Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Основы алгоритмизации и программирования

Отчёт

по учебной практике(ознакомительной)

на тему:

Система тестирования

(конструктор тестов)

Студент

Руководитель

Д.С.Кончик

В.Д.Владымцев

МИНСК 2022

**CОДЕРЖАНИЕ**

ВВЕДЕНИЕ3

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ4

2. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ5

2.1. UML ДИАГРАММА5

2.2. СХЕМА ГОЛОВНОГО МОДУЛЯ5

2.3. USE-CASE ДИАГРАММА5

3. ИНТЕРФЕЙСЫ КЛАССОВ ПРОГРАММЫ6

3.1. TaskView.h

3.2. Task.h

3.3. Comparison.h

3.4. Entry.h

3.5. MultipleChoice.h

3.6. OrderIndication.h

3.7. SingleChoice.h

3.8. TaskService.h

3.9. TaskViewWidget.h

3.10. TaskChoose.h

3.11. Test.h

3.12. mainwindow.h

3.13. StatusBar.h

3.14. TaskListView.h

3.15. TestNameLine.h

3.16. TextEdit.h

4. РЕЗУЛЬТАТЫ18

ЗАКЛЮЧЕНИЕ21

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ22

ПРИЛОЖЕНИЕ А23

ПРИЛОЖЕНИЕ Б24

ПРИЛОЖЕНИЕ В25

**ВВЕДЕНИЕ**

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д.

C++ позволяет писать приложения в объектно-ориентированном стиле, представляя программу как совокупность взаимодействующих между собой классов и объектов. Что упрощает создание крупных приложений.

Qt — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++. Одним из преимуществ фреймворка является подробная документация, сопровождающаяся большим количеством примеров. Исходный код примеров содержит подробные комментарии и описание, что также упрощает изучение Qt.

Qt позволяет запускать написанное с его помощью [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в большинстве современных [операционных систем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) путём простой компиляции программы для каждой системы без изменения [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Включает в себя все основные [классы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%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)), которые могут потребоваться при разработке [прикладного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Является полностью объектно-ориентированным.

1. **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Создание приложения для создания универсальных тестов с использованием библиотек Qt со следующим функционалом:

1. Создание теста с любым количеством вариативных заданий.
2. Возможность выбора одного из типов заданий: одиночный и множественный выбор, сопоставление, указание порядка, ручной ввод.
3. Настройка задания: количества получаемых баллов за правильный ответ и количество пунктов в задании.
4. Сохранение созданного теста в файл и открытие его при запуске программы для последующего редактирования.
5. Экспорт теста в PDF-формат.

**2. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

**2.1. Схема головного модуля**

UML-диаграмма головного модуля представлена в **ПРИЛОЖЕНИИ А.**

**2.2. Алгоритм приложения**

Схема головного модуля представлена в **ПРИЛОЖЕНИИ Б.**

**2.3. Алгоритм приложения**

USE-CASE диаграмма представлена в **ПРИЛОЖЕНИИ В.**

**3. ИНТЕРФЕЙСЫ КЛАССОВ ПРОГРАММЫ**

**3.1. TaskView.h**

#ifndef TASKVIEW\_H

#define TASKVIEW\_H

#include <QDebug>

#include <QWidget>

#include <QGridLayout>

#include <QTextEdit>

namespace Ui {

class TaskView;

}

// Максимальная длина имени задания

#define MAX\_TASK\_NAME\_LEN 20

// ВИЗУАЛЬНАЯ ЧАСТЬ ЗАДАНИЯ

class TaskView : public QWidget

{

Q\_OBJECT

public:

explicit TaskView(QWidget \*parent);

~TaskView();

signals:

// Задание и название задания изменено

void taskEdited();

void nameEdited();

protected:

Ui::TaskView \*ui;

// layout и виджет с вариантами ответов

QGridLayout\* optionsLayout;

QWidget\* optionsWidget;

};

#endif // TASKVIEW\_H

**3.2. Task.h**

#ifndef TASK\_H

#define TASK\_H

#include <QString>

#include <QFile>

#include <QVector>

#include <QRadioButton>

#include <QLineEdit>

#include <QPair>

#include <QComboBox>

#include <QLabel>

#include <QCheckBox>

#include "TaskView.h"

#include "ui\_TaskView.h"

// Начальное, минимальное и максимальное

// количество вариантов ответов

#define START\_OPT\_NUM 5

#define MAX\_OPT\_NUM 10

#define MIN\_OPT\_NUM 2

// Максимальная длина варианта ответа

#define MAX\_POINT\_LEN 33

// ТИПЫ ЗАДАНИЙ

enum TaskType {

SINGLE\_CHOICE, // Одиночный выбор

MULTIPLE\_CHOICE, // Множественный выбор

ORDER\_INDICATION, // Указание порядка

COMPARISON, // Сопоставление

ENTRY, // Ввод

AMOUNT // Число типов заданий

};

// БАЗОВЫЙ КЛАСС ДЛЯ ВСЕХ ЗАДАНИЙ

class Task : public TaskView {

protected:

TaskType taskType; // Тип задания

int32\_t ans\_amount; // Число ответов

// Линии ввода вариантов ответов

QVector<QLineEdit\*> optionsLinesEdit;

// Добавить и удалить вариант ответов

virtual void plus() = 0;

virtual void minus() = 0;

void checkOptionsLimit(); // Проверка количества вариантов ответов

void resetTaskName(); // Сбросить имя задания

// Копировать основные свойства заданий в это задание из другого

void copy\_general(const Task\* copyFrom);

public:

Task(TaskType type, QWidget\* parent);

QString getQuestion(); // Вопрос задания

QString getName(); // Имя задания

TaskType getType(); // Тип задания

int getPrice(); // Получить стоимость

// Конвертировать в строку

virtual QString toString(bool solved) = 0;

// Записать и считать задание

virtual void write(QFile\* file) = 0;

virtual void read(QFile\* file) = 0;

// Получить копию

virtual Task\* getCopy() = 0;

// Записать и считать общие свойства для всех типов заданий

void write\_general(QFile\* file);

void read\_general(QFile\* file);

};

#endif // TASK\_H

**3.3. Comparison.h**

#ifndef COMPARISON\_H

#define COMPARISON\_H

#include "Task.h"

// СОПОСТАВЛЕНИЕ

class Comparison : public Task {

private:

// Комбобоксы и надписи с номерами вариантов ответов

QVector<QComboBox\*> comboBox;

QVector<QLabel\*> label;

protected:

void plus() override;

void minus() override;

public:

explicit Comparison(QWidget \*parent = nullptr);

void write(QFile\* file) override;

void read(QFile\* file) override;

QString toString(bool) override;

Task\* getCopy() override;

};

#endif // COMPARISON\_H

**3.4. Entry.h**

#ifndef ENTRY\_H

#define ENTRY\_H

#include "Task.h"

// ВВОД

class Entry : public Task {

protected:

void plus() override {}

void minus() override {}

public:

explicit Entry(QWidget\* parent = nullptr);

void write(QFile\* file) override;

void read(QFile\* file) override;

QString toString(bool) override;

Task\* getCopy() override;

};

#endif // ENTRY\_H

**3.5. MultipleChoice.h**

#ifndef MULTIPLECHOICE\_H

#define MULTIPLECHOICE\_H

#include "Task.h"

// МНОЖЕСТВЕННЫЙ ВЫБОР

class MultipleChoice : public Task {

private:

// Чекбоксы вариантов ответов

QVector<QCheckBox\*> optionsCheckBoxes;

protected:

void plus() override;

void minus() override;

public:

explicit MultipleChoice(QWidget \*parent = nullptr);

void write(QFile\* file) override;

void read(QFile\* file) override;

QString toString(bool) override;

Task\* getCopy() override;

};

#endif // MULTIPLECHOICE\_H

**3.6. OrderIndication.h**

#ifndef ORDERINDICATION\_H

#define ORDERINDICATION\_H

#include "Task.h"

// УКАЗАНИЕ ПОРЯДКА

class OrderIndication : public Task {

private:

// Комбобоксы вариантов ответов

QVector<QComboBox\*> optionsComboBoxes;

protected:

void plus() override;

void minus() override;

public:

explicit OrderIndication(QWidget\* parent = nullptr);

void write(QFile\* file) override;

void read(QFile\* file) override;

QString toString(bool) override;

Task\* getCopy() override;

};

#endif // ORDERINDICATION\_H

**3.7. SingleChoice.h**

#ifndef SINGLECHOICE\_H

#define SINGLECHOICE\_H

#include "Task.h"

// ОДИНОЧНЫЙ ВЫБОР

class SingleChoice : public Task {

private:

// Радиобаттоны вариантов ответов

QVector<QRadioButton\*> optionsRadioButtons;

protected:

void plus() override;

void minus() override;

public:

explicit SingleChoice(QWidget \*parent = nullptr);

void write(QFile\* file) override;

void read(QFile\* file) override;

QString toString(bool) override;

Task\* getCopy() override;

};

#endif // SINGLECHOICE\_H

**3.8. TaskService.h**

#ifndef TASKSERVICE\_H

#define TASKSERVICE\_H

#include <QString>

#include "TaskTypes\SingleChoice.h"

#include "TaskTypes\MultipleChoice.h"

#include "TaskTypes\OrderIndication.h"

#include "TaskTypes\Comparison.h"

#include "TaskTypes\Entry.h"

class Task;

// Служебный класс для заданий

class TaskService {

public:

// Получить название типа задания по enum

static QString strByType(TaskType type) {

if (type == TaskType::SINGLE\_CHOICE)

return "Одиночный выбор";

else if (type == TaskType::MULTIPLE\_CHOICE)

return "Множественный выбор";

else if (type == TaskType::ORDER\_INDICATION)

return "Указание порядка";

else if (type == TaskType::COMPARISON)

return "Сопоставление";

else if (type == TaskType::ENTRY)

return "Ввод";

else

throw std::exception();

}

// Получить задание по enum

static Task\* objByType(TaskType type) {

if (type == TaskType::SINGLE\_CHOICE)

return new SingleChoice;

else if (type == TaskType::MULTIPLE\_CHOICE)

return new MultipleChoice;

else if (type == TaskType::ORDER\_INDICATION)

return new OrderIndication;

else if (type == TaskType::COMPARISON)

return new Comparison;

else if (type == TaskType::ENTRY)

return new Entry;

else

throw std::exception();

}

};

#endif // TASKSERVICE\_H

**3.9. TaskViewWidget.h**

#ifndef TASKVIEWWIDGET\_H

#define TASKVIEWWIDGET\_H

#include <QWidget>

namespace Ui {

class TaskViewWidget;

}

class TaskViewWidget : public QWidget

{

Q\_OBJECT

public:

explicit TaskViewWidget(QWidget \*parent = nullptr);

~TaskViewWidget();

void setTask(QWidget\* task); // Установить задание в окно показа задания

void setNoTasks(); // Надпись "NO TASKS"

void setNoTest(); // Надпись "NO ACTIVE TEST"

private:

Ui::TaskViewWidget \*ui;

QWidget\* activeWidget; // Какой виджет сейчас стоит в окне

};

#endif // TASKVIEWWIDGET\_H

**3.10. TaskChoose.h**

#ifndef TASKCHOOSE\_H

#define TASKCHOOSE\_H

#include <QDialog>

#include <QRadioButton>

#include <QLayout>

#include <QPushButton>

#include <QVector>

#include <QLabel>

#include <QDebug>

#include <QString>

#include <QWidget>

#include "TaskTypes\Task.h"

// RadioButton с сигналом двойного клика мыши

class RadioButton : public QRadioButton {

Q\_OBJECT

signals:

void doubleClicked();

protected:

virtual void mouseDoubleClickEvent(QMouseEvent\*) override {

emit doubleClicked();

}

public:

RadioButton(QString text, QWidget\* parent = nullptr)

: QRadioButton(text, parent)

{

}

};

// Выбор типа задания

class TaskChoose : public QDialog

{

Q\_OBJECT

private:

// Ok | Cancel

QPushButton\* Ok;

QPushButton\* Cancel;

QVBoxLayout\* componentsLayout; // layout со всеми компонентами окна

QVector<RadioButton\*> radioButton; // Вектор RadioButton'ов

TaskType type; // Тип выбранного задания

public slots:

// Ok | Cancel: clicked

void on\_Ok\_clicked();

void on\_Cancel\_clicked();

public:

TaskChoose();

TaskType getType(); // Получить тип выбранного задания

Task\* getTask(); // Получить выбранное задание

void reset(); // Сбросить (radiobutton в исх положение)

};

#endif // TASKCHOOSE\_H

**3.11. Test.h**

#ifndef TEST\_H

#define TEST\_H

#include <QList>

#include "Task\TaskTypes\Task.h"

#include "Task\TaskService.h"

#include "ModifiedClasses/StatusBar.h"

#include <list>

#include <QWidget>

#include <QFile>

#include <QUrl>

#include <QPrinter>

// Состояние теста

enum TestStatus {

CREATED, // Создан

OPENED, // Открыт из файла

EDITED, // Изменен

SAVED, // Сохранен

REMOVED // Удален

};

// ТЕСТ

class Test : public QObject

{

Q\_OBJECT

private:

QVector<Task\*> taskList; // Задания

QPair<bool, int32\_t> passingTime; // Время прохождения

TestStatus status; // Состояние

QUrl url; // Путь к тесту

void setStatus(TestStatus); // Установить статус

QString toString(bool solved); // Конвертировать тест в строку

int sumOfPrices(); // Стоимость всех заданий

signals:

// Сигналы теста

void testCreated();

void testOpened();

void testEdited();

void testSaved();

void testRemoved();

public:

Test(QUrl);

static StatusBar\* statusBar; // Статусбар из MainWindow

// Добавить, удалить, вставить задание

void addTask(Task\*);

void removeTask(int);

void pasteTask(int numberOfPasted, int placeToPaste);

// Открыть, Создать, Сохранить, Удалить тест

void open();

void create();

void save();

void saveAs(QUrl);

void savePDF(QUrl);

void remove();

void moveTask(int, int); // Изменить номер заданию

bool hasUnsavedChanges(); // Есть ли несохраненные изменения

int size(); // Количество заданий

Task\* back(); // Последнее задание в тесте

Task\* operator[](int); // Задание теста по индексу

QUrl getUrl(); // Получить Url теста

QString getFileName(); // Получить имя файла

// Время прохождения теста

void setPassingTime(bool, int);

void setPassingTime(QPair<bool, int32\_t>);

QPair<bool, int32\_t> getPassingTime();

};

#endif // TEST\_H

**3.12. mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include <QGridLayout>

#include <QFileDialog>

#include <QDebug>

#include <QFileInfo>

#include <QCloseEvent>

#include <QMessageBox>

#include <QAbstractItemModel>

#include <QPrinter>

#include <QTextDocument>

#include <QApplication>

#include "Task\TaskTypes\Task.h"

#include "Test.h"

#include "Task\TaskChoose.h"

// Сколько символов максимум влезет в список заданий теста

#define MAX\_LIST\_VIEW\_LABEL\_LEN 20

// Максимум и минимум заданий в тесте

#define MAX\_TASK\_NUM 100

#define MIN\_TASK\_NUM 0

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

// TaskOperations

void addTask();

void removeTask();

void checkTasksLimit();

// TestOperations

void createTest();

void openTest();

void saveTest();

void saveAsTest();

void removeTest();

void exportPDF();

// testSlots

void connectTestSignals(Test\*);

void slot\_testRemoved();

void slot\_testCreated();

void slot\_testOpened();

// tasksListView

void taskListViewPrint(); // Распечатать список заданий теста

void taskListView\_clicked(const QModelIndex&); // Клик по списку заданий теста

void taskListView\_rowsMoved(const QModelIndex&, int, int, const QModelIndex&, int); // Задание передвинуто

void on\_taskPasted(int numberOfPasted, int currentRow);

void on\_taskCopied(int numberOfCopied);

// mainwindow

int checkUnsavedChanges(); // Проверка на наличие несохраненных изменений

virtual void closeEvent(QCloseEvent\*) override; // При закрытии окна

// time

void timeCheckBox\_clicked(bool checked);

void timeSpinBox\_valueChanged(int arg1);

private:

Ui::MainWindow \*ui;

Test\* test; // Текущий тест

TaskChoose\* taskChoose; // Окно выбора типа задания

};

#endif // MAINWINDOW\_H

**3.13. StatusBar.h**

#ifndef STATUSBAR\_H

#define STATUSBAR\_H

#include <QStatusBar>

#include <QPalette>

#include <QString>

#include <Task\TaskService.h>

#include <QUrl>

// Продолжительность сообщения в статусбаре

#define STATUS\_BAR\_DURATION 3000

class StatusBar : public QStatusBar {

public:

StatusBar(QWidget\* parent = nullptr);

void taskCopied(QString taskName);

void taskPasted(QString taskName);

void taskAdded(TaskType type);

void taskRemoved(QString taskName);

void testOpened(QUrl url);

void testSaved(QUrl url);

void testSavedAs(QUrl url);

void testSavedPDF(QUrl url);

void testCreated(QUrl url);

void testRemoved(QUrl url);

};

#endif // STATUSBAR\_H

**3.14. TaskListView.h**

#ifndef TASKLISTVIEW\_H

#define TASKLISTVIEW\_H

#include <QListWidget>

#include <QKeyEvent>

#include "ModifiedClasses/StatusBar.h"

class TaskListView

: public QListWidget

{

Q\_OBJECT

private:

int copiedRow;

protected:

virtual void keyPressEvent(QKeyEvent \*event) override;

signals:

void taskPasted(int numberOfPasted, int currentRow);

void taskCopied(int numberOfCopied);

public:

TaskListView(QWidget\* parent = nullptr);

void rowDeleted(int row);

void rowMoved(int from, int to);

};

#endif // TASKLISTVIEW\_H

**3.15. TestNameLine.h**

#ifndef TESTNAMELINE\_H

#define TESTNAMELINE\_H

#include <QLineEdit>

#include <QEvent>

#include <QMouseEvent>

#include <QDesktopServices>

#include <QString>

#include <QFileInfo>

// ЛИНИЯ С ИМЕНЕМ ТЕСТА

class TestNameLine : public QLineEdit

{

private:

QUrl fileUrl; // Путь к файлу

public slots:

// Добавление и удаление "\*" возле имени

void addStar();

void removeStar();

public:

TestNameLine(QWidget\* parent = nullptr);

// Двойной клик по линии

virtual void mouseDoubleClickEvent(QMouseEvent\*);

// Установить Url

void setFileUrl(QUrl fileUrl);

};

#endif // TESTNAMELINE\_H

**3.16. TextEdit.h**

#ifndef TEXTEDIT\_H

#define TEXTEDIT\_H

#include <QTextEdit>

// Максимальная длина вопроса

#define MAX\_QUESTION\_LEN 2000

class TextEdit : public QTextEdit {

private:

void onTextChanged();

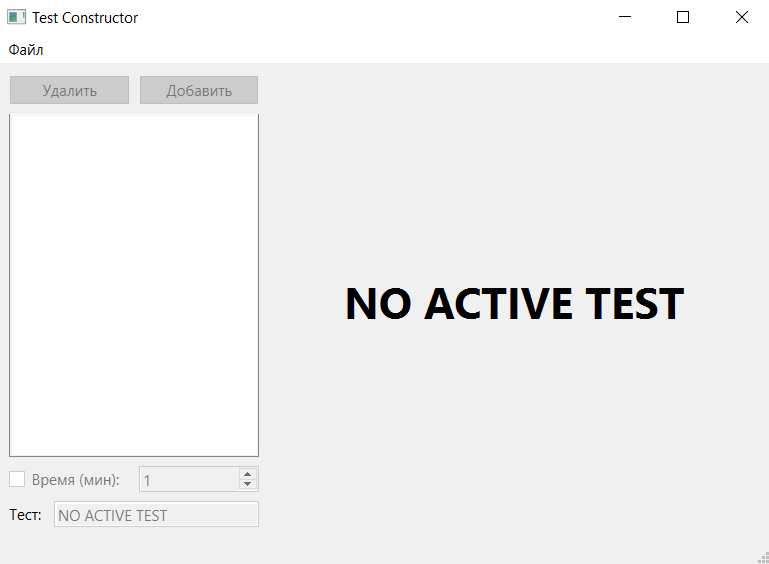
public:

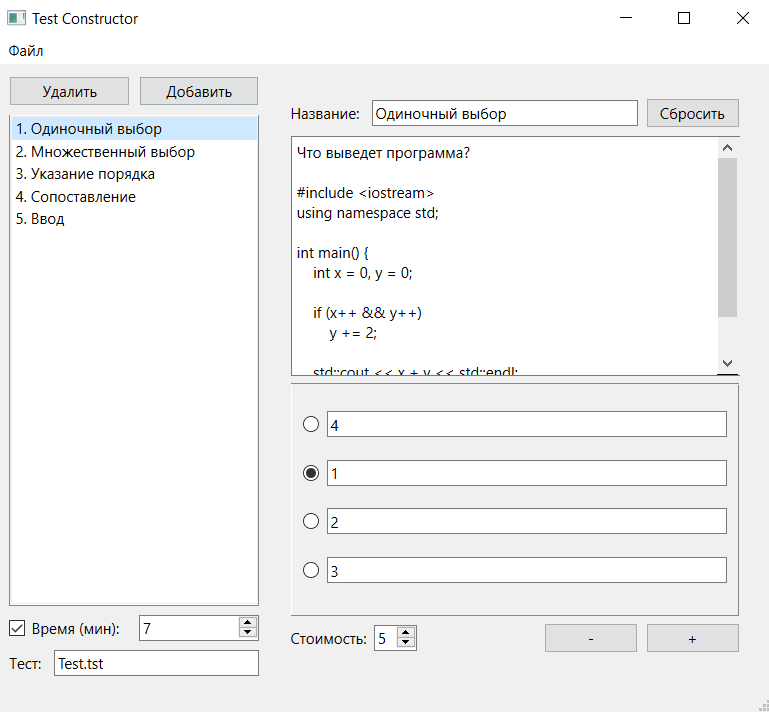
TextEdit(QWidget\* parent = nullptr);

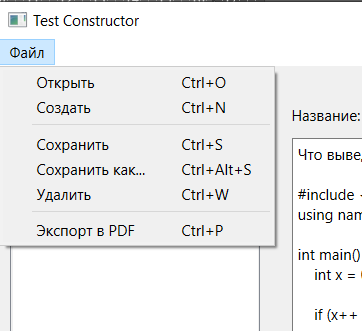
};

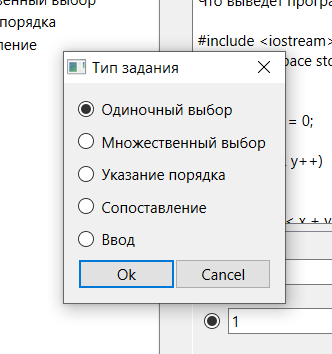
#endif // TEXTEDIT\_H

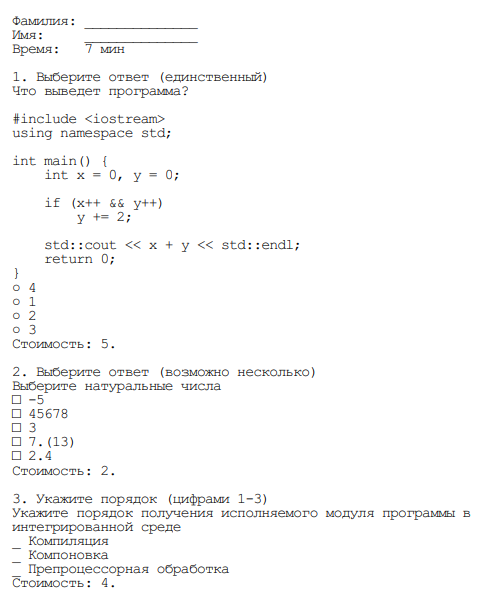
**4. РЕЗУЛЬТАТЫ**

 Интерфейс программы при ее запуске:

Вид уже созданного теста:

Меню управления тестом:

Меню добавления нового задания:

Часть теста, экспортированного в PDF формат:

**ЗАКЛЮЧЕНИЕ**

Был создан простой и удобный в использовании инструмент для создания тестов, который позволяет создавать тесты, не требуя от пользователя навыков в программировании. Имеет наглядный интерфейс, поэтому данное приложение с легкостью сможет освоить каждый.

В процессе работы были закреплены навыки в программировании на языке C++, создание интерфейсов.

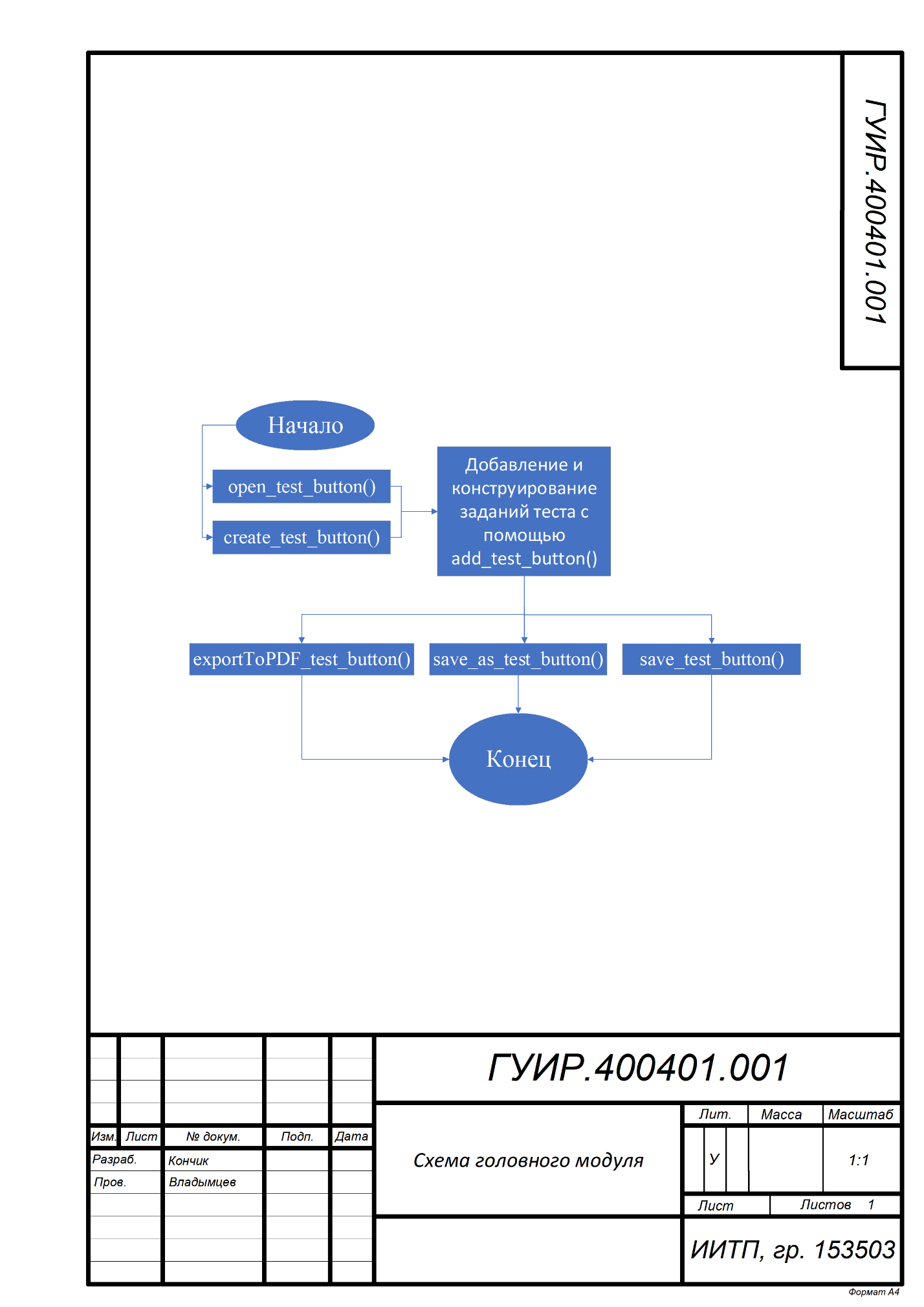
**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Qt 5.10. Профессиональное программирование на С++. — СПб.: БХВ-Петербург, 2018. — 1072 с.: ил. — (В подлиннике)

Qt Documentation [Электронный ресурс]. – Режим доступа: <https://doc.qt.io/>

cppreference [Электронный ресурс]. — Режим доступа: <https://en.cppreference.com/w/>

**ПРИЛОЖЕНИЕ А.**

**ПРИЛОЖЕНИЕ Б.**

**ПРИЛОЖЕНИЕ В.**

