МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра «Математического обеспечения и применения ЭВМ»

|  |  |  |
| --- | --- | --- |
| 1002262_584459121576624_356098510_n |  | logo МОПЭВМ |

Отчет

По лабораторной работе №1

««Исследование паттернов проектирования, систем

контроля версий и систем сборки проектов на примере Git и

CMake»

по курсу

«Архитектура и проектирование программных систем»

Выполнили: студенты группы Ктбо2-8

Гуляев Е.А.

Гуляев М.А.

Проверил:

Шкурко А.Н.

Оценка:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2016 г.

Таганрог 2016

[Отчет 1](#_Toc447889810)

[1. Цель 4](#_Toc447889811)

[2. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ 4](#_Toc447889812)

[3. Описание паттернов проектирования 4](#_Toc447889813)

[3.1. Фасад 4](#_Toc447889814)

[3.1.1. Название и классификация паттерна 4](#_Toc447889815)

[3.1.2. Назначение 4](#_Toc447889816)

[3.1.3. Мотивация 4](#_Toc447889817)

[3.1.4. Участники 4](#_Toc447889818)

[3.1.5. Отношения 4](#_Toc447889819)

[3.1.6. Результаты 4](#_Toc447889820)

[3.1.7. Реализация 4](#_Toc447889821)

[3.1.8. Известные применения. 5](#_Toc447889822)

[3.1.9. Родственные паттерны 5](#_Toc447889823)

[3.2. Фабрика 5](#_Toc447889824)

[3.2.1. Название и классификация паттерна 5](#_Toc447889825)

[3.2.2. Назначение 6](#_Toc447889826)

[3.2.3. Мотивация 6](#_Toc447889827)

[3.2.4. Участники 6](#_Toc447889828)

[3.2.5. Отношения 6](#_Toc447889829)

[3.2.6. Результаты 6](#_Toc447889830)

[3.2.7. Реализация 6](#_Toc447889831)

[3.2.8. Известные применения 6](#_Toc447889832)

[3.2.9. Родственные паттерны 7](#_Toc447889833)

[3.3. Диаграмма 7](#_Toc447889834)

[4. История изменений по проекту 7](#_Toc447889835)

[5. Структура проекта и файлы CMakeLists.txt 9](#_Toc447889836)

[5.1. Простая сборка проекта 9](#_Toc447889837)

[5.2. Статическая библиотека 10](#_Toc447889838)

[5.3. Работа CMake 10](#_Toc447889839)

[5.4. Подключаем библиотеку через FIND() 11](#_Toc447889840)

[6. Исходный код программы 11](#_Toc447889841)

# Цель

Цель лабораторной работы - освоение системы контроля версии Git и сборки проекта с помощью CMake.

# ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Для выполнения лабораторной работы необходимо разбиться бригады по

2 человека. Варианты задания:

1.Фасад и Фабричный метод

Лабораторная работа делится на две части: описание паттернов и их

реализация на языке C/C++ с использованием системы сборки проектов CMake

и любой из систем контроля версий (SVN или Mercurial).

# Описание паттернов проектирования

## Фасад

### Название и классификация паттерна

Фасад - [структурный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D1%8B%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).

### Назначение

Позволяющий скрыть сложность системы путем сведения всех возможных внешних вызовов к одному [объекту](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), делегирующему их соответствующим объектам системы.

### Мотивация

Определить одну точку взаимодействия с подсистемой — фасадный объект, обеспечивающий общий интерфейс с подсистемой, и возложить на него обязанность по взаимодействию с её компонентами. Фасад — это внешний объект, обеспечивающий единственную точку входа для служб подсистемы.

### Участники

На диаграмме.

### Отношения

На диаграмме.

### Результаты

/-/

### Реализация

#pragma once

#include <sstream>

#include <string>

#include "Series.h"

#include "SortingWrapper.h"

class Application

{

public:

Application();

~Application();

void RunApp();

private:

static const int ADD\_VALUE = 1;

static const int SORTING\_SERIES = 2;

static const int SHOW\_SERIES = 3;

static const int FILLING\_SERIES = 4;

static const int FILE\_INPUT = 5;

static const int FILE\_OUTPUT = 6;

static const int ERROR = -404;//- так как время может быть положительным ,если оно совпадет ,то будет печаль

template < class T >

bool Read(T& res);

void PrintMenu();

void ShowInputError();

void AddValueMenu(Series &series);

void ShowSeriesMenu(Series &series)const;

void SortingSeriesMenu(Series &series);

void FillingSeriesMenu(Series &series);

void FileInputMenu(Series &series);

void FileOutputMenu(Series &series);

};

### Известные применения.

В операционной системе [Choices](http://www.google.com/search?q=os%20choices) фасады используются для составления одного каркаса из нескольких. Ключевыми абстракциями в системе [Choices](http://www.google.com/search?q=os%20choices) являются процессы, память и адресные пространства. Для каждой из них есть соответствующая подсистема, реализованная в виде каркаса. Это обеспечивает поддержку переноса [Choices](http://www.google.com/search?q=os%20choices) на разные аппаратные платформы. У двух таких подсистем есть «представители», то есть фасады. Они называются FileSystemlnterface (память) и Domain (адресные пространства).

### Родственные паттерны

Паттерн [абстрактная фабрика](http://codelab.ru/p/abstract_factory/) допустимо использовать вместе с фасадом, чтобы предоставить интерфейс для создания объектов подсистем способом, не зависимым от этих подсистем.

## Фабричный метод

### Название и классификация паттерна

Фабричный метод - [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F),

### Назначение

Предоставляет подклассам интерфейс для создания экземпляров некоторого класса. В момент создания наследники могут определить, какой класс создавать.

### Мотивация

В нашем проекте фабричный метод возвращает нужный нам указатель на объект. Т.е в зависимости от селектора (номера сортировки) фабричный метод возвращает либо сортировка методом выбора, сортировка вставками , сортировка QuickSort(рекурсивный)-свой, библиотечный QuickSort ,сортировка Шелла.

### Участники

На диаграмме.

### Отношения

На диаграмме.

### Результаты

Плюсы и минусы:

+

позволяет сделать код создания объектов более универсальным, не привязываясь к конкретным классам, а оперируя лишь общим интерфейсом

-

необходимость создавать наследника Creator для каждого нового типа продукта.(Если классы не являются наследником от общего класса)

### Реализация

class SortingFactory

{

public:

SortingFactory()=default;

~SortingFactory() = default;

ISorting\* FabricMethod(int selector);

private:

static const int CHOISE = 1;

static const int QUICK = 2;

static const int MYQUICK = 3;

static const int INPUT = 4;

static const int SHELL = 5;

};

ISorting\* SortingFactory::FabricMethod(int selector){

switch (selector)

{

case CHOISE:

return new ChoiseSort();

break;

case QUICK:

return new QuickSort();

break;

case MYQUICK:

return new MyQuickSort();

break;

case INPUT:

return new InputSort();

break;

case SHELL:

return new ShellSort();

break;

default:

return nullptr;

}

}

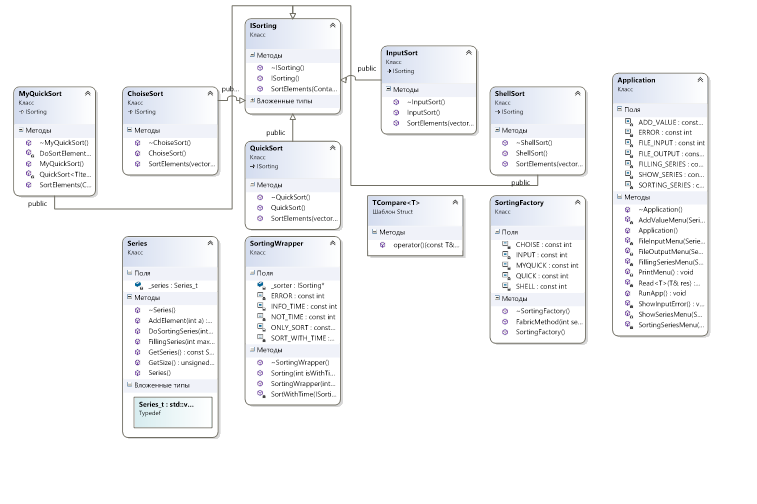
### Известные применения

Фабричные методы где только не встречаются - **большинство библиотек и каркасов так или иначе используют паттерн Фабричный метод** - в частности библиотека ЕТ++[WGM88].

### Родственные паттерны

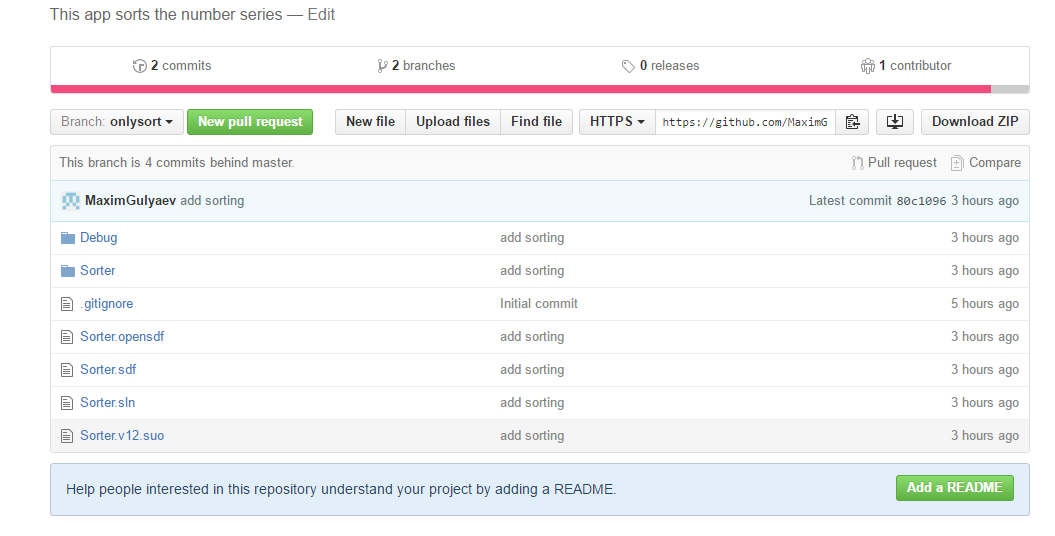
Фабричный метод является как бы основой для «фабрики», «строителя» и «прототипа». В разработке часто именно так и получается, сперва реализуют фабричный метод, а по мере усложнения кода выбирают во что именно его преобразовать, в какой из перечисленных паттернов. При использовании «фабричного метода» каждый объект как бы сам является «фабрикой».

## Диаграмма

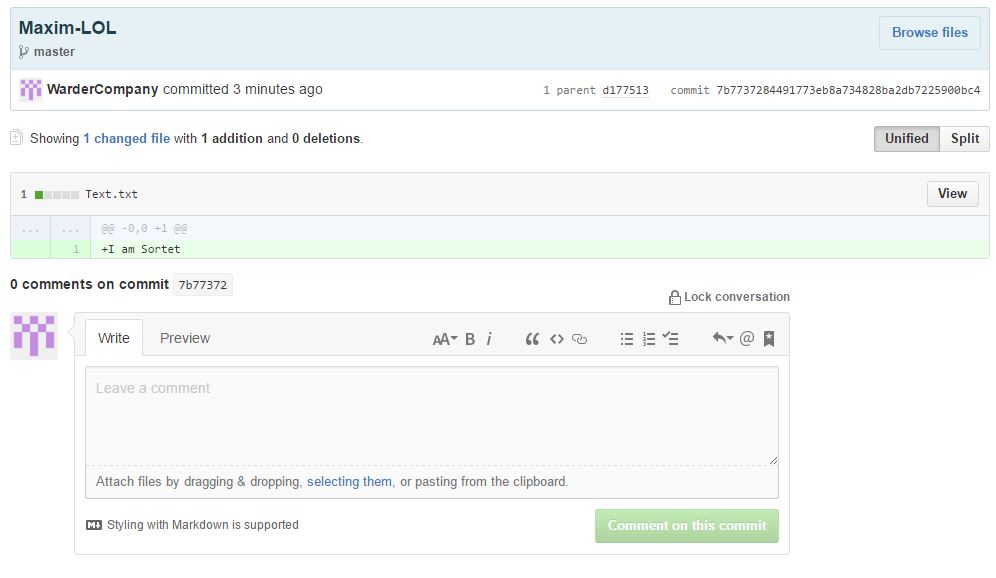


# История изменений по проекту

**1)git checkout -b onlysort  
 git push origin onlysort**

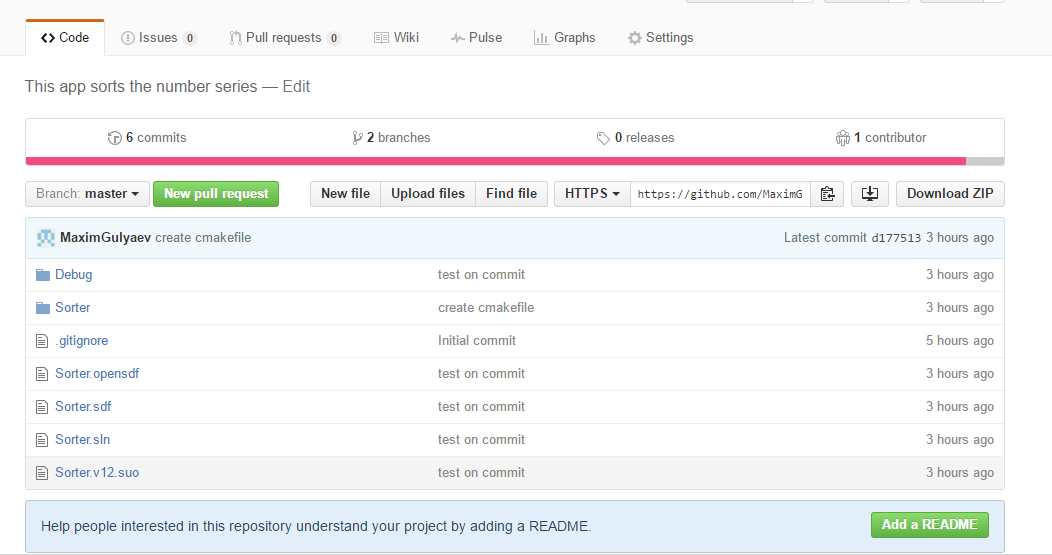


**2)git push origin master**

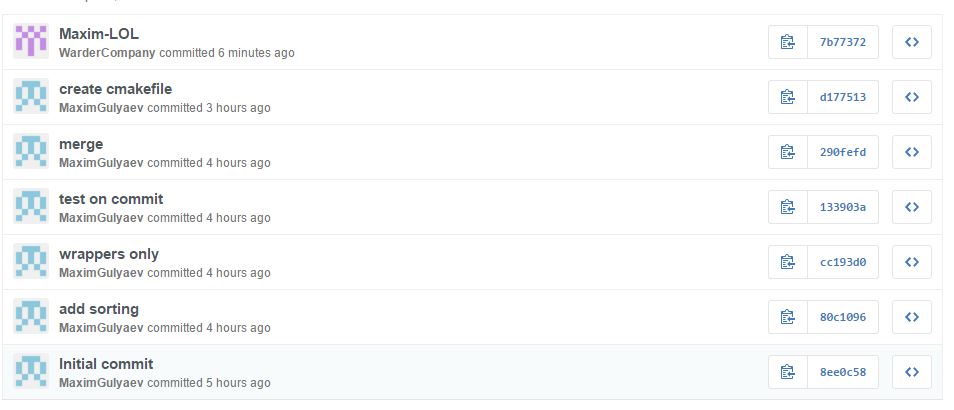


**3)git checkout master**

**git merge onlysort**



4) **История коммитов**

****

# Структура проекта и файлы CMakeLists.txt

## Простая сборка проекта

cmake\_minimum\_required(VERSION 3.3)

project(Sorter)

set(SOURCES Source.cpp

Application.cpp

ChoiseSort.cpp

InputSort.cpp

ISorting.cpp

MyQuickSort.cpp

QuickSort.cpp

Series.cpp

ShellSort.cpp

SortingFactory.cpp

SortingWrapper.cpp

)

set(HEADERS HelpStruct.h

Application.h

ChoiseSort.h

InputSort.h

ISorting.h

MyQuickSort.h

QuickSort.h

Series.h

ShellSort.h

SortingFactory.h

SortingWrapper.h

)

add\_executable(SorteR ${SOURCES} ${HEADERS})

target\_link\_libraries(SorteR)

## Статическая библиотека

cmake\_minimum\_required(VERSION 3.3)

project(Sorter)

set(SOURCES Source.cpp

Application.cpp

ChoiseSort.cpp

InputSort.cpp

ISorting.cpp

MyQuickSort.cpp

QuickSort.cpp

Series.cpp

ShellSort.cpp

SortingFactory.cpp

SortingWrapper.cpp

)

set(HEADERS HelpStruct.h

Application.h

ChoiseSort.h

InputSort.h

ISorting.h

MyQuickSort.h

QuickSort.h

Series.h

ShellSort.h

SortingFactory.h

SortingWrapper.h

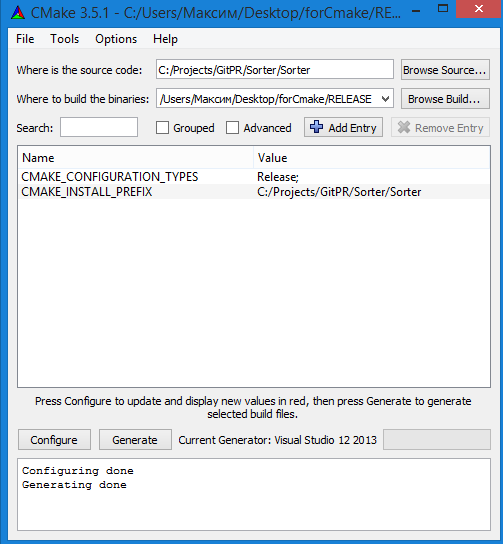
)

find\_package(Boost)

add\_executable(SorteR ${SOURCES} ${HEADERS})

target\_link\_libraries(SorteR)

## Работа CMake



# Исходный код программы

#pragma once

#include <sstream>

#include <string>

#include "Series.h"

#include "SortingWrapper.h"

class Application

{

public:

Application();

~Application();

void RunApp();

private:

static const int ADD\_VALUE = 1;

static const int SORTING\_SERIES = 2;

static const int SHOW\_SERIES = 3;

static const int FILLING\_SERIES = 4;

static const int FILE\_INPUT = 5;

static const int FILE\_OUTPUT = 6;

static const int ERROR = -404;//- так как время может быть положительным ,если оно совпадет ,то будет печаль

template < class T >

bool Read(T& res);

void PrintMenu();

void ShowInputError();

void AddValueMenu(Series &series);

void ShowSeriesMenu(Series &series)const;

void SortingSeriesMenu(Series &series);

void FillingSeriesMenu(Series &series);

void FileInputMenu(Series &series);

void FileOutputMenu(Series &series);

};

#include "Application.h"

Application::Application() = default;

Application::~Application() = default;

void Application::RunApp(){

int action = 0;

Series series;

do{

PrintMenu();

if (!Read(action))

{

ShowInputError();

continue;

}

switch (action)//Переключатель Меню

{

case ADD\_VALUE:

{

AddValueMenu(series);

break;

}

case SORTING\_SERIES:

{

SortingSeriesMenu(series);

break;

}

case SHOW\_SERIES:

{

ShowSeriesMenu(series);

break;

}

case FILLING\_SERIES:

{

FillingSeriesMenu(series);

break;

}

case FILE\_INPUT:

{

FileInputMenu(series);

break;

}

case FILE\_OUTPUT:

{

FileOutputMenu(series);

break;

}

default:

ShowInputError();

}

} while (true);

}

template < class T >

bool Application::Read(T& res){

std::string s;

std::cin >> s;

std::istringstream sin(s);

if (!(sin >> res))

{

return false;

}

if (!sin.eof()) return false;

return true;

}

void Application::PrintMenu(){

std::cout << "Menu" << std::endl;

std::cout << ADD\_VALUE << ". Add value in series" << std::endl;

std::cout << SORTING\_SERIES << ". Sorting series" << std::endl;

std::cout << SHOW\_SERIES << ". Show series" << std::endl;

std::cout << FILLING\_SERIES << ". Add random series" << std::endl;

std::cout << FILE\_INPUT << ". File input" << std::endl;

std::cout << FILE\_OUTPUT << ". File output" << std::endl;

}

void Application::ShowInputError()

{

std::cout << "Error: incorrect input" << std::endl;

}

void Application::AddValueMenu(Series &series){

int value;

std::cout << "Input element" << std::endl;

if (!Read(value))

{

ShowInputError();

return;

}

series.AddElement(value);

std::cout << "Element is added" << "\n\n";

}

void Application::ShowSeriesMenu(Series &series)const{

if (series.GetSize() == 0){

std::cout << "Series is empty" << std::endl;

return;

}

std::cout << series;

}

void Application::SortingSeriesMenu(Series &series){

if (series.GetSize() == 0){

std::cout << "Series is empty" << std::endl;

return;

}

std::cout << "QuickSort or ChoiseSort or MyQuickSort or InputSort or ShellSort <1/2/3/4/5>";

int selector\_sort;

int isWithTime;

if (!Read(selector\_sort))

{

ShowInputError();

return;

}

std::cout << "Info about time/Sort with time/ Only sort <1/2/3>";

if (!Read(isWithTime))

{

ShowInputError();

return;

}

long long time = series.DoSortingSeries(selector\_sort, isWithTime);

if (time == ERROR){

std::cout << " Invalid parametrs " << std::endl;

}

if (time >= 0){

std::cout << time << std::endl;

}

}

void Application::FillingSeriesMenu(Series &series){

int max\_number;

std::cout << "Input max number" << std::endl;

if (!Read(max\_number))

{

ShowInputError();

return;

};

std::cout << "Input quantity" << std::endl;

unsigned int quantity;

if (!Read(quantity))

{

ShowInputError();

return;

};

series.FillingSeries(max\_number, quantity);

std::cout << "Series is created" << std::endl;

}

void Application::FileInputMenu(Series &series){

std::string filename;

std::cout << "Input the file's name ";

std::cin >> filename;

//filename = "C:\data.txt";

std::ifstream fin(filename);

//Проверка успешности открытия файла для чтения

if (!fin.is\_open()) {

std::cout << "File not exist" << std::endl;

return;

}

int numb = 0;

while (fin >> numb){

series.AddElement(numb);

}

//Закрытие файла

fin.close();

}

void Application::FileOutputMenu(Series &series){

std::string filename;

std::cout << "Input the file's name ";

std::cin >> filename;

//filename = "C:\data.txt";

std::ofstream fof(filename);

//Проверка успешности открытия файла для чтения

if (!fof.is\_open()) {

std::cout << "File not exist" << std::endl;

return;

}

fof << series;

//Закрытие файла

fof.close();

}

#pragma once

#include "ISorting.h"

#include <vector>

class ChoiseSort:

public ISorting

{

public:

ChoiseSort();

~ChoiseSort();

virtual void SortElements(std::vector<int>& series) override final;

};

#include "ChoiseSort.h"

ChoiseSort::ChoiseSort() = default;

ChoiseSort::~ChoiseSort() = default;

void ChoiseSort::SortElements(std::vector<int>& series)

{

for (unsigned int i = 0; i < series.size(); ++i) // i - номер текущего шага

{

int pos = i;

for (unsigned int j = i + 1; j < series.size(); ++j) // цикл выбора наименьшего элемента

{

if (series[j] < series[i])

{

pos = j;

}

}

std::swap(series[pos], series[i]);// меняем местами наименьший с serries[i]

}

}

template<typename T>

struct TCompare {

bool operator()(const T& a, const T& b){

return (a < b);

}

};

#pragma once

#include "ISorting.h"

#include <vector>

class InputSort :

public ISorting

{

public:

InputSort();

~InputSort();

virtual void SortElements(std::vector<int>& series) override final;

};

#include "InputSort.h"

InputSort::InputSort() = default;

InputSort::~InputSort() = default;

void InputSort::SortElements(std::vector<int>& series)

{

for (int i = 1; i < series.size(); i++)

{

for (int j = i - 1; j >= 0; j--)

{

if (series[j] < series[i])

break;

std::swap(series[j], series[j + 1]);

}

}

}

#pragma once

#include <vector>

#include <algorithm>

class ISorting

{

public:

typedef std::vector<int> Container\_t;

ISorting();

virtual ~ISorting();

virtual void SortElements(Container\_t& serries) = 0;

};

#include "ISorting.h"

ISorting::ISorting() = default;

ISorting::~ISorting() = default;

#pragma once

#include "ISorting.h"

#include "Series.h"

class MyQuickSort :

public ISorting

{

public:

MyQuickSort();

~MyQuickSort();

virtual void SortElements(Container\_t& series) override final;

private:

template<typename TIterator, typename T, typename TCompare>

void QuickSort(TIterator first, TIterator last, const T& pv, TCompare cmp);

template<typename TIterator, typename TCompare>

void DoSortElements(TIterator first, TIterator last, TCompare cmp);

};

#include "MyQuickSort.h"

#include "HelpStruct.h"

MyQuickSort::MyQuickSort() = default;

MyQuickSort::~MyQuickSort() = default;

void MyQuickSort::SortElements(Container\_t& series)

{

DoSortElements(series.begin(), series.end(), TCompare<int>());

}

template<typename TIterator, typename T, typename TCompare>

void MyQuickSort::QuickSort(TIterator first, TIterator last, const T& pv, TCompare cmp){

T v = \*(first + ((last - first) / 2));

TIterator f = first;

TIterator l = last;

do {

while ((f < last) && cmp(\*f, v))

++f;

while ((l > first) && cmp(v, \*l))

--l;

if (f <= l){

std::swap(\*f, \*l);

++f;

--l;

}

} while (f <= l);

if (l > first)

QuickSort(first, l, v, cmp);

if (last > f)

QuickSort(f, last, v, cmp);

}

template<typename TIterator, typename TCompare>

void MyQuickSort::DoSortElements(TIterator first, TIterator last, TCompare cmp){

if (first < --last)

QuickSort(first, last, \*first, cmp);

}

#pragma once

#include "ISorting.h"

#include "Series.h"

class QuickSort :

public ISorting

{

public:

QuickSort();

~QuickSort();

virtual void SortElements(std::vector<int> &series) override final;

};

#include "QuickSort.h"

QuickSort::QuickSort() = default;

QuickSort::~QuickSort() = default;

void QuickSort::SortElements(std::vector<int> &series)

{

sort(series.begin(), series.end());

}

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <memory>

#include "SortingWrapper.h"

class Series

{

public:

Series();

~Series();

void AddElement(int a);

long long DoSortingSeries(int selector\_sort, int isWithTime);

void FillingSeries(int max\_number, unsigned int quantity);

typedef std::vector<int> Series\_t;

unsigned int GetSize(){ return \_series.size(); }

const Series\_t& GetSeries()const{ return \_series; };

private:

Series\_t \_series;//числовой ряд

};

std::ofstream& operator << (std::ofstream& fos, const Series& series);

std::ostream& operator << (std::ostream& os, const Series& series);

#include "Series.h"

Series::Series() = default;

Series::~Series() = default;

void Series::AddElement(int a){

\_series.push\_back(a);

}

long long Series::DoSortingSeries(int selector\_sort, int isWithTime){

SortingWrapper sorter(selector\_sort);

return sorter.Sorting(isWithTime, \_series);

}

void Series::FillingSeries(int max\_number, unsigned int quantity){

\_series.reserve(quantity);

for (int i = 0; i < quantity; i++){

\_series.push\_back(rand() % max\_number);

}

}

std::ostream& operator << (std::ostream& os, const Series& series)

{

for (auto& element : series.GetSeries())

{

os << element << ' ';

}

os << std::endl;

return os;

}

std::ofstream& operator << (std::ofstream& fos, const Series& series)

{

for (auto& element : series.GetSeries())

{

fos << element << ' ';

static int i = 0; i++;

if (i == 100){

fos << std::endl;

i = 0;

}

}

return fos;

}

#pragma once

#include "ISorting.h"

class ShellSort :

public ISorting

{

public:

ShellSort();

~ShellSort();

virtual void SortElements(std::vector<int>& series) override final;

};

#include "ShellSort.h"

ShellSort::ShellSort() = default;

ShellSort::~ShellSort() = default;

void ShellSort::SortElements(std::vector<int>& series) {

int length = series.size();

for (unsigned k = length / 2; k > 0; k /= 2){

for (unsigned i = k; i < length; i += 1)

{

for (unsigned j = i; j >= k; j -= k)

{

if (series[i] < series[j - k])

series[j] = series[j - k];

else{

break;

series[j] = series[i];

}

}

}

}

}

#pragma once

#include "ChoiseSort.h"

#include "MyQuickSort.h"

#include "QuickSort.h"

#include "InputSort.h"

#include "ShellSort.h"

class SortingFactory

{

public:

SortingFactory()=default;

~SortingFactory() = default;

ISorting\* FabricMethod(int selector);

private:

static const int CHOISE = 1;

static const int QUICK = 2;

static const int MYQUICK = 3;

static const int INPUT = 4;

static const int SHELL = 5;

};

#include "SortingFactory.h"

ISorting\* SortingFactory::FabricMethod(int selector){

switch (selector)

{

case CHOISE:

return new ChoiseSort();

break;

case QUICK:

return new QuickSort();

break;

case MYQUICK:

return new MyQuickSort();

break;

case INPUT:

return new InputSort();

break;

case SHELL:

return new ShellSort();

break;

default:

return nullptr;

}

}

#pragma once

#include <ctime>

#include <chrono>

#include "SortingFactory.h"

#pragma warning

using std::chrono::duration\_cast;

using std::chrono::steady\_clock;

class SortingWrapper{

public:

SortingWrapper(int selector\_sort);

~SortingWrapper();

//Селектор сортировки

long long Sorting(int isWithTime, std::vector<int>& series);

private:

ISorting\* \_sorter;// Решил здесь оставить ,так как думаю ,что пользователю лучше не стоит знать об указателях в принципе

static const int INFO\_TIME = 1;

static const int SORT\_WITH\_TIME = 2;

static const int ONLY\_SORT = 3;

static const int NOT\_TIME = -1;

static const int ERROR = -404;//- так как время может быть положительным ,если оно совпадет ,то будет печаль

//Сортировка с засеканием времени

long long SortWithTime(ISorting\* ptr\_sort, std::vector<int>& series);

};

#include "SortingWrapper.h"

SortingWrapper::SortingWrapper(int selector\_sort) {

SortingFactory \_factory;

\_sorter = \_factory.FabricMethod(selector\_sort);

}

SortingWrapper::~SortingWrapper(){

delete \_sorter;

}

long long SortingWrapper::Sorting(int isWithTime, std::vector<int>& series){

if (\_sorter == nullptr){

isWithTime = ERROR;//Если введен неправильный селектор сортировки ,то переброс сase isWithTime в ошибки

//В том смысле ,если селектор выбран неправильно ,смысл проверять другой селектор

}

std::vector<int> fake\_series = series;

switch (isWithTime)

{

case INFO\_TIME://в наносекундах

return this->SortWithTime(\_sorter, fake\_series);

break;

case SORT\_WITH\_TIME:

return this->SortWithTime(\_sorter, series);

break;

case ONLY\_SORT:

\_sorter->SortElements(series);

return NOT\_TIME;

break;

default:

return ERROR;

break;

}

}

long long SortingWrapper::SortWithTime(ISorting\* ptr\_sort, std::vector<int>& series){

using std::chrono::duration\_cast;

using std::chrono::steady\_clock;

steady\_clock::time\_point start = steady\_clock::now();

{

ptr\_sort->SortElements(series);

}

steady\_clock::time\_point end = steady\_clock::now();

return duration\_cast<std::chrono::nanoseconds>(end - start).count();

}

#include "Application.h"

void main()

{

Application app;

app.RunApp();

}