

Laboratorium 2

0.2

Wygenerowano przez Doxygen 1.8.6

Wt, 17 mar 2015 00:16:04

Spis treści

1 Laboratorium 2

Aplikacja umożliwia użytkownikowi na przeprowadzenia algorytmu mnożenia przez dwa na dowolnej liczbie elementów.

Najważniejsze cechy

Możliwość włączenia opcji benchmarkującej służącej do sprawdzenia ile czasu wykonywał się dany algorytm lub seria tego samego algorytmu

Argumenty wywołania

-n liczba	Ilość liczb do odczytania/przerobienia przez algorytm
-t liczba	Włącza opcje benchmarkującą dla serii powtórzeń
-o tekst	Wprowadza nazwę pliku do zapisu
-i tekst	Wprowadza nazwę pliku do odczytu
-g	Generuje n liczb i zapisuje je do pliku (po wygenerowaniu kończy program)

2 Indeks hierarchiczny

2.1 Hierarchia klas

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

DataFrame	??
MyBenchmark	??
MultiplyByTwo	??
NumberGenerator	??
MyList	??
MyQueue	??
MyStack	??
MyList::MyListElement	??

3 Indeks klas

3.1 Lista klas

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

DataFrame	??
MultiplyByTwo	??
Algorytm mnoży każdą liczbę razy 2	??
MyBenchmark	??
Klasa bazowa/interface do testowania algorytmu	??

MyList	
Lista dwukierunkowa	??
MyList::MyListElement	
Klasa 'małych struktur' gdzie jest numer i wskaźnik do naszego elementu	??
MyQueue	
Klasa reprezentuje kolejkę	??
MyStack	
Klasa reprezentuje stos	??
NumberGenerator	
Klasa generująca losowe liczby	??

4 Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

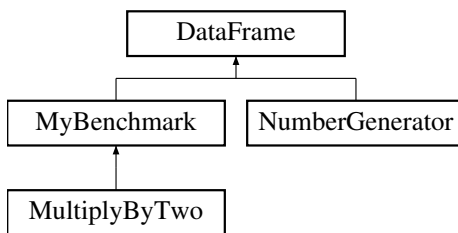
dataframe.cpp	??
dataframe.h	??
main.cpp	??
multiplybytwo.cpp	??
multiplybytwo.h	??
mybenchmark.cpp	??
mybenchmark.h	??
mylist.cpp	??
mylist.h	??
myqueue.h	??
mystack.h	??
numbergenerator.h	??

5 Dokumentacja klas

5.1 Dokumentacja klasy DataFrame

```
#include <dataframe.h>
```

Diagram dziedziczenia dla DataFrame



Metody publiczne

- `DataFrame ()`
Przypisuje zmiennym wartosci domyslne.
- `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.

5.1.1 Opis szczegółowy

Definicja w linii 15 pliku `dataframe.h`.

5.1.2 Dokumentacja konstruktora i destruktor

5.1.2.1 DataFrame::DataFrame ()

Definicja w linii 12 pliku `dataframe.cpp`.

Odwołuje się do `inputFileName`, `outputFileName`, `sizeOfTable` i `tableOfData`.

```

00013 {
00014     tableOfData = 0;
00015     outputFileName = NULL;
00016     inputFileName = NULL;
00017     sizeOfTable = 0;
00018 }
  
```

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

Definicja w linii 64 pliku `dataframe.h`.

```

00064 {}
  
```

5.1.3 Dokumentacja funkcji składowych

5.1.3.1 int DataFrame::loadDataFromFile ()

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

Rozmiar tablicy jest przechowywany w sizeOfTable

Definicja w linii 20 pliku [dataframe.cpp](#).

Odwołuje się do [inputFileName](#), [sizeOfTable](#) i [tableOfData](#).

Odwołania w [main\(\)](#).

```
00021 {
00022     std::ifstream streamToFile;
00023     streamToFile.open (inputFileName, std::ifstream::in);
00024     this->tableOfData = new int[sizeOfTable];
00025     for(unsigned int i=0; i<sizeOfTable ; i++) {
00026         streamToFile >> this-> tableOfData[i];
00027         if (streamToFile.eof()) return 1; //[EoF reached]
00028     }
00029     return 0;
00030 }
```

5.1.3.2 DataFrame DataFrame::operator= (DataFrame dataframe)

Zapisuje kolejne liczby do pliku o nazwie outputFileName

Definicja w linii 44 pliku [dataframe.cpp](#).

Odwołuje się do [inputFileName](#), [outputFileName](#), [sizeOfTable](#) i [tableOfData](#).

```
00045 {
00046     this->tableOfData = dataframe.tableOfData;
00047     this->outputFileName = dataframe.outputFileName;
00048     this->inputFileName = dataframe.inputFileName;
00049     this->sizeOfTable = dataframe.sizeOfTable;
00050     return *this;
00051 }
```

5.1.3.3 int DataFrame::saveDataToFile ()

Wczytuje liczby z pliku o nazwie inputFileName

Definicja w linii 32 pliku [dataframe.cpp](#).

Odwołuje się do [outputFileName](#), [sizeOfTable](#) i [tableOfData](#).

Odwołania w [main\(\)](#).

```
00033 {
00034     std::ofstream streamToFile;
00035     streamToFile.open (outputFileName, std::ofstream::out);
00036     for(unsigned int i=0; i<sizeOfTable ; i++) {
00037         streamToFile << this-> tableOfData[i] << ' ';
00038     }
00039     return 0;
00040 }
```

5.1.4 Dokumentacja atrybutów składowych

5.1.4.1 char* DataFrame::inputFileName

Definicja w linii 29 pliku [dataframe.h](#).

Odwołania w [DataFrame\(\)](#), [loadDataFromFile\(\)](#), [main\(\)](#) i [operator=\(\)](#).

5.1.4.2 char* DataFrame::outputFileName

Definicja w linii 25 pliku [dataframe.h](#).

Odwołania w [DataFrame\(\)](#), [main\(\)](#), [operator=\(\)](#) i [saveDataToFile\(\)](#).

5.1.4.3 unsigned int DataFrame::sizeOfTable

Definicja w linii 34 pliku [dataframe.h](#).

Odwołania w [DataFrame\(\)](#), [MultiplyByTwo::executeAlgorithm\(\)](#), [NumberGenerator::generateNumbers\(\)](#), [loadDataFromFile\(\)](#), [main\(\)](#), [operator=\(\)](#), [saveDataToFile\(\)](#) i [MyBenchmark::testAlgorithm\(\)](#).

5.1.4.4 int* DataFrame::tableOfData

Definicja w linii 21 pliku [dataframe.h](#).

Odwołania w [DataFrame\(\)](#), [MultiplyByTwo::executeAlgorithm\(\)](#), [NumberGenerator::generateNumbers\(\)](#), [loadDataFromFile\(\)](#), [operator=\(\)](#), [saveDataToFile\(\)](#) i [MyBenchmark::testAlgorithm\(\)](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

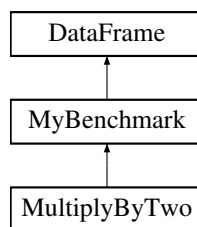
- [dataframe.h](#)
- [dataframe.cpp](#)

5.2 Dokumentacja klasy MultiplyByTwo

Algorytm mnoży każdą liczbę razy 2.

```
#include <multiplybytwo.h>
```

Diagram dziedziczenia dla MultiplyByTwo



Metody publiczne

- void [executeAlgorithm\(\)](#)
Wykonuje algorytm mnożenie x2.
- [~MultiplyByTwo\(\)](#)

Dodatkowe Dziedziczone Składowe

5.2.1 Opis szczegółowy

Algorytm mnoży każdą kolejną liczbę przez 2

Definicja w linii 20 pliku [multiplybytwo.h](#).

5.2.2 Dokumentacja konstruktora i destruktor

5.2.2.1 MultiplyByTwo::~~MultiplyByTwo () [inline]

Definicja w linii 29 pliku [multiplybytwo.h](#).

```
00029 {}
```

5.2.3 Dokumentacja funkcji składowych

5.2.3.1 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:

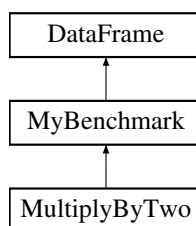
- [multiplybytwo.h](#)
- [multiplybytwo.cpp](#)

5.3 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](#) ()
Usuwa obiekt test biorąc pod uwagę jego prawdziwy typ.

Metody chronione

- virtual void [executeAlgorithm](#) ()=0
Interface metody algorytmu głównego.

Dodatkowe Dziedziczone Składowe

5.3.1 Opis szczegółowy

Używana jako interface dla wszystkich algorytmów aby testować czas wykonywanego algorytmu.

Definicja w linii 20 pliku [mybenchmark.h](#).

5.3.2 Dokumentacja konstruktora i destruktora

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

Definicja w linii 49 pliku [mybenchmark.h](#).

```
00049 {};
```

5.3.3 Dokumentacja funkcji składowych

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

Metoda abstrakcyjna, która jest interfacem do implementacji przez główny algorytm. To znaczy, że każdy algorytm ma być uruchamiany tą funkcją

Implementowany w [MultiplyByTwo](#).

Odwołania w [testAlgorithm\(\)](#).

5.3.3.2 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\(\)](#), [DataFrame::sizeOfTable](#) i [DataFrame::tableOfData](#).

Odwołania w [main\(\)](#).

```
00013 {
00014     time_t benchmarkTimeInTotal = 0;
00015     time_t benchmarkTimeForOneLoop = 0;
00016
00017     int *originalTableOfData = new int[sizeOfTable];
00018     for(unsigned int i=0; i<sizeOfTable; i++)
00019         originalTableOfData[i]=tableOfData[i];
00020
00021     for(unsigned int i=0 ; i<repetition ; i++) {
00022         for(unsigned int i=0; i<sizeOfTable; i++)
00023             tableOfData[i]=originalTableOfData[i];
00024
00025         benchmarkTimeForOneLoop = clock();
00026         this->executeAlgorithm();
00027         benchmarkTimeInTotal += clock() - benchmarkTimeForOneLoop;
00028     }
00029
00030
00031     return (( (double)benchmarkTimeInTotal ) /CLOCKS_PER_SEC);
00032 }
```

Dokumentacja dla tej klasy została wygenerowana z plików:

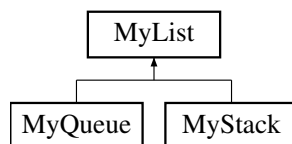
- [mybenchmark.h](#)
- [mybenchmark.cpp](#)

5.4 Dokumentacja klasy MyList

Lista dwukierunkowa.


```
#include <mylist.h>
```

Diagram dziedziczenia dla MyList



Komponenty

- class `MyListElement`
Klasa 'małych struktur' gdzie jest numer i wskaźnik do nas elementu.

Metody publiczne

- `MyList ()`
Konstruktor listy.
- `int size ()`
Zwraca ilość elementów listy.
- `int pop_back ()`
Zwraca element ostatni w liście.
- `int pop_front ()`
Zwraca element pierwszy w liście.
- `void push_back (int arg)`
Wkłada element na ostatnie miejsce na liście.
- `void push_front (int arg)`
Wkłada element na pierwsze miejsce na liście.

Atrybuty prywatne

- `int sizeOfList`
liczba elementów listy
- `MyListElement * firstElement`
wskaźnik do 'małej struktury' która jest pierwsza na liście
- `MyListElement * lastElement`
wskaźnik do 'małej struktury' która jest ostatnia na liście

5.4.1 Opis szczegółowy

Klasa przedstawia listę dwukierunkową dynamiczną

Definicja w linii 18 pliku `mylist.h`.

5.4.2 Dokumentacja konstruktora i destruktor

5.4.2.1 `MyList::MyList ()`

Definicja w linii 11 pliku `mylist.cpp`.

Odwołuje się do `firstElement`, `lastElement` i `sizeOfList`.

```
00012 {  
00013     firstElement = lastElement = new MyListElement(0);  
00014     sizeOfList = 0;  
00015 }
```

5.4.3 Dokumentacja funkcji składowych

5.4.3.1 int MyList::pop_back ()

Zwraca

Zwraca element ostatni w liście

Definicja w linii 37 pliku [mylist.cpp](#).

Odwołuje się do [lastElement](#) i [sizeOfList](#).

Odwołania w [MyStack::pop\(\)](#).

```
00038 {  
00039     if(!(sizeOfList--)) { sizeOfList=0; return 0; }  
00040     int tmpNumber = this -> lastElement -> number;  
00041     MyListElement *originMyListElement = this -> lastElement;  
00042     this -> lastElement = this -> lastElement -> previousElement;  
00043     delete originMyListElement;  
00044     return tmpNumber;  
00045 }
```

5.4.3.2 int MyList::pop_front ()

Zwraca

Zwraca element pierwszy w liście

Definicja w linii 46 pliku [mylist.cpp](#).

Odwołuje się do [firstElement](#) i [sizeOfList](#).

Odwołania w [MyQueue::pop\(\)](#).

```
00047 {  
00048     if(!(sizeOfList--)) { sizeOfList=0; return 0; }  
00049     int tmpNumber = this -> firstElement -> number;  
00050     MyListElement *originMyListElement = this -> firstElement;  
00051     this -> firstElement = this -> firstElement -> nextElement;  
00052  
00053     delete originMyListElement;  
00054     return tmpNumber;  
00055 }
```

5.4.3.3 void MyList::push_back (int arg)

Definicja w linii 18 pliku [mylist.cpp](#).

Odwołuje się do [firstElement](#), [lastElement](#) i [sizeOfList](#).

Odwołania w [MyQueue::push\(\)](#) i [MyStack::push\(\)](#).

```
00019 {  
00020     MyListElement *newMyListElement = new MyListElement(arg);  
00021     if(!(sizeOfList++)) {firstElement = lastElement = newMyListElement;}  
00022     //newMyListElement -> nextElement = 0;  
00023     newMyListElement -> previousElement = this -> lastElement;  
00024     this -> lastElement -> nextElement = newMyListElement;  
00025     this->lastElement = newMyListElement;  
00026 }
```

5.4.3.4 void MyList::push_front (int arg)

Definicja w linii 27 pliku [mylist.cpp](#).

Odwołuje się do [firstElement](#), [lastElement](#) i [sizeOfList](#).

```
00028 {  
00029     MyListElement *newMyListElement = new MyListElement(arg);  
00030     if(!sizeOfList++) {firstElement = lastElement = newMyListElement;}  
00031     //newMyListElement -> previousElement = 0;  
00032     newMyListElement -> nextElement = this -> firstElement;  
00033     this -> firstElement -> previousElement = newMyListElement;  
00034     this->firstElement = newMyListElement;  
00035 }
```

5.4.3.5 int MyList::size ()

Zwraca

ilosc elementow tablicy

Definicja w linii 70 pliku [mylist.cpp](#).

Odwołuje się do [sizeOfList](#).

```
00071 {  
00072     return sizeOfList;  
00073 }
```

5.4.4 Dokumentacja atrybutów składowych

5.4.4.1 MyListElement* MyList::firstElement [private]

Definicja w linii 43 pliku [mylist.h](#).

Odwołania w [MyList\(\)](#), [pop_front\(\)](#), [push_back\(\)](#) i [push_front\(\)](#).

5.4.4.2 MyListElement* MyList::lastElement [private]

Definicja w linii 45 pliku [mylist.h](#).

Odwołania w [MyList\(\)](#), [pop_back\(\)](#), [push_back\(\)](#) i [push_front\(\)](#).

5.4.4.3 int MyList::sizeOfList [private]

Definicja w linii 22 pliku [mylist.h](#).

Odwołania w [MyList\(\)](#), [pop_back\(\)](#), [pop_front\(\)](#), [push_back\(\)](#), [push_front\(\)](#) i [size\(\)](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

- [mylist.h](#)
- [mylist.cpp](#)

5.5 Dokumentacja klasy MyList::MyListElement

Klasa 'małych struktur' gdzie jest numer i wskaźnik do nas elementu.

Metody publiczne

- [MyListElement](#) (int arg)
Konstruktor wewnętrznej klasy 'małych struktur'.

Atrybuty publiczne

- `int number`
Liczba przechowywana.
- `MyListElement * nextElement`
wskaznik do następnej 'małej struktury' w liście
- `MyListElement * previousElement`
wskaznik do poprzedniej 'małej struktury' w liście

5.5.1 Opis szczegółowy

Definicja w linii 25 pliku `mylist.h`.

5.5.2 Dokumentacja konstruktora i destruktora

5.5.2.1 `MyList::MyListElement::MyListElement (int arg)`

Parametry

<code>arg</code>	liczba do zapisania w kolejnym elemencie listy
------------------	--

Definicja w linii 57 pliku `mylist.cpp`.

Odwołuje się do `nextElement`, `number` i `previousElement`.

```
00058 {
00059     this -> number = arg;
00060     this -> nextElement =0;
00061     this -> previousElement =0;
00062 }
```

5.5.3 Dokumentacja atrybutów składowych

5.5.3.1 `MyListElement* MyList::MyListElement::nextElement`

Definicja w linii 37 pliku `mylist.h`.

Odwołania w `MyListElement()`.

5.5.3.2 `int MyList::MyListElement::number`

Definicja w linii 28 pliku `mylist.h`.

Odwołania w `MyListElement()`.

5.5.3.3 `MyListElement* MyList::MyListElement::previousElement`

Definicja w linii 39 pliku `mylist.h`.

Odwołania w `MyListElement()`.

Dokumentacja dla tej klasy została wygenerowana z plików:

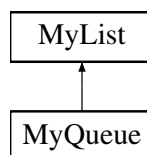
- `mylist.h`
- `mylist.cpp`

5.6 Dokumentacja klasy MyQueue

Klasa reprezentuje kolejke.

```
#include <myqueue.h>
```

Diagram dziedziczenia dla MyQueue



Metody publiczne

- void `push` (int arg)
- int `pop` ()
Wyciąga element z kolejki.

5.6.1 Opis szczegółowy

Definicja w linii 16 pliku `myqueue.h`.

5.6.2 Dokumentacja funkcji składowych

5.6.2.1 int MyQueue::pop () [inline]

Definicja w linii 27 pliku `myqueue.h`.

Odwołuje się do `MyList::pop_front()`.

Odwołania w `main()`.

```

00027         {
00028             return pop_front();
00029         }
  
```

5.6.2.2 void MyQueue::push (int arg) [inline]

Definicja w linii 23 pliku `myqueue.h`.

Odwołuje się do `MyList::push_back()`.

Odwołania w `main()`.

```

00023         {
00024             push_back(arg);
00025         }
  
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

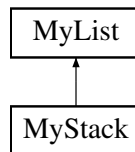
- `myqueue.h`

5.7 Dokumentacja klasy MyStack

Klasa reprezentuje stos.

```
#include <mystack.h>
```

Diagram dziedziczenia dla MyStack



Metody publiczne

- void `push` (int arg)
- int `pop` ()
Wyciąga element ze stosu.

5.7.1 Opis szczegółowy

Stos, którego index po pushu pokazuje na miejsce następne(następne za tym elementem)

Definicja w linii 18 pliku `mystack.h`.

5.7.2 Dokumentacja funkcji składowych

5.7.2.1 int MyStack::pop () [inline]

Definicja w linii 29 pliku `mystack.h`.

Odwołuje się do `MyList::pop_back()`.

```

00029         {
00030             return pop_back();
00031         }
  
```

5.7.2.2 void MyStack::push (int arg) [inline]

Definicja w linii 25 pliku `mystack.h`.

Odwołuje się do `MyList::push_back()`.

```

00025         {
00026             push_back(arg);
00027         }
  
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

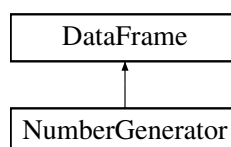
- `mystack.h`

5.8 Dokumentacja klasy NumberGenerator

Klasa generująca losowe liczby.

```
#include <numbergenerator.h>
```

Diagram dziedziczenia dla NumberGenerator



Metody publiczne

- void [generateNumbers](#) ()
Generuje losowe liczby.
- [~NumberGenerator](#) ()

Dodatkowe Dziedziczone Składowe

5.8.1 Opis szczegółowy

Klasa generująca losowe liczby na podstawie czasu maszyny na którym jest uruchomiona Wszystkie funkcje zapisu pliku dziedziczy z klasy [DataFrame](#)

Definicja w linii 23 pliku [numbergenerator.h](#).

5.8.2 Dokumentacja konstruktora i destruktor

5.8.2.1 [NumberGenerator::~NumberGenerator](#) () [inline]

Definicja w linii 44 pliku [numbergenerator.h](#).

```
00044 {}
```

5.8.3 Dokumentacja funkcji składowych

5.8.3.1 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(unsigned 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](#)

6 Dokumentacja plików

6.1 Dokumentacja pliku dataframe.cpp

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

6.2 dataframe.cpp

```

00001 /*
00002  * dataframe.cpp
00003  *
00004  * Created on: Mar 7, 2015
00005  * Author: serek8
00006  */
00007
00010 #include "dataframe.h"
00011
00012 DataFrame::DataFrame()
00013 {
00014     tableOfData = 0;
00015     outputFileName = NULL;
00016     inputFileName = NULL;
00017     sizeOfTable = 0;
00018 }
00019
00020 int DataFrame :: loadDataFromFile()
00021 {
00022     std::ifstream streamToFile;
00023     streamToFile.open (inputFileName, std::ifstream::in);
00024     this->tableOfData = new int[sizeOfTable];
00025     for(unsigned int i=0; i<sizeOfTable ; i++) {
00026         streamToFile >> this-> tableOfData[i];
00027         if (streamToFile.eof()) return 1; //[EoF reached]
00028     }
00029     return 0;
00030 }
00031
00032 int DataFrame :: saveDataToFile()
00033 {
00034     std::ofstream streamToFile;
00035     streamToFile.open (outputFileName, std::ofstream::out);
00036     for(unsigned int i=0; i<sizeOfTable ; i++) {
00037         streamToFile << this-> tableOfData[i] <<' ';
00038     }
00039     return 0;
00040 }
00041
00042
00043
00044 DataFrame DataFrame :: operator= (DataFrame dataframe)
00045 {
00046     this->tableOfData = dataframe.tableOfData;
00047     this->outputFileName = dataframe.outputFileName;
00048     this->inputFileName = dataframe.inputFileName;
00049     this->sizeOfTable = dataframe.sizeOfTable;
00050     return *this;
00051 }

```

6.3 Dokumentacja pliku dataframe.h

```
#include <fstream>
```

Komponenty

- class [DataFrame](#)

6.4 dataframe.h

```

00001 /*
00002  * dataframe.h
00003  *
00004  * Created on: Mar 6, 2015
00005  * Author: serek8
00006  */
00008 #ifndef DATAFRAME_H_
00009 #define DATAFRAME_H_
00010
00011 #include <fstream>
00012
00013
00014
00015 class DataFrame

```



```

00016 {
00017 public:
00021     int *tableOfData;
00025     char *outputFileName;
00029     char *inputFileName;
00034     unsigned int sizeOfTable;
00035
00039     DataFrame();
00049     int loadDataFromFile();
00050
00056     int saveDataToFile();
00057
00063     DataFrame operator= (DataFrame dataframe);
00064     virtual ~DataFrame() {}
00065
00066 };
00067 };
00068
00069
00070
00071 #endif /* DATAFRAME_H_ */

```

6.5 Dokumentacja pliku main.cpp

```

#include <iostream>
#include <unistd.h>
#include "multiplybytwo.h"
#include "numbergenerator.h"
#include "dataframe.h"
#include "mystack.h"
#include "myqueue.h"

```

Funkcje

- int `main` (int argc, char *argv[])

6.5.1 Dokumentacja funkcji

6.5.1.1 int main (int argc, char * argv[])

Ilość powtórzeń przez algorytmu

Zmienna używana przez GETOPT

Flaga która mówi o tym czy włączyć generator liczb losowych

Część testowa programu

Definicja w linii 17 pliku `main.cpp`.

Odwwołuje się do `MultiplyByTwo::executeAlgorithm()`, `NumberGenerator::generateNumbers()`, `DataFrame::inputFileName`, `DataFrame::loadDataFromFile()`, `DataFrame::outputFileName`, `MyQueue::pop()`, `MyQueue::push()`, `DataFrame::saveDataToFile()`, `DataFrame::sizeOfTable` i `MyBenchmark::testAlgorithm()`.

```

00018 {
00019     DataFrame podstawoweInfoIO;
00020     int quantityRepetitionOfAlgorithm = 0;
00021
00022     int opt;
00023     bool isSetNumberGenerator=false;
00024     bool isTest=false;
00025
00026     while ((opt = getopt(argc, argv, "n:t:o:i:gx")) != -1) {
00027         switch(opt){
00028             case 'n': // ilość liczb do przetworzenia
00029                 podstawoweInfoIO.sizeOfTable = atoi(optarg);
00030                 break;
00031
00032             case 't': // włącza benchmark i przyjmuje liczbę powtórzeń dla benchmarka

```

```

00033         quantityRepetitionOfAlgorithm = atoi(optarg);
00034         break;
00035
00036     case 'o':
00037         podstawoweInfoIO.outputFileName = optarg;
00038         break;
00039
00040     case 'i':
00041         podstawoweInfoIO.inputFileName=optarg;
00042         break;
00043
00044     case 'g': // wlacza generator liczb, po zakonczeniu generowania konczy program
00045         isSetNumberGenerator=true;
00046         break;
00047
00048     case 'x': // miejsce dla programisty dla sprawdzania kodu
00049         isTest =1;
00050         break;
00051
00052     case '?':
00053     default:
00054         std::cout<<"\nPodano zly argument";
00055         return -1;
00056     }
00057 }
00061 if(isTest)
00062 {
00063     MyQueue stack;
00064     stack.push(1); stack.push(2); stack.push(3);
00065     std::cout<<stack.pop();
00066     std::cout<<stack.pop();
00067     std::cout<<stack.pop();
00068     return 0;
00069 }
00070
00071
00072 /*
00073  * Sprawdzam czy program zostal uzyty tylko do wygenerowania liczb losowych
00074  * jesli tak to tworze te liczby zgodnie quantityNumber i zamykam program
00075  */
00076 if(isSetNumberGenerator) {
00077     NumberGenerator generator;
00078     generator= podstawoweInfoIO;
00079     generator.generateNumbers();
00080     generator.saveDataToFile();
00081     std::cout<<"\nZapisane.\n";
00082     return 0;
00083 }
00084
00085
00086 MultiplyByTwo algorytm_x2 ;
00087 algorytm_x2= podstawoweInfoIO;
00088
00089 /*
00090  * Wczytuje liczby z pliku do przeprowadzenia algorytmu
00091  */
00092 if(algorytm_x2.loadDataFromFile()) {
00093     std::cout<<"\nNie istnieje tyle liczb w pliku !\nKoncze program";
00094     return 1;
00095 }
00096
00097 /*
00098  * Sprawdzam czy otrzymalem agrument o testowaniu algorytmu,
00099  * a nastepnie przeprowadzam test albo uruchamiam normalnie algorytm
00100  */
00101 if(quantityRepetitionOfAlgorithm){
00102     std::cout<<"\nCzas algorytmu: "<<algorytm_x2.testAlgorithm(
quantityRepetitionOfAlgorithm)<<' \n';
00103 }
00104 else {
00105     algorytm_x2.executeAlgorithm();
00106 }
00107
00108 /*
00109  * Zapisuje wyniki do pliku
00110  */
00111 algorytm_x2.saveDataToFile();
00112
00113 return 0;
00114 }

```

6.6 main.cpp

```
00001 /*
```

```

00002  * main.cpp
00003  *
00004  * Created on: Mar 6, 2015
00005  * Author: serek8
00006  */
00008 #include <iostream>
00009 #include <unistd.h>
00010 #include "multiplybytwo.h"
00011 #include "numbergenerator.h"
00012 #include "dataframe.h"
00013 #include "mystack.h"
00014 #include "myqueue.h"
00015
00016
00017 int main(int argc, char *argv[])
00018 {
00019     DataFrame podstawoweInfoIO;
00020     int quantityRepetitionOfAlgorithm = 0;
00021
00022     int opt;
00023     bool isSetNumberGenerator=false;
00024     bool isTest=false;
00025
00026     while ((opt = getopt(argc, argv, "n:t:o:i:gx")) != -1) {
00027         switch(opt){
00028             case 'n': // ilosc liczb do przetworzenia
00029                 podstawoweInfoIO.sizeOfTable = atoi(optarg);
00030                 break;
00031
00032             case 't': // wlacza benchmark i przyjmuje liczbe powtorzen dla benchmarka
00033                 quantityRepetitionOfAlgorithm = atoi(optarg);
00034                 break;
00035
00036             case 'o':
00037                 podstawoweInfoIO.outputFileName = optarg;
00038                 break;
00039
00040             case 'i':
00041                 podstawoweInfoIO.inputFileName=optarg;
00042                 break;
00043
00044             case 'g': // wlacza generator liczb, po zakonczeniu generowania konczy program
00045                 isSetNumberGenerator=true;
00046                 break;
00047
00048             case 'x': // miejsce dla programisty dla sprawdzania kodu
00049                 isTest =1;
00050                 break;
00051
00052             case '?':
00053             default:
00054                 std::cout<<"\nPodano zly argument";
00055                 return -1;
00056         }
00057     }
00061     if(isTest)
00062     {
00063         MyQueue stack;
00064         stack.push(1); stack.push(2); stack.push(3);
00065         std::cout<<stack.pop();
00066         std::cout<<stack.pop();
00067         std::cout<<stack.pop();
00068         return 0;
00069     }
00070
00071
00072     /*
00073     * Sprawdzam czy program zostal uzyty tylko do wygenerowania liczb losowych
00074     * jesli tak to tworze te liczby zgodnie quantityNumber i zamykam program
00075     */
00076     if(isSetNumberGenerator) {
00077         NumberGenerator generator;
00078         generator= podstawoweInfoIO;
00079         generator.generateNumbers();
00080         generator.saveDataToFile();
00081         std::cout<<"\nZapisane.\n";
00082         return 0;
00083     }
00084
00085
00086     MultiplyByTwo algorytm_x2 ;
00087     algorytm_x2= podstawoweInfoIO;
00088
00089     /*
00090     * Wczytuje liczby z pliku do przeprowadzenia algorytmu
00091     */
00092     if(algorytm_x2.loadDataFromFile()) {

```

```

00093         std::cout<<"\nNie istnieje tyle liczb w pliku !\nKoncze program";
00094         return 1;
00095     }
00096
00097     /*
00098     * Sprawdzam czy otrzymałem argument o testowaniu algorytmu,
00099     * a następnie przeprowadzam test albo uruchamiam normalnie algorytm
00100     */
00101     if(quantityRepetitionOfAlgorithm){
00102         std::cout<<"\nCzas algorytmu: "<<algorytm_x2.testAlgorithm(
quantityRepetitionOfAlgorithm)<<"\n";
00103     }
00104     else {
00105         algorytm_x2.executeAlgorithm();
00106     }
00107
00108     /*
00109     * Zapisuje wyniki do pliku
00110     */
00111     algorytm_x2.saveDataToFile();
00112
00113     return 0;
00114 }

```

6.7 Dokumentacja pliku multiplybytwo.cpp

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

6.8 multiplybytwo.cpp

```

00001 /*
00002  * multiplybytwo.cpp
00003  *
00004  * Created on: Mar 7, 2015
00005  * Author: serek8
00006  */
00009 #include "multiplybytwo.h"
00010
00011 void MultiplyByTwo :: executeAlgorithm()
00012 {
00013     for(unsigned int i=0; i<sizeofTable; i++) {
00014
00015         tableOfData[i]*=2;
00016     }
00017
00018
00019
00020 }
00021
00022

```

6.9 Dokumentacja pliku multiplybytwo.h

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

Komponenty

- class `MultiplyByTwo`
Algorytm mnoży każdą liczbę razy 2.

6.10 multiplybytwo.h

```

00001 /*
00002  * multiplybytwo.h
00003  *
00004  * Created on: Mar 6, 2015

```

```

00005  *      Author: serek8
00006  */
00008  #ifndef MULTIPLYBYTWO_H_
00009  #define MULTIPLYBYTWO_H_
00010
00011  #include "mybenchmark.h"
00012  #include "dataframe.h"
00013
00014
00020  class MultiplyByTwo : public MyBenchmark
00021  {
00022  public:
00027      void executeAlgorithm();
00028
00029      ~MultiplyByTwo() {}
00030
00031
00032
00033      using DataFrame::operator=;
00034
00035  };
00036
00037  #endif /* MULTIPLYBYTWO_H_ */

```

6.11 Dokumentacja pliku mybenchmark.cpp

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

6.12 mybenchmark.cpp

```

00001  /*
00002  * mybenchmark.cpp
00003  *
00004  * Created on: Mar 6, 2015
00005  * Author: serek8
00006  */
00009  #include "mybenchmark.h"
00010
00011
00012  double MyBenchmark::testAlgorithm(unsigned int repetition)
00013  {
00014      time_t benchmarkTimeInTotal = 0;
00015      time_t benchmarkTimeForOneLoop = 0;
00016
00017      int *originalTableOfData = new int[sizeOfTable];
00018      for(unsigned int i=0; i<sizeOfTable; i++)
00019          originalTableOfData[i]=tableOfData[i];
00020
00021      for(unsigned int i=0 ; i<repetition ; i++) {
00022          for(unsigned int i=0; i<sizeOfTable; i++)
00023              tableOfData[i]=originalTableOfData[i];
00024
00025          benchmarkTimeForOneLoop = clock();
00026          this->executeAlgorithm();
00027          benchmarkTimeInTotal += clock() - benchmarkTimeForOneLoop;
00028      }
00029
00030
00031      return (( (double)benchmarkTimeInTotal ) /CLOCKS_PER_SEC);
00032  }
00033
00034

```

6.13 Dokumentacja pliku mybenchmark.h

```
#include <ctime>
#include "dataframe.h"
```

Komponenty

- class `MyBenchmark`

Klasa bazowa/interface do testowania algorytmu.

6.14 mybenchmark.h

```

00001 /*
00002  * mybenchmark.h
00003  *
00004  * Created on: Mar 6, 2015
00005  * Author: serek8
00006  */
00007 #ifndef MYBENCHMARK_H_
00008 #define MYBENCHMARK_H_
00009
00010 #include <ctime>
00011 #include "dataframe.h"
00012 class MyBenchmark : public DataFrame
00013 {
00014 protected:
00015     virtual void executeAlgorithm() = 0;
00016
00017 public:
00018     double testAlgorithm(unsigned int repetition);
00019     virtual ~MyBenchmark() {};
00020     //using DataFrame::operator=;
00021 };
00022 #endif /* MYBENCHMARK_H_ */

```

6.15 Dokumentacja pliku mylist.cpp

```
#include "mylist.h"
```

6.16 mylist.cpp

```

00001 /*
00002  * mylist.cpp
00003  *
00004  * Created on: Mar 15, 2015
00005  * Author: serek8
00006  */
00007
00008 #include "mylist.h"
00009
00010 MyList::MyList()
00011 {
00012     firstElement = lastElement = new MyListElement(0);
00013     sizeOfList = 0;
00014 }
00015
00016 void MyList::push_back(int arg)
00017 {
00018     MyListElement *newMyListElement = new MyListElement(arg);
00019     if(!sizeOfList++) {firstElement = lastElement = newMyListElement;}
00020     //newMyListElement -> nextElement = 0;
00021     newMyListElement -> previousElement = this -> lastElement;
00022     this -> lastElement -> nextElement = newMyListElement;
00023     this->lastElement = newMyListElement;
00024 }
00025 void MyList::push_front(int arg)
00026 {
00027     MyListElement *newMyListElement = new MyListElement(arg);
00028     if(!sizeOfList++) {firstElement = lastElement = newMyListElement;}
00029     //newMyListElement -> previousElement = 0;
00030     newMyListElement -> nextElement = this -> firstElement;
00031     this -> firstElement -> previousElement = newMyListElement;
00032     this->firstElement = newMyListElement;
00033 }
00034 int MyList::pop_back()
00035 {

```

```

00039         if(!(sizeOfList--)) { sizeOfList=0; return 0; }
00040         int tmpNumber = this -> lastElement -> number;
00041         MyListElement *originMyListElement = this -> lastElement;
00042         this -> lastElement = this -> lastElement -> previousElement;
00043         delete originMyListElement;
00044         return tmpNumber;
00045     }
00046     int MyList :: pop_front()
00047     {
00048         if(!(sizeOfList--)) { sizeOfList=0; return 0; }
00049         int tmpNumber = this -> firstElement -> number;
00050         MyListElement *originMyListElement = this -> firstElement;
00051         this -> firstElement = this -> firstElement -> nextElement;
00052         delete originMyListElement;
00053         return tmpNumber;
00054     }
00055 }
00056
00057 MyList :: MyListElement :: MyListElement(int arg)
00058 {
00059     this -> number = arg;
00060     this -> nextElement =0;
00061     this -> previousElement =0;
00062 }
00063
00064
00065
00066
00067
00068
00069
00070 int MyList::size()
00071 {
00072     return sizeOfList;
00073 }

```

6.17 Dokumentacja pliku mylist.h

```
#include <iostream>
```

Komponenty

- class `MyList`

Lista dwukierunkowa.

- class `MyList::MyListElement`

Klasa 'małych struktur' gdzie jest numer i wskaźnik do następnego elementu.

6.18 mylist.h

```

00001 /*
00002  * mylist.h
00003  *
00004  * Created on: Mar 12, 2015
00005  * Author: serek8
00006  */
00007
00008 #ifndef MYLIST_H_
00009 #define MYLIST_H_
00010
00011 #include <iostream>
00012
00013 class MyList{
00014 private:
00015     int sizeOfList;
00016
00017     class MyListElement {
00018     public:
00019         int number;
00020
00021         MyListElement(int arg);
00022
00023         MyListElement *nextElement;
00024         MyListElement *previousElement;
00025     };
00026 }

```

```
00040     };
00041
00043     MyListElement *firstElement;
00045     MyListElement *lastElement;
00047 public:
00048     MyList();
00049
00054     int size();
00059     int pop_back();
00064     int pop_front();
00068     void push_back(int arg);
00072     void push_front(int arg);
00073
00074 };
00075
00076
00077
00078 #endif /* MYLIST_H_ */
```

6.19 Dokumentacja pliku myqueue.h

```
#include "mylist.h"
```

Komponenty

- class `MyQueue`

Klasa reprezentuje kolejke.

6.20 myqueue.h

```
00001 /*
00002  * myqueue.h
00003  *
00004  * Created on: Mar 16, 2015
00005  * Author: serek8
00006  */
00007
00008 #ifndef MYQUEUE_H_
00009 #define MYQUEUE_H_
00010 #include "mylist.h"
00011
00016 class MyQueue : public MyList
00017 {
00018 public:
00019     /*
00020      * @brief Dodaje element do kolejki
00021      * @param arg Liczba dodawana do kolejki
00022      */
00023     void push(int arg) {
00024         push_back(arg);
00025     }
00027     int pop() {
00028         return pop_front();
00029     }
00030 };
00031
00032 #endif /* MYQUEUE_H_ */
```

6.21 Dokumentacja pliku mystack.h

```
#include "mylist.h"
```

Komponenty

- class `MyStack`

Klasa reprezentuje stos.

6.22 mystack.h

```

00001 /*
00002  * mystack.h
00003  *
00004  * Created on: Mar 16, 2015
00005  * Author: serek8
00006  */
00007
00008 #ifndef MYSTACK_H_
00009 #define MYSTACK_H_
00010
00011 #include "mylist.h"
00012
00018 class MyStack : public MyList
00019 {
00020 public:
00021     /*
00022      * @brief Dodaje element do kolejki
00023      * @param arg Liczba dodawana do stosu
00024      */
00025     void push(int arg) {
00026         push_back(arg);
00027     }
00029     int pop() {
00030         return pop_back();
00031     }
00032 };
00033
00034 #endif /* MYSTACK_H_ */

```

6.23 Dokumentacja pliku numbergenerator.h

```

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

```

Komponenty

- class `NumberGenerator`
Klasa generująca losowe liczby.

6.24 numbergenerator.h

```

00001 /*
00002  * numbergenerator.h
00003  *
00004  * Created on: Mar 11, 2015
00005  * Author: serek8
00006  */
00007
00008 #ifndef NUMBERGENERATOR_H_
00009 #define NUMBERGENERATOR_H_
00010
00011 #include <stdlib.h>      /* srand, rand */
00012 #include <time.h>        /* time */
00013 #include <iostream>
00014 #include "dataframe.h"
00015
00023 class NumberGenerator : public DataFrame
00024 {
00025 public:
00031 void generateNumbers()
00032 {
00033     time_t randomTime = clock();
00034     this->tableOfData = new int[sizeOfTable];
00035     for(unsigned int i=0; i<sizeOfTable ; i++)
00036     {
00037         srand (randomTime = clock());
00038         this->tableOfData[i] = rand()%100;
00039         randomTime = clock();
00040     }
00041 }

```

```
00042
00043 using DataFrame::operator=;
00044 ~NumberGenerator() {}
00045 };
00046
00047 #endif /* NUMBERGENERATOR_H_ */
```

6.25 Dokumentacja pliku strona-glowna.dox