytm wczytywania do stosu. Obiekt tego typu reprezentuje algorytm wykonujący wykonujący wczytywanie zadanej ilości elementów 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

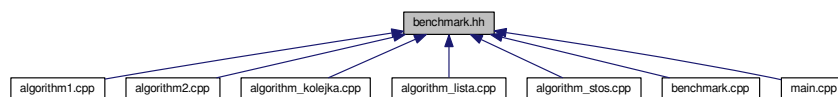
Definition at line 7 of file benchmark.cpp.

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](#) modelująca program benchmarkujący. Obiekt tego typu reprezentuje program sprawdzający szybkość wykonywania algorytmów.

Macros

- #define [SIZE](#) 10000

6.12.1 Macro Definition Documentation

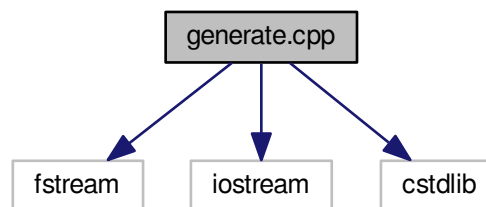
6.12.1.1 #define SIZE 10000

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 ()

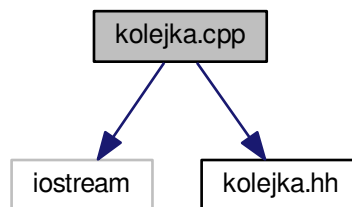
Return values

0	- gdy funkcja zadziała poprawnie.
1	- gdy wystąpi błąd otwarcia pliku do zapisu.

Definition at line 14 of file generate.cpp.

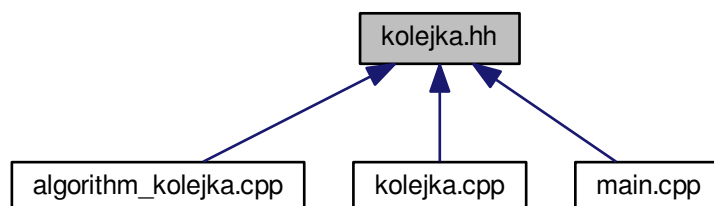
6.14 kolejka.cpp File Reference

```
#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:



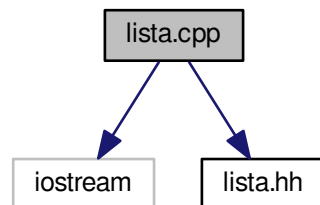
Classes

- class [Kolejka](#)

Klasa [Kolejka](#) modelująca strukturę danych typu kolejka. Obiekt tego typu reprezentuje strukturę danych typu kolejka wraz z operacjami możliwymi do wykonania na tej strukturze.

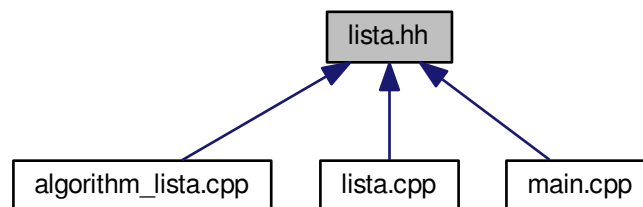
6.16 lista.cpp File Reference

```
#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](#) modelująca strukturę danych typu lista. Obiekt tego typu reprezentuje strukturę danych typu lista wraz z operacjami możliwymi do wykonania na tej strukturze.

- struct [Lista::Komorka](#)

Struktura [Komorka](#). Obiekt tego typu reprezentuje pojedynczą komórkę wraz ze wskaźnikiem na następną komórkę 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 tworząca i testująca algorytm. Wczytuje dane otrzymane na strumień wejściowy do tablicy `data[]`. Następnie tworzy obiekt [Benchmark](#) oraz obiekt `Potegowanie`. Później uruchamia metodę testującą w obiekcie klasy [Benchmark](#) dla obiektu klasy `Potegowanie`.

6.18.1 Function Documentation

6.18.1.1 `int main ()`

Return values

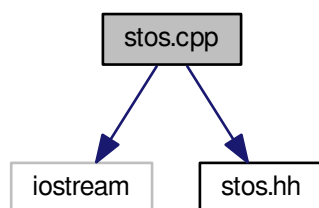
<code>0</code>	- domyślna wartość zwracana przez funkcję.
----------------	--

Definition at line 21 of file main.cpp.

6.19 stos.cpp File Reference

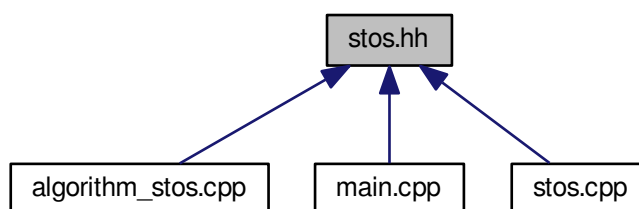
```
#include <iostream>
#include "stos.hh"
```

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](#)

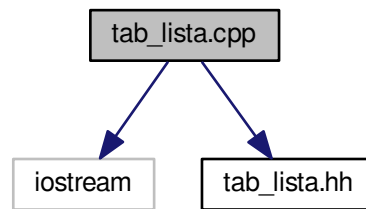
Klasa [Stos](#) modelująca strukturę danych typu stos. Obiekt tego typu reprezentuje strukturę danych typu stos wraz z operacjami możliwymi do wykonania na tej strukturze.

6.21 strona-glowna.dox File Reference

6.22 tab_lista.cpp File Reference

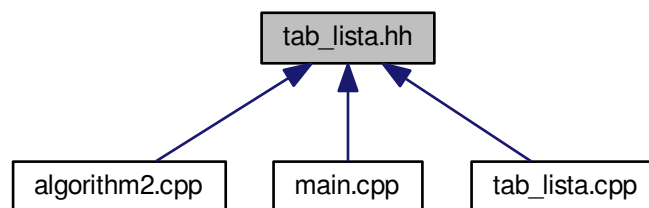
```
#include <iostream>
#include "tab_lista.hh"
```

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](#) modelująca strukturę danych typu lista. Obiekt tego typu reprezentuje strukturę danych typu lista zaimplementowana na tablicy dynamicznej. Obiekt zawiera również podstawowe metody listy.

Index

- ~Algorithm2
 - Algorithm2, [10](#)
- ~AlgorithmKolejka
 - AlgorithmKolejka, [13](#)
- ~AlgorithmLista
 - AlgorithmLista, [16](#)
- ~AlgorithmStos
 - AlgorithmStos, [18](#)
- ~Benchmark
 - Benchmark, [20](#)
- ~Kolejka
 - Kolejka, [22](#)
- ~Lista
 - Lista, [26](#)
- ~Mnozenie
 - Mnozenie, [29](#)
- ~Stos
 - Stos, [31](#)
- ~TabLista
 - TabLista, [34](#)
- algorithm1.cpp, [37](#)
- algorithm1.hh, [37](#)
- Algorithm2, [9](#)
 - ~Algorithm2, [10](#)
 - Algorithm2, [10](#)
 - runAlgorithm, [10](#)
 - runAlgorithmSzybkieSortowanie, [11](#)
 - tab, [11](#)
 - tablista, [11](#)
 - testAlgorithm, [11](#)
- algorithm2.cpp, [38](#)
- algorithm2.hh, [38](#)
- algorithm_kolejka.cpp, [39](#)
- algorithm_kolejka.hh, [39](#)
- algorithm_lista.cpp, [39](#)
- algorithm_lista.hh, [40](#)
- algorithm_stos.cpp, [40](#)
- algorithm_stos.hh, [41](#)
- AlgorithmKolejka, [12](#)
 - ~AlgorithmKolejka, [13](#)
 - AlgorithmKolejka, [13](#)
 - kolejka, [14](#)
 - runAlgorithm, [13](#)
 - tab, [14](#)
 - testAlgorithm, [14](#)
- AlgorithmLista, [14](#)
 - ~AlgorithmLista, [16](#)
 - AlgorithmLista, [15](#)
 - lista, [16](#)
- runAlgorithm, [16](#)
- tab, [16](#)
- testAlgorithm, [16](#)
- AlgorithmStos, [17](#)
 - ~AlgorithmStos, [18](#)
 - AlgorithmStos, [18](#)
 - runAlgorithm, [18](#)
 - stos, [19](#)
 - tab, [19](#)
 - testAlgorithm, [18](#)
- Benchmark, [19](#)
 - ~Benchmark, [20](#)
 - Benchmark, [20](#)
 - nazwy, [21](#)
 - runAlgorithm, [20](#)
 - runAlgorithmSzybkieSortowanie, [20](#)
 - testAlgorithm, [21](#)
- benchmark.cpp, [41](#)
 - LENGTH, [42](#)
 - REPEATS, [42](#)
- benchmark.hh, [42](#)
 - SIZE, [43](#)
- decrease
 - Stos, [31](#)
- dequeue
 - Kolejka, [23](#)
- elem
 - Lista::Komorka, [25](#)
- enqueue
 - Kolejka, [23](#)
- f
 - Kolejka, [24](#)
- generate.cpp, [43](#)
 - main, [43](#)
 - SIZE, [43](#)
- head
 - Lista, [27](#)
- increase
 - Kolejka, [23](#)
 - Stos, [31](#)
 - TabLista, [34](#)
- insert
 - Lista, [26](#)
 - TabLista, [34](#)

- Kolejka, [21](#)
 - ~Kolejka, [22](#)
 - dequeue, [23](#)
 - enqueue, [23](#)
 - f, [24](#)
 - increase, [23](#)
 - Kolejka, [22](#)
 - r, [24](#)
 - size, [24](#)
 - tab, [24](#)
- kolejka
 - AlgorithmKolejka, [14](#)
- kolejka.cpp, [44](#)
- kolejka.hh, [44](#)
- Komorka
 - Lista::Komorka, [25](#)
- LENGTH
 - benchmark.cpp, [42](#)
- last
 - Stos, [32](#)
 - TabLista, [36](#)
- Lista, [25](#)
 - ~Lista, [26](#)
 - head, [27](#)
 - insert, [26](#)
 - Lista, [26](#)
 - remove, [27](#)
 - tail, [27](#)
- lista
 - AlgorithmLista, [16](#)
- lista.cpp, [45](#)
- lista.hh, [45](#)
- Lista::Komorka, [24](#)
 - elem, [25](#)
 - Komorka, [25](#)
 - next, [25](#)
- main
 - generate.cpp, [43](#)
 - main.cpp, [46](#)
- main.cpp, [46](#)
 - main, [46](#)
- Mnozenie, [27](#)
 - ~Mnozenie, [29](#)
 - Mnozenie, [29](#)
 - runAlgorithm, [29](#)
 - tab, [29](#)
 - testAlgorithm, [29](#)
- nazwy
 - Benchmark, [21](#)
- next
 - Lista::Komorka, [25](#)
- pop
 - Stos, [31](#)
- push
 - Stos, [32](#)
- quicksort
 - TabLista, [35](#)
- r
 - Kolejka, [24](#)
- REPEATS
 - benchmark.cpp, [42](#)
- remove
 - Lista, [27](#)
 - TabLista, [35](#)
- rozmiar
 - TabLista, [35](#)
- runAlgorithm
 - Algorithm2, [10](#)
 - AlgorithmKolejka, [13](#)
 - AlgorithmLista, [16](#)
 - AlgorithmStos, [18](#)
 - Benchmark, [20](#)
 - Mnozenie, [29](#)
- runAlgorithmSzybkieSortowanie
 - Algorithm2, [11](#)
 - Benchmark, [20](#)
- SIZE
 - benchmark.hh, [43](#)
 - generate.cpp, [43](#)
- size
 - Kolejka, [24](#)
 - Stos, [32](#)
 - TabLista, [36](#)
- Stos, [30](#)
 - ~Stos, [31](#)
 - decrease, [31](#)
 - increase, [31](#)
 - last, [32](#)
 - pop, [31](#)
 - push, [32](#)
 - size, [32](#)
 - Stos, [30](#)
 - tab, [33](#)
- stos
 - AlgorithmStos, [19](#)
- stos.cpp, [46](#)
- stos.hh, [47](#)
- strona-glowna.dox, [47](#)
- tab
 - Algorithm2, [11](#)
 - AlgorithmKolejka, [14](#)
 - AlgorithmLista, [16](#)
 - AlgorithmStos, [19](#)
 - Kolejka, [24](#)
 - Mnozenie, [29](#)
 - Stos, [33](#)
 - TabLista, [36](#)
- tab_lista.cpp, [47](#)
- tab_lista.hh, [48](#)
- TabLista, [33](#)
 - ~TabLista, [34](#)

- increase, [34](#)
- insert, [34](#)
- last, [36](#)
- quicksort, [35](#)
- remove, [35](#)
- rozmiar, [35](#)
- size, [36](#)
- tab, [36](#)
- TabLista, [34](#)
- tablista
 - Algorithm2, [11](#)
- tail
 - Lista, [27](#)
- testAlgorithm
 - Algorithm2, [11](#)
 - AlgorithmKolejka, [14](#)
 - AlgorithmLista, [16](#)
 - AlgorithmStos, [18](#)
 - Benchmark, [21](#)
 - Mnozenie, [29](#)