

Podstawowe struktury danych

1

Wygenerowano przez Doxygen 1.8.6

N, 15 mar 2015 00:09:34

Spis treści

1	Sprawozdanie	1
1.1	Zadanie	1
1.2	Wyniki	1
1.3	Podsumowanie	2
2	Indeks hierarchiczny	3
2.1	Hierarchia klas	3
3	Indeks klas	5
3.1	Lista klas	5
4	Indeks plików	7
4.1	Lista plików	7
5	Dokumentacja klas	9
5.1	Dokumentacja klasy benchmark	9
5.1.1	Opis szczegółowy	9
5.1.2	Dokumentacja funkcji składowych	10
5.1.2.1	analize	10
5.1.2.2	test	11
5.2	Dokumentacja klasy tabx2	12
5.2.1	Opis szczegółowy	13
5.2.2	Dokumentacja konstruktora i destruktora	13
5.2.2.1	tabx2	13
5.2.2.2	~tabx2	13
5.2.3	Dokumentacja funkcji składowych	13
5.2.3.1	test	13
5.2.4	Dokumentacja atrybutów składowych	13
5.2.4.1	size	13
5.2.4.2	tab	13
6	Dokumentacja plików	15
6.1	Dokumentacja pliku benchmark.cpp	15

6.2	Dokumentacja pliku benchmark.hh	15
6.3	Dokumentacja pliku generator.cpp	16
6.3.1	Dokumentacja funkcji	17
6.3.1.1	data_generator	17
6.4	Dokumentacja pliku generator.hh	17
6.4.1	Dokumentacja funkcji	18
6.4.1.1	data_generator	18
6.5	Dokumentacja pliku main.cpp	18
6.5.1	Dokumentacja funkcji	19
6.5.1.1	main	19
6.6	Dokumentacja pliku strona.dox	19
6.7	Dokumentacja pliku tabx2.cpp	19
6.8	Dokumentacja pliku tabx2.hh	20
6.8.1	Opis szczegółowy	21
6.8.2	Dokumentacja definicji	21
6.8.2.1	TABX2_HH	21
Indeks		22

Rozdział 1

Sprawozdanie

Data

11.03.2015r.

Wersja

0.1

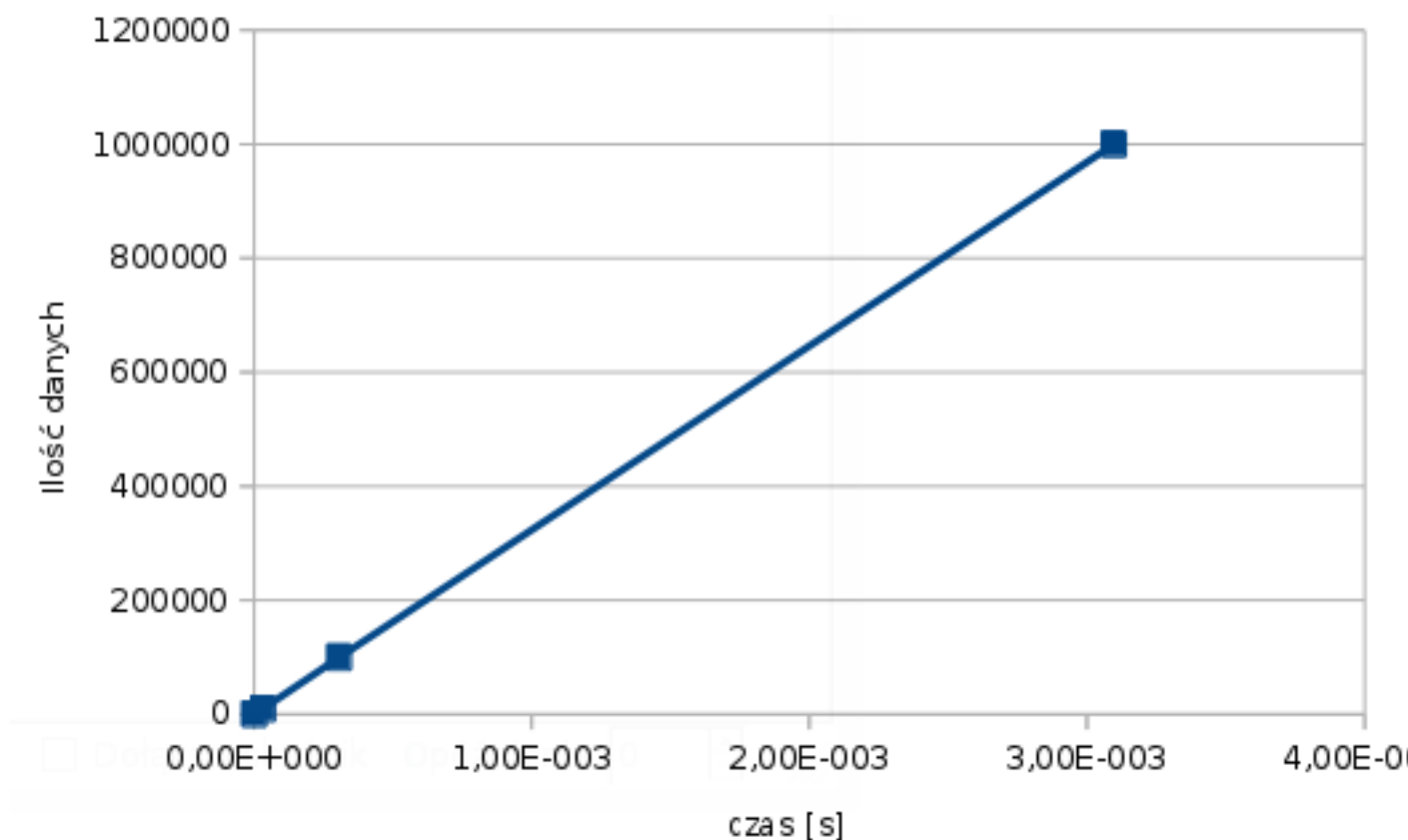
1.1 Zadanie

Celem ćwiczenia było stworzenie programu benchmarkującego , który dla wybranych danych będzie zliczał średni czas wykonania dowolnego algorytmu (w tym przypadku mnożenia elementów tablicy przez 2). Należało również stworzyć program generujący losowe liczby.

1.2 Wyniki

Dla dziesięciu milionów liczb program zwraca 7 danych wyjściowych (zgodnie z algorytmem 10^n , gdzie n jest równocześnie ilością zwracanych czasów oraz maksymalną liczbą danych dla jakiej przeprowadzany był test

Na podstawie otrzymanych danych mamy :



1.3 Podsumowanie

Wykres dodany do dokumentacji z niewiadomych względów nie jest wyświetlany poprawnie (dodano sprawozdanie również w formacie pdf). Zgodnie z przewidywaniami złożoność obliczeniowa jest liniowa, jedyną rzeczą, która zwraca uwagę jest fakt iż czas wykonania jednej operacji jest dłuższy od czasu wykonania 10 operacji. Dla większej ilości danych wyniki są poprawne. Wydaje się, że zbyt mało danych jest obecnych w środkowej części wykresu, co powinno być zostać zmienione w celu poprawy jakości odbioru wykresu (dla charakterystyki liniowej jest to akurat bez znaczenia, ale np. dla logarytmicznej byłoby to widoczne).

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

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

benchmark	9
tabx2	12

Rozdział 3

Indeks klas

3.1 Lista klas

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

benchmark	9
tabx2	12

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

benchmark.cpp	Deklaracja funkcji z klasy Benchmark	15
benchmark.hh	Definicja klasy Benchmark	15
generator.cpp	Deklaracja funkcji generującej liczby losowe	16
generator.hh	Definicja generatora liczb losowych	17
main.cpp	18
tabx2.cpp	Deklaracja klasy main	19
tabx2.hh	Definicja klasy tabx2	20

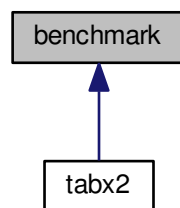
Rozdział 5

Dokumentacja klas

5.1 Dokumentacja klasy benchmark

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

Diagram dziedziczenia dla benchmark



Metody publiczne

- void [analyze](#) (int repeat, int data_amount)

Metoda pozwala przeprowadzić analizę czasu trwania funkcji test w następujący sposób : wykonuje funkcję test repeat razy , po czym uśrednia jej wynik. Cały proces jest wykonywany dla data_amount razy , gdzie data amount jest potęgą liczby 10.

Metody prywatne

- virtual void [test](#) (int length)=0

Klasa benchmark umożliwia przeprowadzenie pomiaru czasu trwania programu dla określonych danych.

5.1.1 Opis szczegółowy

Definicja w linii 12 pliku benchmark.hh.

5.1.2 Dokumentacja funkcji składowych

5.1.2.1 `void benchmark::analyze (int repeat, int data_amount)`

Parametry

in	<i>repeat</i>	- ilość powtórzeń testu który chcemy wykonać
in	<i>data_amount</i>	- ilość wynikowych danych podawana jako potęga liczby 10

Zwraca

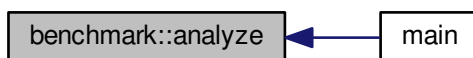
plik .csv z czasami poszczególnych pomiarów oraz ilość testowanych danych

Definicja w linii 7 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



5.1.2.2 virtual void benchmark::test (int length) [private],[pure virtual]

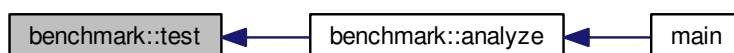
Metoda wirtualna, reprezentuje czynność której czas chcemy zmierzyć

Parametry

in	<i>length</i>	- ilość danych dla jakich przeprowadzamy test
----	---------------	---

Implementowany w [tabx2](#).

Oto graf wywoływań tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

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

5.2 Dokumentacja klasy tabx2

```
#include <tabx2.hh>
```

Diagram dziedziczenia dla tabx2

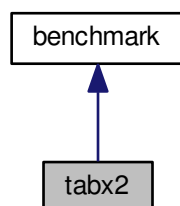
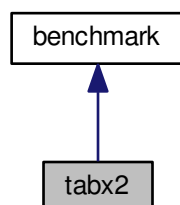


Diagram współpracy dla tabx2:



Metody publiczne

- void [test](#) (int length)
- [tabx2](#) (int [size](#))
- [~tabx2](#) ()

zwykły destruktor

Atrybuty prywatne

- int [size](#)
Modeluje pojęcie [tabx2](#).
- int * [tab](#)

Wskaźnik na tablice przechowująca dane.

5.2.1 Opis szczegółowy

Definicja w linii 15 pliku tabx2.hh.

5.2.2 Dokumentacja konstruktora i destruktora

5.2.2.1 tabx2::tabx2 (int *size*)

Konstruktor wczytujący określoną ilość danych i alokujący je dynamicznie

Parametry

<i>in</i>	<i>size</i>	- długość tablice
-----------	-------------	-------------------

Definicja w linii 15 pliku tabx2.cpp.

5.2.2.2 tabx2::~~tabx2 ()

Definicja w linii 35 pliku tabx2.cpp.

5.2.3 Dokumentacja funkcji składowych

5.2.3.1 void tabx2::test (int *length*) [virtual]

Mnoży określoną ilość danych przez 2

Parametry

<i>in</i>	<i>length</i>	- ilość danych do przemnożenia
-----------	---------------	--------------------------------

Implementuje [benchmark](#).

Definicja w linii 7 pliku tabx2.cpp.

5.2.4 Dokumentacja atrybutów składowych

5.2.4.1 int tabx2::size [private]

Tworzy tablice alokowaną dynamicznie o pojemności wybranej przez użytkownika, umożliwia wykonywanie mnożenia przez 2 wszystkich elementów tablicy

Rozmiar tablicy

Definicja w linii 27 pliku tabx2.hh.

5.2.4.2 int* tabx2::tab [private]

Definicja w linii 30 pliku tabx2.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [tabx2.hh](#)
- [tabx2.cpp](#)

Rozdział 6

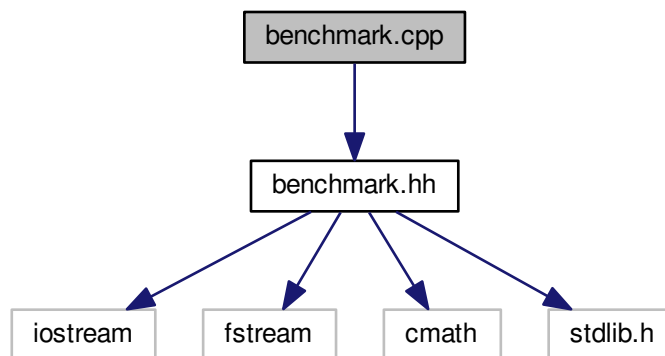
Dokumentacja plików

6.1 Dokumentacja pliku benchmark.cpp

Deklaracja funkcji z klasy Benchmark.

```
#include "benchmark.hh"
```

Wykres zależności załączania dla benchmark.cpp:

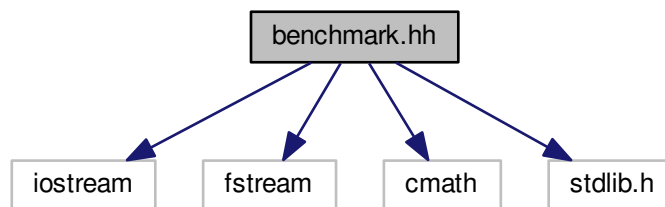


6.2 Dokumentacja pliku benchmark.hh

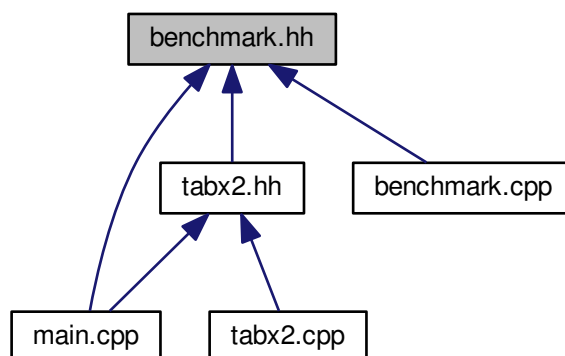
Definicja klasy Benchmark.

```
#include <iostream>
#include <fstream>
#include <cmath>
#include "stdlib.h"
```

Wykres zależności załączania dla benchmark.hh:

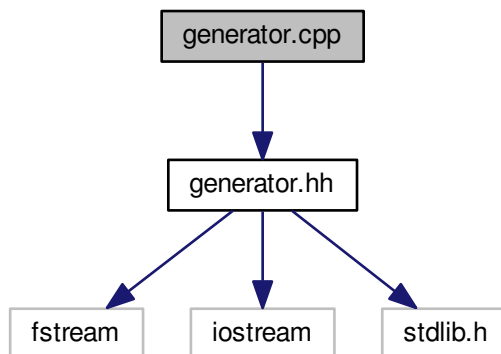


Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



```
#include "generator.hh"
```

Wykres zależności załączania dla generator.cpp:



Funkcje

- bool `data_generator` (int `data_amount`)

Generuje liczby losowe.

6.3.1 Dokumentacja funkcji

6.3.1.1 bool `data_generator` (int `data_amount`)

Funkcja generuje naturalne liczby losowe z przedziału 0-100, które następnie są zapisywane do pliku `random_data.dat`

Parametry

in	<code>data_amount</code>	- ilość liczb wynikowych które chcemy uzyskać
----	--------------------------	---

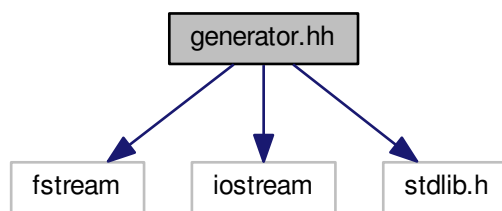
Definicja w linii 9 pliku `generator.cpp`.

6.4 Dokumentacja pliku generator.hh

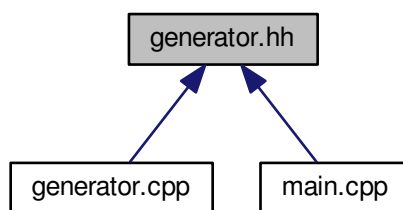
Definicja generatora liczb losowych.

```
#include <fstream>
#include <iostream>
#include <stdlib.h>
```

Wykres zależności załączania dla generator.hh:



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



Funkcje

- bool `data_generator` (int `data_amount`)
Generuje liczby losowe.

6.4.1 Dokumentacja funkcji

6.4.1.1 bool `data_generator` (int `data_amount`)

Funkcja generuje naturalne liczby losowe z przedziału 0-100, które następnie są zapisywane do pliku `random_data.dat`

Parametry

in	<code>data_amount</code>	- ilość liczb wynikowych które chcemy uzyskać
----	--------------------------	---

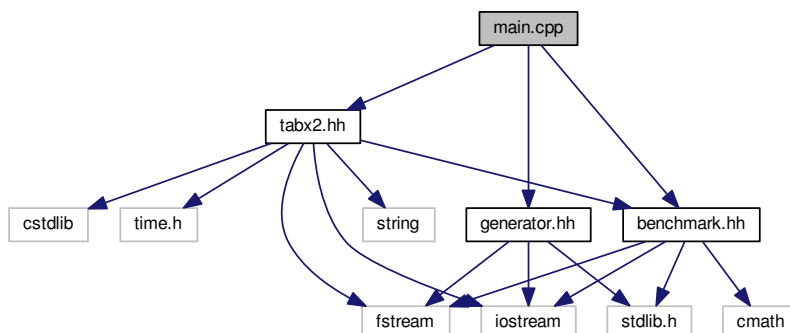
Definicja w linii 9 pliku `generator.cpp`.

6.5 Dokumentacja pliku `main.cpp`

```
#include "tabx2.hh"
```

```
#include "benchmark.hh"
#include "generator.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

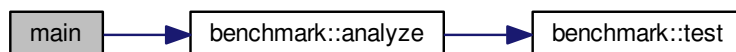
- int `main` ()

6.5.1 Dokumentacja funkcji

6.5.1.1 int main ()

Definicja w linii 6 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



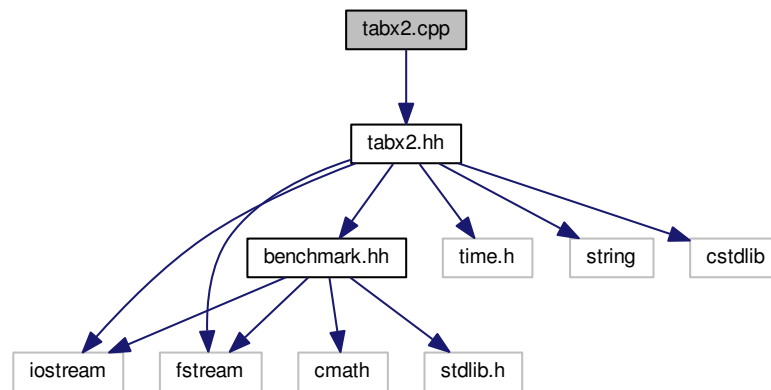
6.6 Dokumentacja pliku strona.dox

6.7 Dokumentacja pliku tabx2.cpp

Deklaracja klasy main.

```
#include "tabx2.hh"
```

Wykres zależności załączania dla tabx2.cpp:

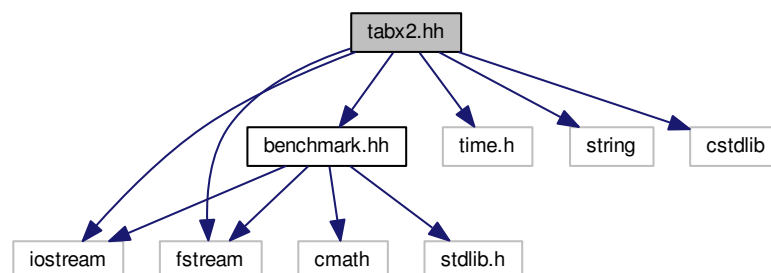


6.8 Dokumentacja pliku tabx2.hh

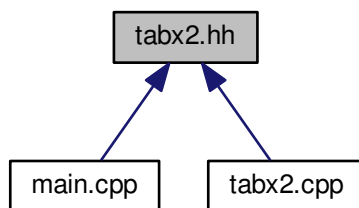
Definicja klasy [tabx2](#).

```
#include <iostream>
#include <time.h>
#include <string>
#include <fstream>
#include <cstdlib>
#include "benchmark.hh"
```

Wykres zależności załączania dla tabx2.hh:



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



Komponenty

- class `tabx2`

Definicje

- `#define TABX2_HH`

6.8.1 Opis szczegółowy

Plik zawiera definicję klasy `tabx2`. Modeluje główne pojęcia związane z programem

Definicja w pliku `tabx2.hh`.

6.8.2 Dokumentacja definicji

6.8.2.1 `#define TABX2_HH`

Definicja w linii 2 pliku `tabx2.hh`.

Skorowidz

- ~tabx2
 - tabx2, [13](#)
- analyze
 - benchmark, [10](#)
- benchmark, [9](#)
 - analyze, [10](#)
 - test, [11](#)
- benchmark.cpp, [15](#)
- benchmark.hh, [15](#)
- data_generator
 - generator.cpp, [17](#)
 - generator.hh, [18](#)
- generator.cpp, [16](#)
 - data_generator, [17](#)
- generator.hh, [17](#)
 - data_generator, [18](#)
- main
 - main.cpp, [19](#)
- main.cpp, [18](#)
 - main, [19](#)
- size
 - tabx2, [13](#)
- strona.dox, [19](#)
- TABX2_HH
 - tabx2.hh, [21](#)
- tab
 - tabx2, [13](#)
- tabx2, [12](#)
 - ~tabx2, [13](#)
 - size, [13](#)
 - tab, [13](#)
 - tabx2, [13](#)
 - test, [13](#)
- tabx2.cpp, [19](#)
- tabx2.hh, [20](#)
 - TABX2_HH, [21](#)
- test
 - benchmark, [11](#)
 - tabx2, [13](#)