Lab 1

1.1

Generated by Doxygen 1.8.9.1

Thu Mar 12 2015 03:33:33

Contents

Chapter 1

Todo List

Member Benchmark::generateRaport (long double nextTime, int size)
FILE OVERWRITING, NEED TO IMPLEMENT NEW NAMES

Member InputFiles::InputFiles ()
EXCEPTIONS HANDLING

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Benchmark Class Reference

```
#include <benchmark_frm.h>
```

Public Member Functions

- void test (InputFiles files)

    Main testing function.

Private Member Functions

- void generateRaport (long double nextTime, int size)

    Generates raport with program tests outputs.
- long double getAvr (std::vector< long double >times)

    Measures the average duration from 10 samples.
- void measureTime (int ∗dataTable, int dataSize)

    Measures the duration of the work of assignment function.

Private Attributes

- std::vector< long double > testTimes

    A container for calculated times.

4.1.1 Detailed Description

Making a framework for testing inserted data structure. Using time to estimate computational complexity.

4.1.2 Member Function Documentation

4.1.2.1 Benchmark::generateRaport (long double *nextTime*, int *size*) `[private]`

Generates raport with program tests outputs.

Parameters

nextTime	A new calculated time (for new file size).
Size	A size of the currently working file.

Todo FILE OVERWRITING, NEED TO IMPLEMENT NEW NAMES

```
                                                             {
    /*! \todo FILE OVERWRITING, NEED TO IMPLEMENT NEW NAMES */
    std::ofstream raportFile;
    std::string stringNextTime = std::to_string(nextTime);

     // .xls as excel file format
     raportFile.open("test.xls", std::ios::in | std::ios::app);
         assert(raportFile.is_open() && ("I can't open file."));
     // need to change '.' on ',' due to excel string format
     boost::algorithm::replace_first(stringNextTime, ".", ",");

     raportFile << size << "\n" << stringNextTime << "\n";
     raportFile.close();
}
```

4.1.2.2 Benchmark::getAvr (std::vector< long double > *times*) [private]

Measures the average duration from 10 samples.

Parameters

times	A container with times from tests.

```
                                                          {
     long double avrg = 0.0;

     //add 10 values, than count average
     for (int i = 0; i < (signed)times.size(); i++){
         avrg += times[i];
     }

     avrg /= (long double)times.size();
     return avrg;
}
```

4.1.2.3 Benchmark::measureTime (int ∗ *dataTable,* int *dataSize*) [private]

Measures the duration of the work of assignment function.

Parameters

dataTable	A container with random integers from earlier made files.
dataSize	A size of the file.

```
                                                         {
     // container for counted working times
     std::vector<long double> estimateTimes;

     for (int j = 0; j < 10; j++){
         // Here starts the timer
         boost::timer::cpu_timer startTime;
         for (int i = 0; i < dataSize; i++){
             dataTable[i] *= 2;
         }
         // Here it ends
         boost::timer::cpu_times endTime = startTime.elapsed();
         // add new time to the vector
         estimateTimes.push_back(static_cast<long double>(endTime.wall * SEC));
     }
     // for better display
     std::cout.fixed;
     long double DurTime = getAvr(estimateTimes);
     std::cout << "Time (average, 10 samples) for " << dataSize << " elements: " << DurTime << " sec"<<
    std::endl;
     generateRaport(DurTime, dataSize);
}
```

4.1.2.4 Benchmark::test (InputFiles *files*)

Main testing function.

Parameters

files	random generated files with integers

```
                                    {
    // temp memory container
    int* tabForData = NULL;
    int tempValue = 0;
    int count = 0;
    std::fstream newFile;

    for (int i = 0; i < files.return_number_files() -
   FIRST_ARGUMENT; i++){
        // Opening file + making new table with content
        tabForData = new int[files.return_file_size(i)];
        newFile.open((files.return_file_name(i) + ".txt"), std::ios::in);

        // Checking if file is opened correctly
        assert(newFile.is_open() && ("I can't open file."));

        for (int j = 0; j < files.return_file_size(i); j++){
            newFile >> tempValue;
            tabForData[j] = tempValue;
        }
        newFile.close();

        // Testing time here
        measureTime(tabForData, files.return_file_size(i));
        delete[] tabForData;
    }
}
```

4.1.3 Member Data Documentation

4.1.3.1 Benchmark::testTimes [private]

A container for calculated times.

The documentation for this class was generated from the following files:

- src/benchmark_frm.h
- src/benchmark_frm.cpp

4.2 InputFiles Class Reference

```
#include <inputfile_txt.h>
```

Public Member Functions

- void generate_random_int_data ()

    Puts random int data into files.
- InputFiles ()

    A default constructor.
- InputFiles (int filNr, std::vector< int >filSiz)

    A constructor.
- const std::string return_file_name (int Nmbr)

    Return names of files (only for read purpose)
- const int return_file_size (int Nmbr)

    Return sizes of files (only for read purpose)

- const int return_number_files ()

    Return number of files.
- void show_info ()

    Showes info obout files.

Private Attributes

- std::vector< std::string > filesNamesTab

    Container for generated file names.
- int filesNumber

    Number of generated files.
- std::vector< int > filesSizes

    Container for file sizes.

4.2.1 Detailed Description

Making an object which contain text files with generated random integer numbers.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 InputFiles::InputFiles ( )

A default constructor.

Adding number of files(UNDEF_VALUE = 1); Generating file name; Adding size of file (UNDEF_VALUE = 1);

Just in case, when program starts without any parameters.

Todo EXCEPTIONS HANDLING

```
                        {
    /*! \todo EXCEPTIONS HANDLING */
    // When there are no arguments from command prompt:
    filesNumber = UNDEF_VALUE;
     std::string TempName = std::tmpnam(nullptr);
     filesNamesTab.push_back(TempName);
     filesSizes.push_back(UNDEF_VALUE);
}
```

4.2.2.2 InputFiles::InputFiles (int *filNr,* std::vector< int > *filSiz*)

A constructor.

Adding number of files; Generating files names; Adding sizes of files; Parameters inherit from list of arguments from command prompt

Parameters

filNr	number of files
filSiz	sizes of files

Open files with new names

Check if file is opened correctly

```
                                                               {
     filesNumber = filNr;
     filesSizes = filSiz;
     // Create new names for files
     std::string TempName;
```

```

    for (int i = 1; i < filesNumber; i++){
        // Generate new unique name for file
        TempName = std::tmpnam(nullptr);
        // Delete all prohibit char from string
        boost::algorithm::erase_all(TempName, "/");
        boost::algorithm::erase_all(TempName, "\\");
        // Put name into names container
        filesNamesTab.push_back(TempName);
    }
    //! Open files with new names
    std::ofstream newFile;
    for (int i = 1; i < filesNumber; i++){
        newFile.open(filesNamesTab[i - PROGRAM_NAME] + ".txt");
            //! Check if file is opened correctly
            assert(newFile.is_open() && "I can't open this file.");
        newFile.close();
    }
}
```

4.2.3 Member Function Documentation

4.2.3.1 InputFiles::generate_random_int_data ( )

Puts random int data into files.

Generating random integers data (size from filesSizes vector) and putting them into files (names from filesNames↩ Tab) < Seed for Mersenne Twister 19937 generator

Mersenne Twister 19937 generator

More info about this generator: http://pl.wikipedia.org/wiki/Mersenne_Twister

Uniform distribution random number

Max number: uncomment next line More info about this distribution: http://pl.wikipedia.org/wiki/↩ Rozk%C5%82ad_jednostajny

Check if file is opened correctly

```
                                          {
    int seedGen = time(NULL); /*!< Seed for Mersenne Twister 19937 generator */

    //! Mersenne Twister 19937 generator
    /*!
        More info about this generator:
        <a href="linkURL">http://pl.wikipedia.org/wiki/Mersenne_Twister</a>
    */
    std::mt19937 randomNumbr(seedGen);

    //! Uniform distribution random number
    /*!
        Max number: uncomment next line
        More info about this distribution:
        <a href="linkURL">http://pl.wikipedia.org/wiki/Rozk%C5%82ad_jednostajny</a>
    */
    //std::cout << std::numeric_limits<int>::max() << std::endl;
    std::uniform_int_distribution<>newDistr;

    std::ofstream NewFile;
    for (int i = 1; i < filesNumber; i++){
        NewFile.open((filesNamesTab[i - PROGRAM_NAME] + ".txt"), std::ios::in);
            //! Check if file is opened correctly
            assert(NewFile.is_open() && ("I can't open file."));
        //Generate random int data
        for (int j = 0; j < filesSizes[i - FIRST_ARGUMENT]; j++){
            NewFile << newDistr(randomNumbr) << "\n";
        }

        NewFile.close();
    }
}
```

4.2.3.2 InputFiles::return_file_name (int *Nmbr*) [inline]

Return names of files (only for read purpose)

Parameters

Nmbr	Number of the file.

```
{
        return filesNamesTab[Nmbr];
    }
```

4.2.3.3 InputFiles::return_file_size (int *Nmbr*) [inline]

Return sizes of files (only for read purpose)

Parameters

Nmbr	Number of the file.

```
{
        return filesSizes[Nmbr];
    }
```

4.2.3.4 InputFiles::return_number_files ( ) [inline]

Return number of files.

```
{
        return filesNumber;
    }
```

4.2.3.5 InputFiles::show_info ( )

Showes info obout files.

Display: number of files, names of files, sizes of files

```
{
    std::cout << "--------" << std::endl;
    std::cout << filesNumber - FIRST_ARGUMENT << std::endl;
    for (int i = 0; i < (signed)filesNamesTab.size(); i++){
        std::cout << filesNamesTab[i] << std::endl;
    }
    for (int i = 0; i < (signed)filesSizes.size(); i++){
        std::cout << filesSizes[i] << std::endl;
    }
    std::cout << "--------" << std::endl;
}
```

4.2.4 Member Data Documentation

4.2.4.1 std::vector<std::string> InputFiles::filesNamesTab [private]

Container for generated file names.

4.2.4.2 InputFiles::filesNumber [private]

Number of generated files.

4.2.4.3 InputFiles::filesSizes [private]

Container for file sizes.

The documentation for this class was generated from the following files:

- src/inputfile_txt.h
- src/inputfile_txt.cpp

Chapter 5

File Documentation

5.1 src/benchmark_frm.cpp File Reference

Source code for Benchmark class.

```
#include "benchmark_frm.h"
```

5.1.1 Detailed Description

Source code for Benchmark class.

5.2 src/benchmark_frm.h File Reference

A Benchmark class.

```
#include <vector>
#include <fstream>
#include <boost\timer\timer.hpp>
#include <boost\chrono\duration.hpp>
#include <boost\algorithm\string\replace.hpp>
#include "inputfile_txt.h"
```

Classes

- class Benchmark

Variables

- const long double SEC = 0.000000001

5.2.1 Detailed Description

A Benchmark class.

5.2.2 Variable Documentation

5.2.2.1 const long double SEC = 0.000000001

5.3 src/inputfile_txt.cpp File Reference

Source code for InputFile class.

```
#include "inputfile_txt.h"
```

5.3.1 Detailed Description

Source code for InputFile class.

5.4 src/inputfile_txt.h File Reference

A InputFile class.

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <cstdio>
#include <cassert>
#include <ctime>
#include <random>
#include <boost/algorithm/string/erase.hpp>
```

Classes

- class InputFiles

Variables

- const int FIRST_ARGUMENT = 1

  A const value for representing first argument from command prompt (name of the program) ∗/.
- const int PROGRAM_NAME = 1

  The same as FIRST_ARGUMENT ∗/.
- const int UNDEF_VALUE = 1

  A value for undefined arguments ∗/.

5.4.1 Detailed Description

A InputFile class.

5.4.2 Variable Documentation

5.4.2.1 const int FIRST_ARGUMENT = 1

A const value for representing first argument from command prompt (name of the program) ∗/.

5.4.2.2 const int PROGRAM_NAME = 1

The same as FIRST_ARGUMENT ∗/.

5.4.2.3 const int UNDEF_VALUE = 1

A value for undefined arguments ∗/.

5.5 src/main.cpp File Reference

Master file.

```
#include "inputfile_txt.h"
#include "benchmark_frm.h"
```

Functions

- int main (int argc, char ∗argv[ ])

5.5.1 Detailed Description

Master file.

5.5.2 Function Documentation

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

```
{
    // Container for sizes from command prompt
    std::vector<int>FilesSizes;

    // First argument is a name of the program so i = 1
    for (int i = 1; i < argc ; i++)
        FilesSizes.push_back(atoi(argv[i]));

    InputFiles newFilesList(argc, FilesSizes);
    Benchmark NewTest;

    newFilesList.generate_random_int_data();
    NewTest.test(newFilesList);
}
```

Index