

lab1  
0.0001

Wygenerowano przez Doxygen 1.8.6

Śr, 11 mar 2015 22:44:32



# Spis treści

<b>1</b>	<b>lab1</b>	<b>1</b>
<b>2</b>	<b>Indeks hierarchiczny</b>	<b>3</b>
2.1	Hierarchia klas . . . . .	3
<b>3</b>	<b>Indeks klas</b>	<b>5</b>
3.1	Lista klas . . . . .	5
<b>4</b>	<b>Indeks plików</b>	<b>7</b>
4.1	Lista plików . . . . .	7
<b>5</b>	<b>Dokumentacja klas</b>	<b>9</b>
5.1	Dokumentacja klasy Benchmark . . . . .	9
5.1.1	Opis szczegółowy . . . . .	10
5.1.2	Dokumentacja konstruktora i destruktora . . . . .	10
5.1.2.1	Benchmark . . . . .	10
5.1.2.2	Benchmark . . . . .	10
5.1.2.3	~Benchmark . . . . .	10
5.1.3	Dokumentacja funkcji składowych . . . . .	10
5.1.3.1	runAlgorithm . . . . .	10
5.1.3.2	testAlgorithm . . . . .	10
5.1.4	Dokumentacja atrybutów składowych . . . . .	11
5.1.4.1	repeats . . . . .	11
5.2	Dokumentacja klasy Mnozenie . . . . .	11
5.2.1	Opis szczegółowy . . . . .	12
5.2.2	Dokumentacja konstruktora i destruktora . . . . .	12
5.2.2.1	Mnozenie . . . . .	12
5.2.2.2	Mnozenie . . . . .	12
5.2.2.3	~Mnozenie . . . . .	13
5.2.3	Dokumentacja funkcji składowych . . . . .	13
5.2.3.1	runAlgorithm . . . . .	13
5.2.3.2	testAlgorithm . . . . .	13
5.2.4	Dokumentacja atrybutów składowych . . . . .	13

5.2.4.1	tab	13
<b>6</b>	<b>Dokumentacja plików</b>	<b>15</b>
6.1	Dokumentacja pliku algorithm1.cpp	15
6.2	Dokumentacja pliku algorithm1.hh	15
6.3	Dokumentacja pliku benchmark.cpp	16
6.3.1	Dokumentacja definicji	16
6.3.1.1	LENGTH	16
6.4	Dokumentacja pliku benchmark.hh	17
6.4.1	Dokumentacja definicji	17
6.4.1.1	SIZE	17
6.5	Dokumentacja pliku generate.cpp	17
6.5.1	Dokumentacja definicji	18
6.5.1.1	SIZE	18
6.5.2	Dokumentacja funkcji	18
6.5.2.1	main	18
6.6	Dokumentacja pliku main.cpp	18
6.6.1	Dokumentacja funkcji	19
6.6.1.1	main	19
6.7	Dokumentacja pliku strona-glowna.dox	19
<b>Indeks</b>		<b>20</b>

# Rozdział 1

## lab1

Autor

Filip Malinowski

Program sluzacy do uruchamiania algorytmow i badania ich szybkosci dzialania.

Format wywolania:

`./prj/make clean`

`./prj/make`



## Rozdział 2

# Indeks hierarchiczny

### 2.1 Hierarchia klas

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

Benchmark . . . . .	9
Mnozenie . . . . .	11





## Rozdział 3

# Indeks klas

### 3.1 Lista klas

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

#### Benchmark

Klasa [Benchmark](#) modelująca program benchmarkujący. Obiekt tego typu reprezentuje program sprawdzający szybkość wykonywania algorytmów . . . . . 9

#### Mnozenie

Klasa [Mnozenie](#) modelująca algorytm potęgowania. Obiekt tego typu reprezentuje algorytm wykonujący działanie mnożenia każdego elementu tablicy tab przez 2 . . . . . 11



## Rozdział 4

# Indeks plików

### 4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">algorithm1.cpp</a>	15
<a href="#">algorithm1.hh</a>	15
<a href="#">benchmark.cpp</a>	16
<a href="#">benchmark.hh</a>	17
<a href="#">generate.cpp</a>	17
<a href="#">main.cpp</a>	18



## Rozdział 5

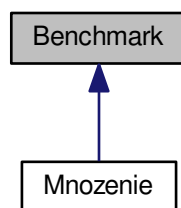
# Dokumentacja klas

### 5.1 Dokumentacja klasy Benchmark

Klasa `Benchmark` modelująca program benchmarkujący. Obiekt tego typu reprezentuje program sprawdzający szybkość wykonywania algorytmów.

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

Diagram dziedziczenia dla `Benchmark`



#### Metody publiczne

- `Benchmark ()`  
*Konstruktor obiektu `Benchmark`.*
- `Benchmark (int _repeats)`  
*Konstruktor parametryczny obiektu `Benchmark`.*
- `~Benchmark ()`  
*Destruktor obiektu `Benchmark`.*
- `virtual void testAlgorithm (Benchmark *_algorithm) 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.*

## Atrybuty prywatne

- `int repeats`

### 5.1.1 Opis szczegółowy

Definicja w linii 11 pliku `benchmark.hh`.

### 5.1.2 Dokumentacja konstruktora i destruktor

#### 5.1.2.1 `Benchmark::Benchmark ( )` `[inline]`

Definicja w linii 20 pliku `benchmark.hh`.

#### 5.1.2.2 `Benchmark::Benchmark ( int _repeats )` `[inline]`

##### Parametry

<code>in</code>	<code>_repeats</code>	- ilość powtórzeń testu dla pojedynczego zestawu danych.
-----------------	-----------------------	--

Definicja w linii 26 pliku `benchmark.hh`.

#### 5.1.2.3 `Benchmark::~~Benchmark ( )` `[inline]`

Definicja w linii 31 pliku `benchmark.hh`.

### 5.1.3 Dokumentacja funkcji składowych

#### 5.1.3.1 `virtual void Benchmark::runAlgorithm ( int _border )` `[inline]`, `[virtual]`

##### Parametry

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

Reimplementowana w [Mnozenie](#).

Definicja w linii 50 pliku `benchmark.hh`.

Oto graf wywołań tej funkcji:



#### 5.1.3.2 `void Benchmark::testAlgorithm ( Benchmark * _algorithm ) const` `[virtual]`

## Parametry

<code>in</code>	<code>_algorithm</code>	- testowany algorytm.
-----------------	-------------------------	-----------------------

Definicja w linii 10 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:



### 5.1.4 Dokumentacja atrybutów składowych

#### 5.1.4.1 `int Benchmark::repeats` `[private]`

Definicja w linii 13 pliku benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

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

## 5.2 Dokumentacja klasy Mnozenie

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

```
#include <algorithm1.hh>
```

Diagram dziedziczenia dla Mnozenie

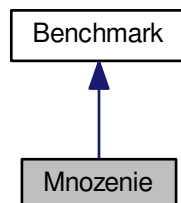
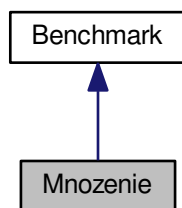


Diagram współpracy dla Mnozenie:



## Metody publiczne

- [Mnozenie](#) ()  
*Konstruktor obiektu [Mnozenie](#).*
- [Mnozenie](#) (int \_tab[SIZE])  
*Konstruktor parametryczny obiektu [Mnozenie](#).*
- [~Mnozenie](#) ()  
*Destruktor obiektu [Mnozenie](#).*
- virtual void [testAlgorithm](#) (Benchmark \*\_algorithm)  
*Metoda testowania algorytmu. Metoda sluzi do testowania szybkości działania algorytmu. W klasie [Mnozenie](#) nie ma konkretnego działania.*
- virtual void [runAlgorithm](#) (int \_border)  
*Metoda uruchamiania algorytmu. Metoda sluzi to wykonywania danego algorytmu. Mnozy kazdy element tablicy przez liczbę 2.*

## Atrybuty prywatne

- int [tab](#) [SIZE]

### 5.2.1 Opis szczegółowy

Definicja w linii 9 pliku algorithm1.hh.

### 5.2.2 Dokumentacja konstruktora i destruktora

#### 5.2.2.1 `Mnozenie::Mnozenie ( ) [inline]`

Definicja w linii 17 pliku algorithm1.hh.

#### 5.2.2.2 `Mnozenie::Mnozenie ( int _tab[SIZE] ) [inline]`



## Parametry

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

Definicja w linii 23 pliku `algorithm1.hh`.

5.2.2.3 `Mnozenie::~~Mnozenie ( ) [inline]`

Definicja w linii 28 pliku `algorithm1.hh`.

## 5.2.3 Dokumentacja funkcji składowych

5.2.3.1 `void Mnozenie::runAlgorithm ( int _border ) [virtual]`

## Parametry

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

Reimplementowana z [Benchmark](#).

Definicja w linii 7 pliku `algorithm1.cpp`.

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

## Parametry

<code>in</code>	<code>_algorithm</code>	- testowany algorytm.
-----------------	-------------------------	-----------------------

Definicja w linii 36 pliku `algorithm1.hh`.

## 5.2.4 Dokumentacja atrybutów składowych

5.2.4.1 `int Mnozenie::tab[SIZE] [private]`

Definicja w linii 11 pliku `algorithm1.hh`.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [algorithm1.hh](#)
- [algorithm1.cpp](#)



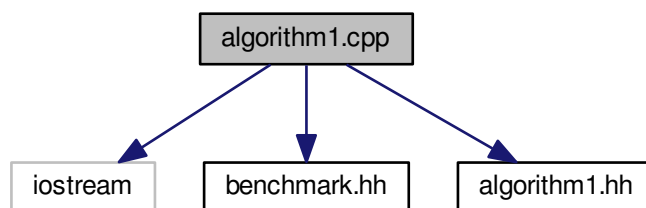
## Rozdział 6

# Dokumentacja plików

### 6.1 Dokumentacja pliku algorithm1.cpp

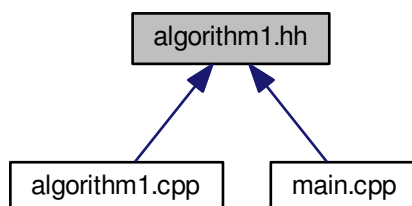
```
#include <iostream>
#include "benchmark.hh"
#include "algorithm1.hh"
```

Wykres zależności załączania dla algorithm1.cpp:



### 6.2 Dokumentacja pliku algorithm1.hh

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



## Komponenty

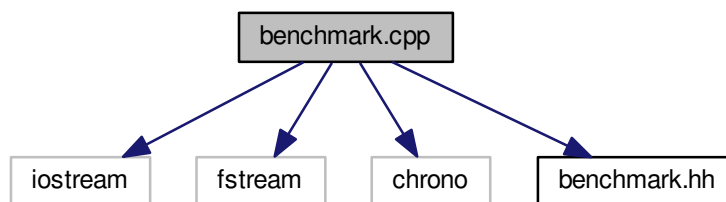
- class [Mnozenie](#)

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

## 6.3 Dokumentacja pliku benchmark.cpp

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

Wykres zależności załączania dla benchmark.cpp:



## Definicje

- #define [LENGTH](#) 5

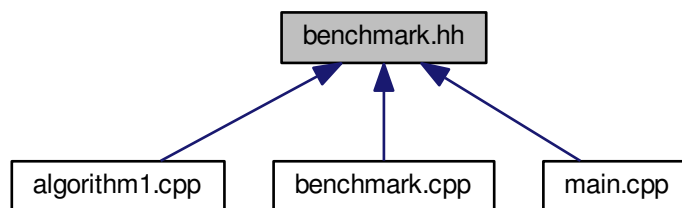
### 6.3.1 Dokumentacja definicji

#### 6.3.1.1 #define LENGTH 5

Definicja w linii 7 pliku benchmark.cpp.

## 6.4 Dokumentacja pliku benchmark.hh

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



### Komponenty

- class [Benchmark](#)

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

### Definicje

- `#define SIZE 10000000`

#### 6.4.1 Dokumentacja definicji

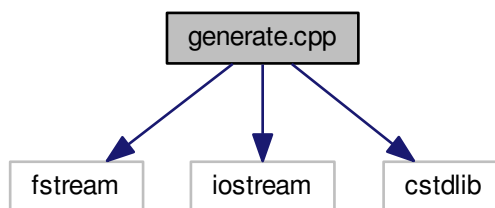
##### 6.4.1.1 `#define SIZE 10000000`

Definicja w linii 4 pliku benchmark.hh.

## 6.5 Dokumentacja pliku generate.cpp

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

Wykres zależności załączania dla generate.cpp:



## Definicje

- `#define SIZE 10000000`

## Funkcje

- `int main ()`

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

### 6.5.1 Dokumentacja definicji

#### 6.5.1.1 `#define SIZE 10000000`

Definicja w linii 5 pliku generate.cpp.

### 6.5.2 Dokumentacja funkcji

#### 6.5.2.1 `int main ( )`

Zwracane wartości

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

Definicja w linii 14 pliku generate.cpp.

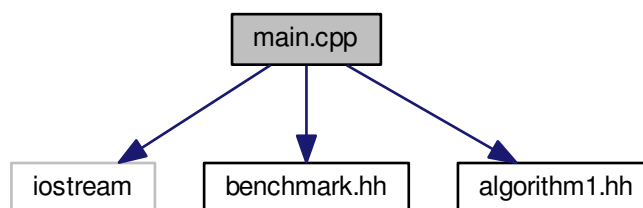
## 6.6 Dokumentacja pliku main.cpp

```

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

```

Wykres zależności załączania dla main.cpp:



## Funkcje

- 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.*

### 6.6.1 Dokumentacja funkcji

#### 6.6.1.1 int main ( )

Zwracane wartości

0	- domyślna wartość zwracana przez funkcję.
---	--

Definicja w linii 14 pliku main.cpp.

## 6.7 Dokumentacja pliku strona-glowna.dox

# Skorowidz

- ~Benchmark
  - Benchmark, [10](#)
- ~Mnozenie
  - Mnozenie, [13](#)
- algorithm1.cpp, [15](#)
- algorithm1.hh, [15](#)
- Benchmark, [9](#)
  - ~Benchmark, [10](#)
  - Benchmark, [10](#)
  - repeats, [11](#)
  - runAlgorithm, [10](#)
  - testAlgorithm, [10](#)
- benchmark.cpp, [16](#)
  - LENGTH, [16](#)
- benchmark.hh, [17](#)
  - SIZE, [17](#)
- generate.cpp, [17](#)
  - main, [18](#)
  - SIZE, [18](#)
- LENGTH
  - benchmark.cpp, [16](#)
- main
  - generate.cpp, [18](#)
  - main.cpp, [19](#)
- main.cpp, [18](#)
  - main, [19](#)
- Mnozenie, [11](#)
  - ~Mnozenie, [13](#)
  - Mnozenie, [12](#)
  - runAlgorithm, [13](#)
  - tab, [13](#)
  - testAlgorithm, [13](#)
- repeats
  - Benchmark, [11](#)
- runAlgorithm
  - Benchmark, [10](#)
  - Mnozenie, [13](#)
- SIZE
  - benchmark.hh, [17](#)
  - generate.cpp, [18](#)
- strona-glowna.dox, [19](#)
- tab
  - Mnozenie, [13](#)
- testAlgorithm
  - Benchmark, [10](#)
  - Mnozenie, [13](#)