My Project

Generated by Doxygen 1.8.6

Thu Mar 12 2015 01:26:34

Contents

1 Laboratorium 1 **1**

2 Hierarchical Index **3**

2.0.1 Hierarchia klas . . . 3

3 Class Index **5**

3.0.2 Lista klas . . . 5

4 File Index **7**

4.0.3 Lista plików . . . 7

5 Class Documentation **9**

5.0.4 Dokumentacja klasy DataFrame . . . 9

5.0.4.1 Opis szczegółowy . . . 9

5.0.4.2 Dokumentacja konstruktora i destruktora . . . 10

5.0.4.3 Dokumentacja funkcji składowych . . . 10

5.0.4.4 Dokumentacja atrybutów składowych . . . 10

5.0.5 Dokumentacja klasy MultiplyByTwo . . . 10

5.0.5.1 Opis szczegółowy . . . 11

5.0.5.2 Dokumentacja konstruktora i destruktora . . . 11

5.0.5.3 Dokumentacja funkcji składowych . . . 11

5.0.6 Dokumentacja klasy MyBenchmark . . . 11

5.0.6.1 Opis szczegółowy . . . 12

5.0.6.2 Dokumentacja konstruktora i destruktora . . . 12

5.0.6.3 Dokumentacja funkcji składowych . . . 12

5.0.7 Dokumentacja klasy NumberGenerator . . . 12

5.0.7.1 Opis szczegółowy . . . 13

5.0.7.2 Dokumentacja konstruktora i destruktora . . . 13

5.0.7.3 Dokumentacja funkcji składowych . . . 13

6 File Documentation **15**

6.0.8 Dokumentacja pliku dataframe.h . . . 15

6.0.9 Dokumentacja pliku multiplybytwo.h . . . 15

6.0.10 Dokumentacja pliku mybenchmark.h 15
6.0.11 Dokumentacja pliku numbergenerator.h 15
6.0.12 Dokumentacja pliku dataframe.cpp 16
6.0.13 Dokumentacja pliku main.cpp 16
6.0.13.1 Dokumentacja funkcji 16
6.0.14 Dokumentacja pliku multiplybytwo.cpp 16
6.0.15 Dokumentacja pliku mybenchmark.cpp 16

Chapter 1

Laboratorium 1

Aplikacja umozliwia uzytkownikowi na przeprowadzenia algorytmu mnozenia

przez dwa na dowolnej liczbie elementow.

Najważniejsze cechy

Możliwość włączenia opcji benchmarkującej służącej do sprawdzenia

ile czasu wykonywal sie dany algorytm lub seria tego samego algorytmu

Argumenty wywołania

```
-n liczba       Ilość liczb do odczytania/przerobienia przez algorytm
-t liczba       Włącza opcje benchmarkującą dla seri powtorzen
-o tekst        Wprowadza nazwe pliku do zapisu
-i tekst        Wprowadza nazwe pliku do odczytu
-g              Generuje n liczb i zapisuje je do pliku (po wygenerowaniu konczy program)
```

Chapter 2

Hierarchical Index

2.0.1 Hierarchia klas

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

DataFrame 9
MultiplyByTwo 10
NumberGenerator 12
MyBenchmark 11
MultiplyByTwo 10

Chapter 3

Class Index

3.0.2 Lista klas

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

DataFrame 9
MultiplyByTwo
Algorytm mnozy kazda liczbe razy 2 10
MyBenchmark
Klasa bazowa/interface do testowania algorytmu 11
NumberGenerator
Klasa generujaca losowe liczby 12

Chapter 4

File Index

4.0.3 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

dataframe.cpp 16
dataframe.d . . . . . **??**
dataframe.h 15
main.cpp 16
main.d . . . . . **??**
multiplybytwo.cpp 16
multiplybytwo.d . . . . . **??**
multiplybytwo.h 15
mybenchmark.cpp 16
mybenchmark.d . . . . . **??**
mybenchmark.h 15
numbergenerator.h 15

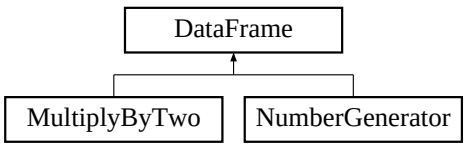
Chapter 5

Class Documentation

5.0.4 Dokumentacja klasy DataFrame

`#include <dataframe.h>`

Diagram dziedziczenia dla DataFrame



Metody publiczne

- int loadDataFromFile ()

  Ładuje dane z pliku.
- int saveDataToFile ()

  Zapisuje dane do pliku.
- DataFrame operator= (DataFrame dataframe)

  Kopiuje elementy roznych obiektow.
- virtual ~DataFrame ()

Atrybuty publiczne

- int * tableOfData

  Zawiera adres do tablicy {size} elementów.
- char * outputFileName

  Zawiera nazwe pliku do zapisu.
- char * inputFileName

  Zawiera nazwe pliku do odczytu.
- unsigned int sizeOfTable

  Rozmiar tablicy tableOfData.

5.0.4.1 Opis szczegółowy

Definicja w linii 15 pliku dataframe.h.

5.0.4.2 Dokumentacja konstruktora i destruktora

5.0.4.2.1 virtual DataFrame::~DataFrame ( ) [inline], [virtual]

Definicja w linii 60 pliku dataframe.h.

```
00060 {}
```

5.0.4.3 Dokumentacja funkcji składowych

5.0.4.3.1 int DataFrame::loadDataFromFile ( )

Wczytuje dane z pliku i zapisuje je dynamicznie do tablicy jednowymiarowej, na ktora wskazuje wskaźnik ∗tableOfData

Rozmiar tablicy jest przechowywany w sizeOfTable

Definicja w linii 13 pliku dataframe.cpp.

Odwołuje się do inputFileName, sizeOfTable i tableOfData.

Odwołania w main().

```
00014 {
00015         std::ifstream streamToFile;
00016         streamToFile.open (inputFileName, std::ifstream::in);
00017         this->tableOfData = new int[sizeOfTable];
00018         for(unsigned int i=0; i<sizeOfTable ; i++) {
00019                 streamToFile >> this-> tableOfData[i];
00020                 if (streamToFile.eof()) return 1; //[EoF reached]
00021         }
00022         return 0;
00023 }
```

5.0.4.3.2 DataFrame DataFrame::operator= (DataFrame *dataframe*)

Zapisuje kolejne liczby do pliku o nazwie outputFileName

Definicja w linii 37 pliku dataframe.cpp.

Odwołuje się do inputFileName, outputFileName, sizeOfTable i tableOfData.

```
00038 {
00039         this->tableOfData = dataframe.tableOfData;
00040         this->outputFileName = dataframe.outputFileName;
00041         this->inputFileName = dataframe.inputFileName;
00042         this->sizeOfTable = dataframe.sizeOfTable;
00043         return *this;
00044 }
```

5.0.4.3.3 int DataFrame::saveDataToFile ( )

Wczytuje liczby z pliku o nazwie intputFileName

Definicja w linii 25 pliku dataframe.cpp.

Odwołuje się do outputFileName, sizeOfTable i tableOfData.

Odwołania w main().

```
00026 {
00027         std::ofstream streamToFile;
00028         streamToFile.open (outputFileName, std::ofstream::out);
00029         for(unsigned int i=0; i<sizeOfTable ; i++) {
00030                 streamToFile << this-> tableOfData[i] <<' ';
00031         }
00032         return 0;
00033 }
```

5.0.4.4 Dokumentacja atrybutów składowych

5.0.4.4.1 char* DataFrame::inputFileName

Definicja w linii 29 pliku dataframe.h.

Odwołania w loadDataFromFile(), main() i operator=().

5.0.4.4.2 char* DataFrame::outputFileName

Definicja w linii 25 pliku dataframe.h.

Odwołania w main(), operator=() i saveDataToFile().

5.0.4.4.3 unsigned int DataFrame::sizeOfTable

Definicja w linii 34 pliku dataframe.h.

Odwołania w MultiplyByTwo::executeAlgorithm(), NumberGenerator::generateNumbers(), loadDataFromFile(), main(), operator=() i saveDataToFile().

5.0.4.4.4 int* DataFrame::tableOfData

Definicja w linii 21 pliku dataframe.h.

Odwołania w MultiplyByTwo::executeAlgorithm(), NumberGenerator::generateNumbers(), loadDataFromFile(), operator=() i saveDataToFile().

Dokumentacja dla tej klasy została wygenerowana z plików:

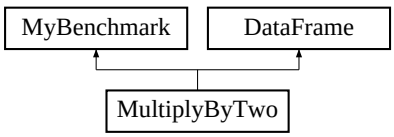
- dataframe.h
- dataframe.cpp

5.0.5 Dokumentacja klasy MultiplyByTwo

Algorytm mnozy kazda liczbe razy 2.

`#include <multiplybytwo.h>`

Diagram dziedziczenia dla MultiplyByTwo



Metody publiczne

- void executeAlgorithm ()

  Wykonuje algorytm mnozenie x2.
- ~MultiplyByTwo ()

Dodatkowe Dziedziczone Składowe

5.0.5.1 Opis szczegółowy

Algorytm mnozy kazda kolejna liczbe przez 2

Definicja w linii 20 pliku multiplybytwo.h.

5.0.5.2 Dokumentacja konstruktora i destruktora

5.0.5.2.1 MultiplyByTwo::~MultiplyByTwo ( ) [inline]

Definicja w linii 29 pliku multiplybytwo.h.

```
00029 {}
```

5.0.5.3 Dokumentacja funkcji składowych

5.0.5.3.1 void MultiplyByTwo::executeAlgorithm ( ) [virtual]

Implementuje MyBenchmark.

Definicja w linii 11 pliku multiplybytwo.cpp.

Odwołuje się do DataFrame::sizeOfTable i DataFrame::tableOfData.

Odwołania w main().

```
00012 {
00013         for(unsigned int i=0; i<sizeOfTable; i++) {
00014
00015                 tableOfData[i]*=2;
00016         }
00017
00018
00019
00020 }
```

Dokumentacja dla tej klasy została wygenerowana z plików:

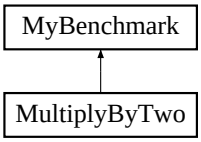
- multiplybytwo.h
- multiplybytwo.cpp

5.0.6 Dokumentacja klasy MyBenchmark

Klasa bazowa/interface do testowania algorytmu.

`#include <mybenchmark.h>`

Diagram dziedziczenia dla MyBenchmark



Metody publiczne

- double testAlgorithm (unsigned int repetition)

  Benchmarkuje algorytm główny.
- virtual ~MyBenchmark ()

  Usuwam obiekt test biorąc pod uwage jego prawdziwy typ.

Metody chronione

- virtual void executeAlgorithm ()=0

  Interface metody algorytmu glownego.

5.0.6.1 Opis szczegółowy

Używana jako interface dla wszystkich algorytmow aby testowac czas wykonywanego algorymtu.

Definicja w linii 20 pliku mybenchmark.h.

5.0.6.2 Dokumentacja konstruktora i destruktora

5.0.6.2.1 virtual MyBenchmark::~MyBenchmark ( ) [inline], [virtual]

Definicja w linii 49 pliku mybenchmark.h.

```
00049 {};
```

5.0.6.3 Dokumentacja funkcji składowych

5.0.6.3.1 virtual void MyBenchmark::executeAlgorithm ( ) [protected], [pure virtual]

Metoda abstrakcyjna, ktora jest interfacem do implementacji przez glowny algorytm. To znaczy, ze kazdy algorytm ma byc uruchamiany tą funkcja

Implementowany w MultiplyByTwo.

Odwołania w testAlgorithm().

5.0.6.3.2 double MyBenchmark::testAlgorithm (unsigned int *repetition*)

Obliczam czas wykonywanego algorytmu dzięki zastosowaniu metody abstrakcyjnej executeAlgorithm() i zaimplementowaniu tego interfacu w algorytmie głównym

Definicja w linii 12 pliku mybenchmark.cpp.

Odwołuje się do executeAlgorithm().

Odwołania w main().

```
00013 {
00014         time_t benchmarkTime = clock();
00015
00016         for(unsigned int i=0 ; i<repetition ; i++)      {
00017                 this->executeAlgorithm();
00018         }
00019
00020         benchmarkTime = clock() - benchmarkTime;
00021
00022         return (( (double)benchmarkTime ) /CLOCKS_PER_SEC);
00023 }
```

Dokumentacja dla tej klasy została wygenerowana z plików:

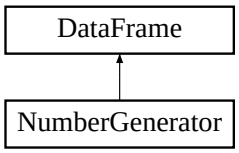
- mybenchmark.h
- mybenchmark.cpp

5.0.7 Dokumentacja klasy NumberGenerator

Klasa generujaca losowe liczby.

`#include <numbergenerator.h>`

Diagram dziedziczenia dla NumberGenerator



Metody publiczne

- void generateNumbers ()

  Generuje losowe liczby.
- ~NumberGenerator ()

Dodatkowe Dziedziczone Składowe

5.0.7.1 Opis szczegółowy

Klasa generujaca losowe liczby na podstawie czasu maszyny na ktorym jest uruchomiona Wszystkie funkcje zapisu pliku dziedziczy z klasy DataFrame

Definicja w linii 23 pliku numbergenerator.h.

5.0.7.2 Dokumentacja konstruktora i destruktora

5.0.7.2.1 NumberGenerator::~NumberGenerator ( ) [inline]

Definicja w linii 44 pliku numbergenerator.h.

```
00044 {}
```

5.0.7.3 Dokumentacja funkcji składowych

5.0.7.3.1 void NumberGenerator::generateNumbers ( ) [inline]

Generuje losowe liczby na podstawie czasu maszyny

Definicja w linii 31 pliku numbergenerator.h.

Odwołuje się do DataFrame::sizeOfTable i DataFrame::tableOfData.

Odwołania w main().

```
00032 {
00033         time_t randomTime = clock();
00034         this->tableOfData = new int[sizeOfTable];
00035         for(int i=0; i<sizeOfTable ; i++)
00036         {
00037                 srand (randomTime = clock());
00038                 this->tableOfData[i] = rand()%100;
00039                 randomTime = clock();
00040         }
00041 }
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- numbergenerator.h

Chapter 6

File Documentation

6.0.8 Dokumentacja pliku dataframe.h

```
#include <fstream>
```

Komponenty

- class DataFrame

6.0.9 Dokumentacja pliku multiplybytwo.h

```
#include "mybenchmark.h"
#include "dataframe.h"
```

Komponenty

- class MultiplyByTwo

  Algorytm mnozy kazda liczbe razy 2.

6.0.10 Dokumentacja pliku mybenchmark.h

```
#include <ctime>
```

Komponenty

- class MyBenchmark

  Klasa bazowa/interface do testowania algorytmu.

6.0.11 Dokumentacja pliku numbergenerator.h

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "dataframe.h"
```

Komponenty

- class NumberGenerator

  Klasa generujaca losowe liczby.

6.0.12 Dokumentacja pliku dataframe.cpp

```
#include "dataframe.h"
```

6.0.13 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <unistd.h>
#include <stdlib.h>
#include "multiplybytwo.h"
#include "numbergenerator.h"
#include "dataframe.h"
```

Funkcje

- int main (int argc, char ∗argv[])

6.0.13.1 Dokumentacja funkcji

6.0.13.1.1 int main (int *argc,* char ∗ *argv[]*)

Ilosc powtorzen przez algorytmu

Zmienna uzywana przez GETOPT

Flaga ktora mowi o tym czy wlaczyc generator liczb losowych

Definicja w linii 16 pliku main.cpp.

Odwołuje się do MultiplyByTwo::executeAlgorithm(), NumberGenerator::generateNumbers(), DataFrame::inputFileName, DataFrame::loadDataFromFile(), DataFrame::outputFileName, DataFrame::saveDataToFile(), DataFrame::sizeOfTable i MyBenchmark::testAlgorithm().

```
00017 {
00018         DataFrame podstawoweInfoIO;
00019         int quantityRepetitionOfAlgorithm = 0;
00020
00021         int opt;
00022         bool isSetNumberGenerator=false;
00023
00024         while ((opt = getopt(argc, argv, "n:t:o:i:g")) != -1) {
00025                 switch(opt){
00026                 case 'n':
00027                         podstawoweInfoIO.sizeOfTable = atoi(optarg);
00028                         break;
00029
00030                 case 't':
00031                         quantityRepetitionOfAlgorithm = atoi(optarg);
00032                         break;
00033
00034                 case 'o':
00035                         podstawoweInfoIO.outputFileName = optarg;
00036                         break;
00037
00038                 case 'i':
00039                         podstawoweInfoIO.inputFileName=optarg;
00040                         break;
00041
```

```
00042                 case 'g':
00043                         isSetNumberGenerator=true;
00044                         break;
00045
00046                 case '?':
00047                 default:
00048                         std::cout<<"\nPodano zly argument";
00049                         return -1;
00050
00051                 }
00052         }
00053         /*
00054          * Sprawdzam czy program zostal uzyty tylko do wygenerowania liczb losowych
00055          * jesli tak to tworze te liczby zgodnie quantityNumber i zamykam program
00056          */
00057         if(isSetNumberGenerator) {
00058         NumberGenerator generator;
00059         generator= podstawoweInfoIO;
00060         generator.generateNumbers();
00061         generator.saveDataToFile();
00062         std::cout<<"\nZapisane.\n";
00063         return 0;
00064         }
00065
00066
00067         MultiplyByTwo algorytm_x2 ;
00068         algorytm_x2= podstawoweInfoIO;
00069
00070         // Laduje liczby do przeprowadzenia algorytmu
00071         if(algorytm_x2.loadDataFromFile()) {
00072                 std::cout<<"\nNie istnieje tyle liczb w pliku !\nKoncze program";
00073                 return 1;
00074         }
00075
00076         /*
00077          * Sprawdzam czy otrzymalem agrument o testowaniu algorytmu,
00078          * a nastepnie przeprawadzam test albo uruchamiam normalnie algorytm
00079          */
00080         if(quantityRepetitionOfAlgorithm){
00081                 std::cout<<"\nCzas algorytmu: "<<algorytm_x2.testAlgorithm(
      quantityRepetitionOfAlgorithm)<<'\n';
00082         }
00083         else {
00084                 algorytm_x2.executeAlgorithm();
00085         }
00086
00087
00088
00089         algorytm_x2.saveDataToFile();
00090
00091         return 0;
00092 }
```

6.0.14 Dokumentacja pliku multiplybytwo.cpp

```
#include "multiplybytwo.h"
```

6.0.15 Dokumentacja pliku mybenchmark.cpp

```
#include "mybenchmark.h"
```