МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №10**

по дисциплине: Объектно-ориентированное программирование Тема: Закрепление навыков программирования в объектно-ориентированном стиле. Визуальные компоненты. Знакомство с QT.

Выполнил: студент группы ПВ-223

Дмитриев А.А.

Проверил:

Черников С.В.

Белгород 2024 г.

**Цель работы:** приобретение практических навыков создания приложений на языке С++.

**Задание:**

Для выполнения лабораторных работ с использованием операционной системы Microsoft Windows необходимо установить следующее программное обеспечение: Tortoise Mercurial актуальной версии; Qt SDK или иной дистрибутив Qt согласованный с преподавателем. При изучении дисциплины необходимо обеспечить доступ в интернет для осуществления доступа к системе контроля версий. Лабораторные работы выполняются на объектно-ориентированном языке программирования C++. Изменения, вносимые студентами в свои проекты, следует периодически отдавать на сервер системы контроля версий для осуществления контроля над выполнением заданий со стороны преподавателя. Для успешной сдачи лабораторной работы следует выполнить следующий перечень заданий.

**Задание 1.**

Создать репозиторий под контролем git на открытой площадке gitlab.com

Задать имя репозитория &lt;год&gt;\_&lt;группа&gt;\_&lt;имя студента в транслите&gt;\_&lt;номер

варианта по журналу&gt;.

**Задание 2.**

Выполнить проектирование задачи в соответствии с вариантом (табл. 1).

Для реализации поставленной задачи необходимо с проектировать, реализовать

и использовать шаблон «умные указатели». Соответственно это учесть при

проектировании программного обеспечения.

**Задача 3.**

Выполнить реализацию в соответствии с вариантом задачи (табл. 1),

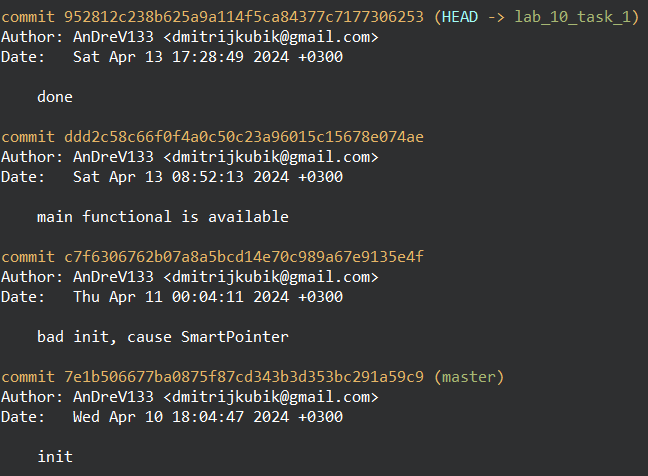
использую среду разработки QT. Реализация должна быть кроссплатформенной

и выполнена на основе графических окон.

2 вариант: Создать приложения для просмотра графического материала, поиск файлов с разрешениями jpeg, bmp, png, сохранение ссылок на них в файл, для

быстрого доступа при повторном запуске приложения.

№1

****

№2

#ifndef SMARTPOINTER\_H

#define SMARTPOINTER\_H

template<typename T>

class SmartPointer {

T \*obj;

public:

SmartPointer() {}

SmartPointer(T \*obj) {

this->obj = obj;

}

~SmartPointer() {

delete obj;

}

T\* operator->() { return obj; }

T& operator\* () { return \*obj; }

};

#endif // SMARTPOINTER\_H

№3

#ifndef FILESAVER\_H

#define FILESAVER\_H

#include <forward\_list>

#include <string>

#include <fstream>

#include <QStringList>

#include <QSet>

#include <QList>

#include "ISaveable.h"

#include "SmartPointer.hpp"

class FileSaver {

private:

std::string recordFileName;

public:

FileSaver(std::string recordFileName) {

this->recordFileName = recordFileName;

}

void save(ISaveable &s) {

std::ofstream recordFile(recordFileName, std::ofstream::app);

if (recordFile.is\_open()) {

recordFile << s.onSave() << '\n';

recordFile.close();

} else {

throw std::ios\_base::failure("make or write to file is impossible");

}

}

QStringList getPathesToImages() {

std::ifstream recordFile(recordFileName, std::ofstream::app);

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

throw std::ios\_base::failure("file cant be read");

}

SmartPointer<QSet<QString>> set = new QSet<QString>();

std::string buf;

while (getline(recordFile, buf))

{

set->insert(QString(buf.c\_str()));

}

recordFile.close();

return QStringList(QList<QString>(set->begin(), set->end()));

}

};

#endif // FILESAVER\_H

#ifndef ISAVEABLE\_H

#define ISAVEABLE\_H

#include <string>

class ISaveable {

public:

virtual std::string onSave() = 0;

};

#endif // ISAVEABLE\_H

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "Viewer.h"

#include "SmartPointer.hpp"

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "FileSaver.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_showButton\_clicked();

void on\_reviewButton\_clicked();

private:

Ui::MainWindow \*ui;

QString fileName;

QWidget\* widget;

FileSaver\* fs;

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QString>

#include <QLabel>

#include <QPixmap>

#include <QWindow>

#include <QWidget>

#include <memory>

#include <iostream>

#include <exception>

#include "SmartPointer.hpp"

#include "FileSaver.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

fs = new FileSaver("cache.txt");

ui->comboBox->addItems(fs->getPathesToImages());

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_reviewButton\_clicked()

{

fileName = QFileDialog::getOpenFileName(

this,

QString(),

QString(),

"jpeg (\*.jpeg);;bmp (\*.bmp);;png (\*.png)"

);

SmartPointer<Viewer> v = new Viewer(fileName.toStdString());

ui->comboBox->setCurrentText(fileName);

fs->save(\*v);

ui->comboBox->clear();

ui->comboBox->addItems(fs->getPathesToImages());

}

void MainWindow::on\_showButton\_clicked()

{

widget = new QWidget();

QLabel \*label = new QLabel();

QString path = ui->comboBox->currentText();

qDebug(path.toStdString().c\_str());

label->setText(path);

label->setPixmap(QPixmap(path));

QVBoxLayout \*layout = new QVBoxLayout();

layout->addWidget(label);

widget->setLayout(layout);

widget->show();

}

#ifndef VIEWER\_H

#define VIEWER\_H

#include <string>

#include "ISaveable.h"

class Viewer : public ISaveable {

private:

std::string filePath;

std::string getFileExtension(const std::string& filePath) {

auto pos = filePath.rfind('.');

if (pos <= 0) return "";

return filePath.substr(pos+1, std::string::npos);

}

public:

Viewer(const std::string &filePath) {

std::string fileExt = getFileExtension(filePath);

if (fileExt == "jpeg" || fileExt == "bmp" || fileExt == "png")

this->filePath = filePath;

else

std::\_\_throw\_runtime\_error("uncorrect file extension.");

}

std::string onSave() override {

return filePath;

}

};

#endif // VIEWER\_H

#include "mainwindow.h"

#include <QApplication>

#include <QLocale>

#include <QTranslator>

#include <iostream>

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

{

QApplication a(argc, argv);

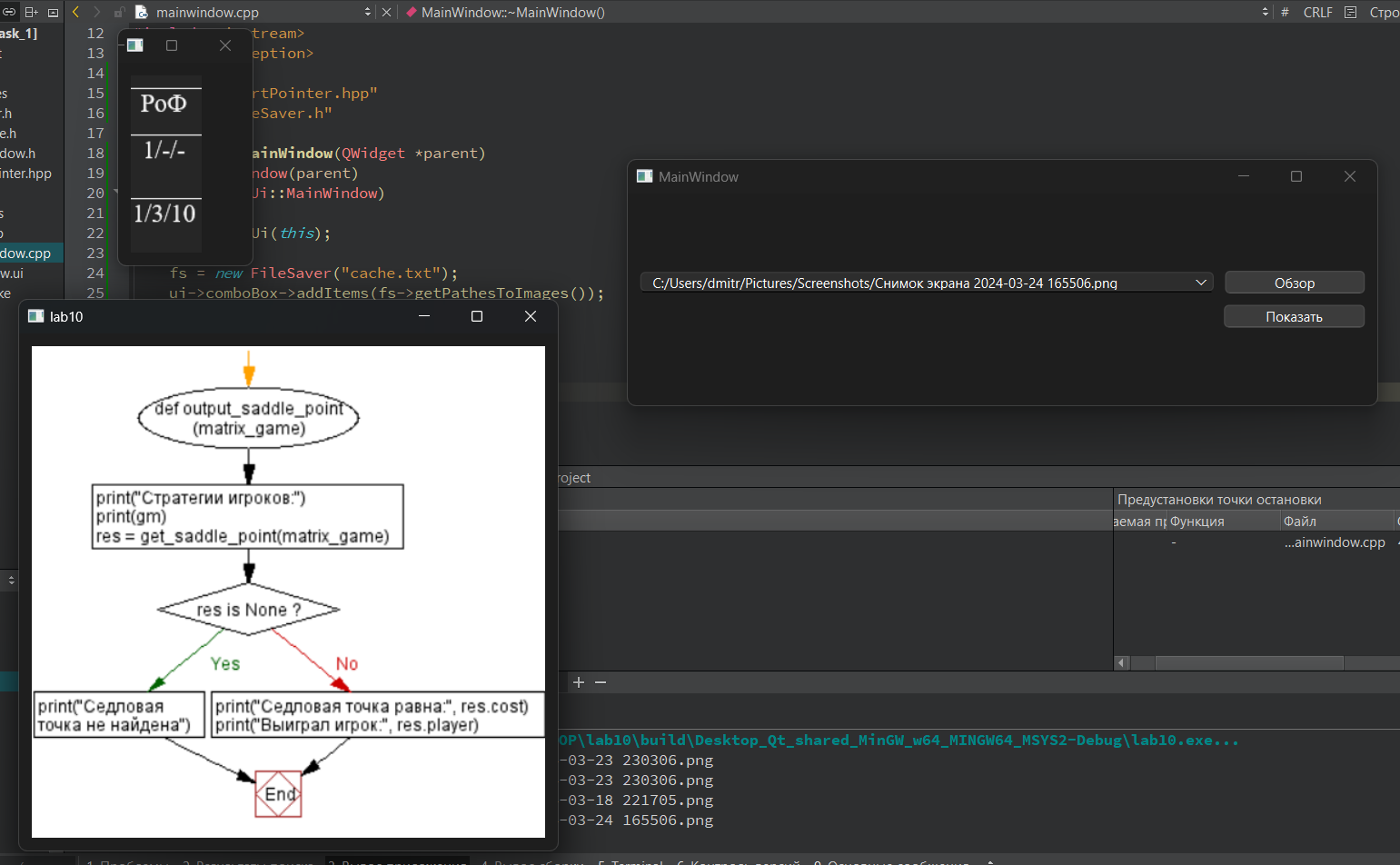
MainWindow w;

w.show();

return a.exec();

}

Пример работы:

****

**Вывод:** В ходе лабораторной работы закрепили навыки программирования в объектно-ориентированном стиле. Познакомились с инструментарием QT.