Podstawowe struktury danych

1

Wygenerowano przez Doxygen 1.8.6

Śr, 18 mar 2015 08:38:14

Spis treści

Rozdział 1

Sprawozdanie

Data

11.03.2015r.

Wersja

0.1

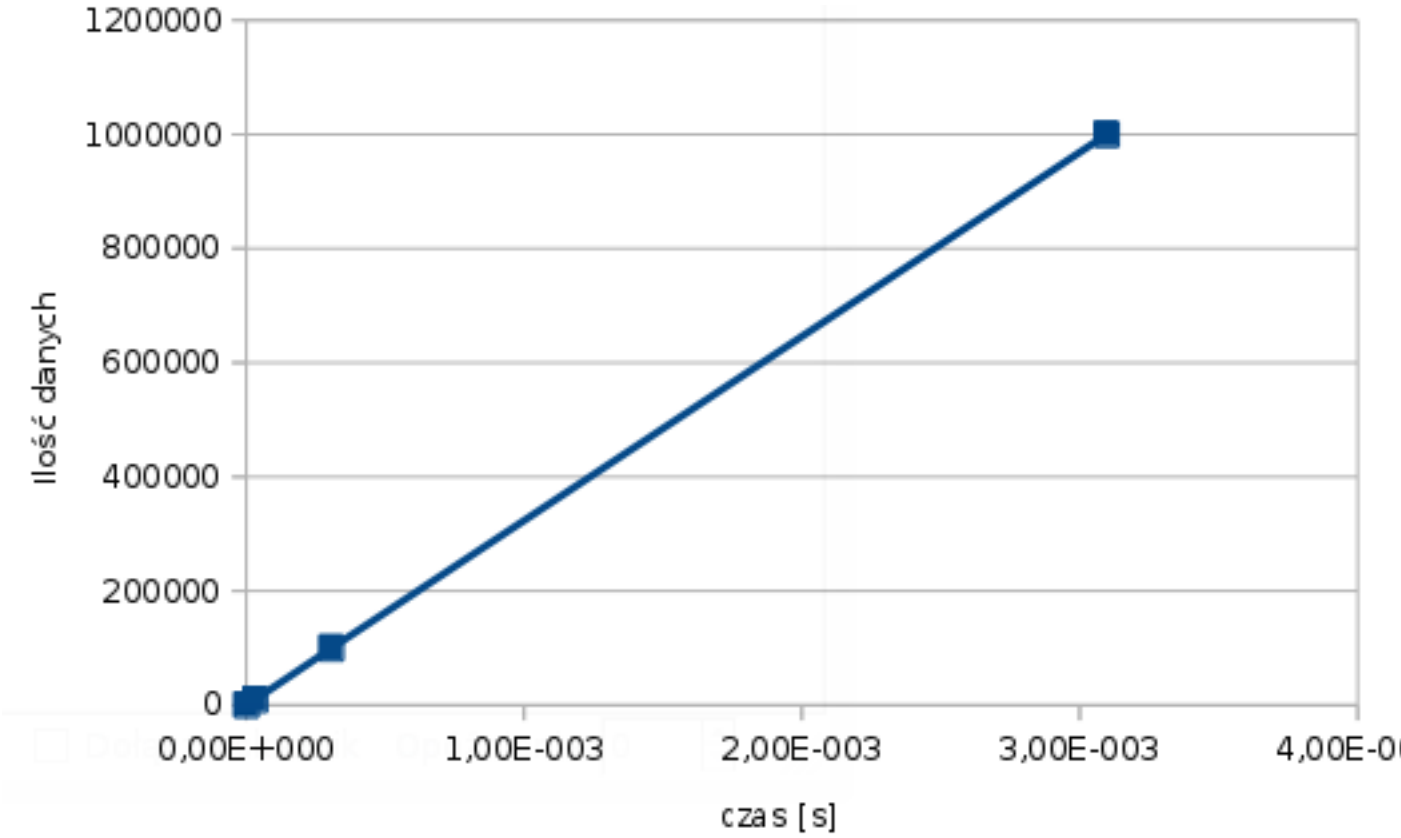
1.1 Zadanie

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 maxymalną 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 wiekszej ilości danych wyniki są poprawne Wydaje się , że zbyt mało danych jest obecnych w środkowej częsci wykresu co powinno być zostać zmienione w celu poprawy jakości odbioru wykresu (dla charakterystyki liniowej jest to akurat bez znaczenia ale .np dla logarytmicznej było by widoczne).

Rozdział 2

Indeks hierarchiczny

2.1 Hierarchia klas

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

Rozdział 3

Indeks klas

3.1 Lista klas

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

Rozdział 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

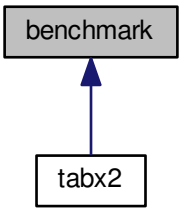
Rozdział 5

Dokumentacja klas

5.1 Dokumentacja klasy benchmark

`#include <benchmark.hh>`

Diagram dziedziczenia dla benchmark



Metody publiczne

- void analyze (int repeat, int data_amount)

  Metoda analyze zlicza czas wykonywania funkcji test()
  Uwaga! do poprawnego działania wymagane jest posiadanie programu gnuplot.

Metody prywatne

- virtual void test (int length)=0

  Metoda test funkcja wirtualna , której czas działania ma być aproksymowany przez metoda analyze()

5.1.1 Opis szczegółowy

Definicja w linii 11 pliku benchmark.hh.

5.1.2 Dokumentacja funkcji składowych

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

Metoda analyze zlicza czas fukcji test()

Przykład wywołania funkcji :

analyze(100,7) -> Przeprowadza analize czesu trwania funkcji test() dla 1 miliona danych , każdy czas trwania funkcji jest ustalany na podstawie średniej arytmetycznej ze 100 prób.

Parametry

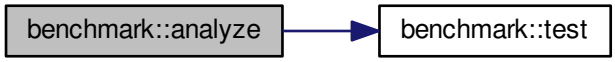
in	*repeat*	- ilość powtórzeń testu
in	*data_amount*	- ilosc wynikowych danych podawana jako potega liczby 10

Zwraca

plik.dat z czasami poszczegolnych pomiarow oraz ilosc testowanych danych oraz plik plot.png bedacy graficznym przedstawieniem danych na wykresie

Definicja w linii 15 pliku benchmark.cpp.

Oto graf wywołań dla tej funkcji:

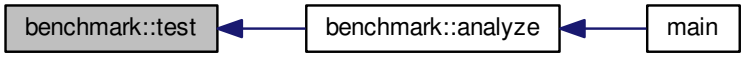


Oto graf wywoływań tej funkcji:

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

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

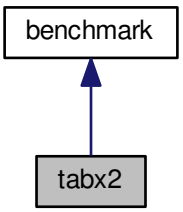
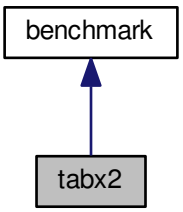


Diagram współpracy dla tabx2:



Metody publiczne

- void test (int length)

  Tworzy tablice alokowana dynamicznie o pojemnosci wybranej przez uzytkownika , umozliwia wykonywanie mnozenia przez 2 wszystkich elementow tablicy.

- tabx2 (int size)

  Konstruktor paramteryczny.

- ~tabx2 ()

  Zwykly destruktor.

Atrybuty prywatne

- int size

  Rozmiar tablicy.
- int * tab

  Wskaznik na tablice przechowujaca dane.

5.2.1 Opis szczegółowy

Definicja w linii 14 pliku tabx2.hh.

5.2.2 Dokumentacja konstruktora i destruktora

5.2.2.1 tabx2::tabx2 (int *size*)

Konstruktor wczytujacy okreslona ilosc danych i alokujacy je dynamicznie

Parametry

in	*size*	- dlugosc tablice

Definicja w linii 21 pliku tabx2.cpp.

5.2.2.2 tabx2::~tabx2 ( )

Definicja w linii 41 pliku tabx2.cpp.

5.2.3 Dokumentacja funkcji składowych

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

Mnozy okreslona ilosc danych przez 2

Parametry

in	*length*	- ilosc danych do przemnozenia

Implementuje benchmark.

Definicja w linii 13 pliku tabx2.cpp.

5.2.4 Dokumentacja atrybutów składowych

5.2.4.1 int tabx2::size [private]

Definicja w linii 22 pliku tabx2.hh.

5.2.4.2 int* tabx2::tab [private]

Definicja w linii 25 pliku tabx2.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- tabx2.hh
- tabx2.cpp

Rozdział 6

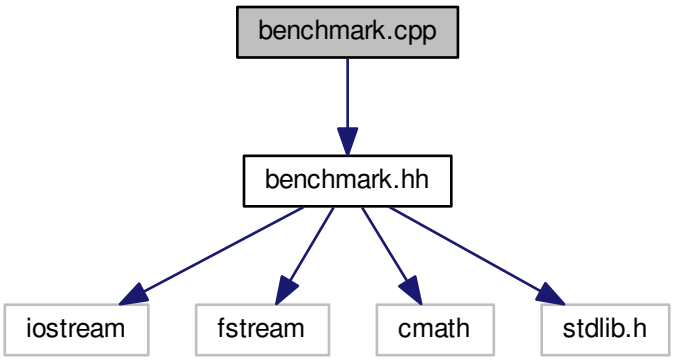
Dokumentacja plików

6.1 Dokumentacja pliku benchmark.cpp

Deklaracja funkzji z klasy Benchmark.

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

Wykres zależności załączania dla benchmark.cpp:

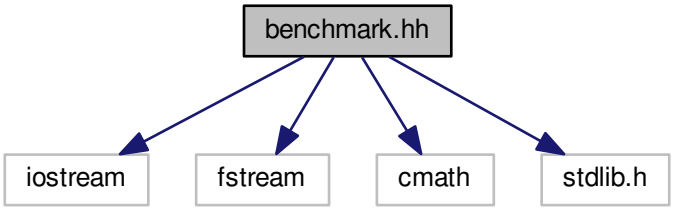


6.2 Dokumentacja pliku benchmark.hh

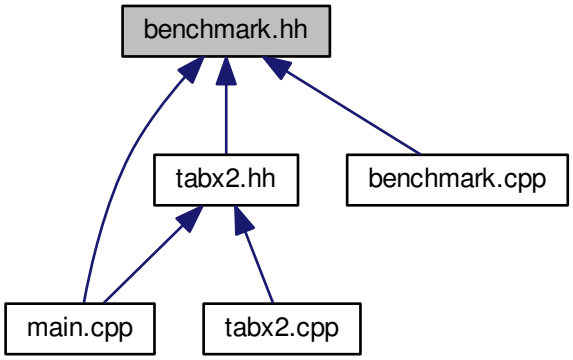
Definicja klasy Benchmark.

```
#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:



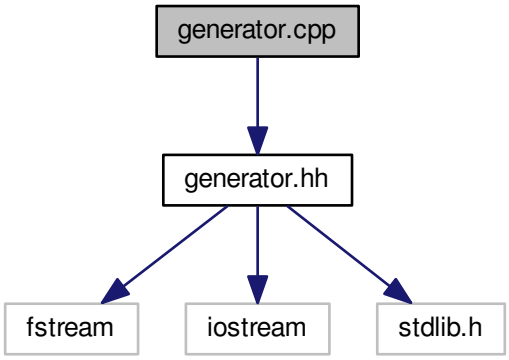
Komponenty

- class benchmark

6.3 Dokumentacja pliku generator.cpp

Deklaracja funckji generujacej liczby losowe.

```
#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, ktore nastepnie sa zapisywane do pliku random_-data.dat

Parametry

in	*data_amount*	- ilosc liczb wynikowych ktore chcemy uzyskac

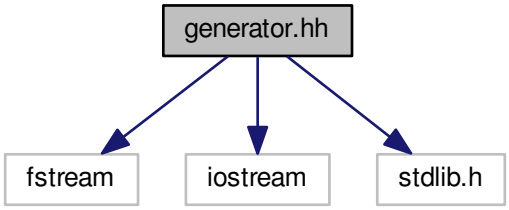
Definicja w linii 9 pliku generator.cpp.

6.4 Dokumentacja pliku generator.hh

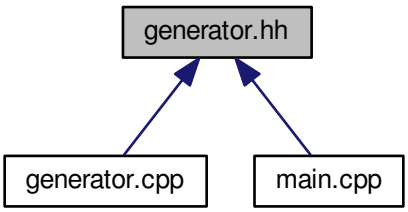
Definicja generatora liczb losowych.

```
#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, ktore nastepnie sa zapisywane do pliku random_-data.dat

Parametry

in	*data_amount*	- ilosc liczb wynikowych ktore chcemy uzyskac

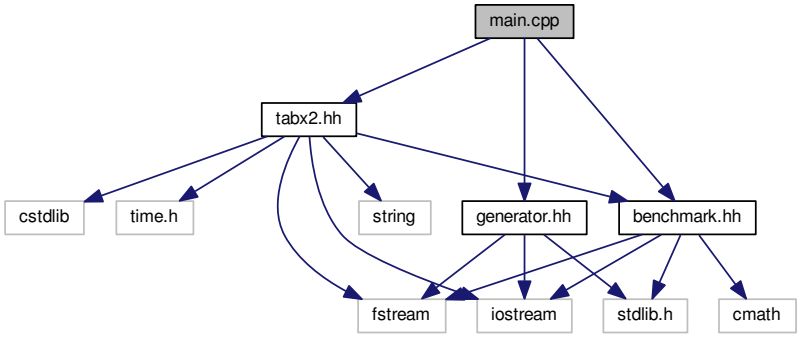
Definicja w linii 9 pliku generator.cpp.

6.5 Dokumentacja pliku main.cpp

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

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

Wykres zależności załączania dla main.cpp:



Funkcje

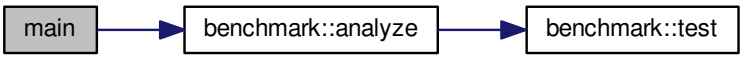
- int main ()

6.5.1 Dokumentacja funkcji

6.5.1.1 int main ( )

Definicja w linii 6 pliku main.cpp.

Oto graf wywołań dla tej funkcji:

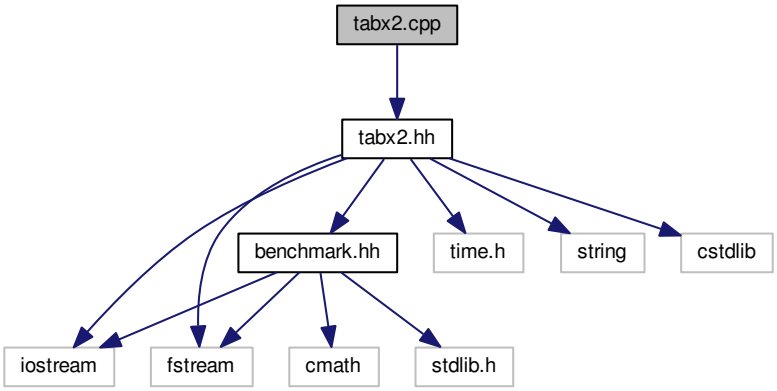


6.6 Dokumentacja pliku strona.dox

6.7 Dokumentacja pliku tabx2.cpp

Deklaracja klasy tabx2.

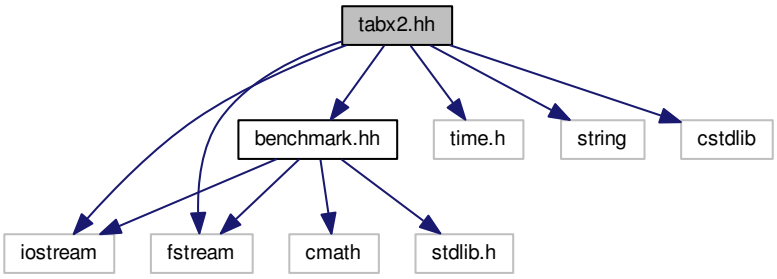
```
#include "tabx2.hh"
```
Wykres zależności załączania dla tabx2.cpp:



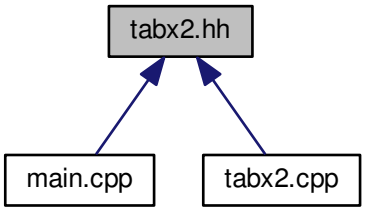
6.8 Dokumentacja pliku tabx2.hh

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 Dokumentacja definicji

6.8.1.1 #define TABX2_HH

Definicja w linii 2 pliku tabx2.hh.

Skorowidz