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

(тестер)

Студент

Руководитель

Е.С.Кахновский

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. mainwindow.h

3.11. Test.h

3.12. TestPassingWindow.h

3.13. TestResult.h

3.14. TestResultWindow.h

4. РЕЗУЛЬТАТЫ16

ЗАКЛЮЧЕНИЕ18

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ19

ПРИЛОЖЕНИЕ А20

ПРИЛОЖЕНИЕ Б21

ПРИЛОЖЕНИЕ В22

**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 прохождения тестов с использованием библиотек Qt со следующим функционалом:

1. Авторизация пользователя при открытии теста.
2. Прохождение теста и информирование об оставшемся времени.
3. Система подсчета баллов пройденного теста.
4. Возможность просмотра своих ошибок после прохождения теста.

**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;

}

// ВИДЫ ОТВЕТОВ

enum Answers {

RIGHT, // Правильные

USER // Пользователя

};

// ВИЗУАЛЬНАЯ ЧАСТЬ ЗАДАНИЯ

class TaskView : public QWidget

{

Q\_OBJECT

protected:

Ui::TaskView \*ui;

// layout и виджет с вариантами ответов

QGridLayout\* optionsLayout;

QWidget\* optionsWidget;

Answers currentAnswers;

public:

explicit TaskView(QWidget \*parent);

~TaskView();

};

#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

int32\_t price; // Стоимость задания

// Линии ввода вариантов ответов

QVector<QLineEdit\*> optionsLinesEdit;

// Добавить и удалить вариант ответов

virtual void plus() = 0;

virtual void minus() = 0;

// общие действие для всех типов заданий

void read\_general(QFile\* file);

void setWatchMistakesEnabled\_general(bool);

// Показать ответы

virtual void switchAnswers(Answers answers) = 0;

protected slots:

void on\_switchAnswersButton\_clicked();

public:

Task(TaskType type, QWidget\* parent);

QString getQuestion(); // Вопрос задания

QString getName(); // Имя задания

TaskType getType(); // Тип задания

// Установить "Задание: a / b"

void setTaskNum(int number, int total);

// Как выполнено задание. <правильно?, цена>

virtual QPair<bool, int> execute() = 0;

// Активировать компоненты просмотра ошибок

virtual void setWatchMistakesEnabled(bool) = 0;

// Cчитать задание

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

};

#endif // TASK\_H

**3.3. Comparison.h**

#ifndef COMPARISON\_H

#define COMPARISON\_H

#include "Task.h"

// СОПОСТАВЛЕНИЕ

class Comparison : public Task {

private:

QVector<int32\_t> rightAnswers; // Правильные ответы

QVector<int32\_t>\* userAnswers; // Ответы пользователя

// Комбобоксы и надписи с номерами вариантов ответов

QVector<QComboBox\*> comboBox;

QVector<QLabel\*> label;

protected:

QPair<bool, int> execute() override;

void plus() override;

void minus() override;

void switchAnswers(Answers answers) override;

public:

Comparison(QWidget \*parent = nullptr);

void read(QFile\* file) override;

void setWatchMistakesEnabled(bool) override;

};

#endif // COMPARISON\_H

**3.4. Entry.h**

#ifndef ENTRY\_H

#define ENTRY\_H

#include "Task.h"

// ВВОД

class Entry : public Task {

private:

QString rightAnswer; // Правильный ответ

QString\* userAnswer; // Ответ пользователя

protected:

QPair<bool, int> execute() override;

void plus() override {}

void minus() override {}

void switchAnswers(Answers answers) override;

public:

Entry(QWidget\* parent = nullptr);

void read(QFile\* file) override;

void setWatchMistakesEnabled(bool) override;

};

#endif // ENTRY\_H

**3.5. MultipleChoice.h**

#ifndef MULTIPLECHOICE\_H

#define MULTIPLECHOICE\_H

#include "Task.h"

// МНОЖЕСТВЕННЫЙ ВЫБОР

class MultipleChoice : public Task {

private:

QVector<bool> rightAnswers; // Правильные ответы

QVector<bool>\* userAnswers; // Ответы пользователя

// Чекбоксы вариантов ответов

QVector<QCheckBox\*> optionsCheckBoxes;

protected:

QPair<bool, int> execute() override;

void plus() override;

void minus() override;

void switchAnswers(Answers answers) override;

public:

MultipleChoice(QWidget \*parent = nullptr);

void read(QFile\* file) override;

void setWatchMistakesEnabled(bool) override;

};

#endif // MULTIPLECHOICE\_H

**3.6. OrderIndication.h**

#ifndef ORDERINDICATION\_H

#define ORDERINDICATION\_H

#include "Task.h"

// УКАЗАНИЕ ПОРЯДКА

class OrderIndication : public Task {

private:

QVector<int32\_t> rightAnswers; // Правильные ответы

QVector<int32\_t>\* userAnswers; // Ответы пользователя

// Комбобоксы вариантов ответов

QVector<QComboBox\*> optionsComboBoxes;

protected:

QPair<bool, int> execute() override;

void plus() override;

void minus() override;

void switchAnswers(Answers answers) override;

public:

OrderIndication(QWidget\* parent = nullptr);

void read(QFile\* file) override;

void setWatchMistakesEnabled(bool) override;

};

#endif // ORDERINDICATION\_H

**3.7. SingleChoice.h**

#ifndef SINGLECHOICE\_H

#define SINGLECHOICE\_H

#include "Task.h"

// ОДИНОЧНЫЙ ВЫБОР

class SingleChoice : public Task {

private:

int32\_t rightAnswer; // Правильный ответ

int32\_t\* userAnswer; // Ответ пользователя

// Радиобаттоны вариантов ответов

QVector<QRadioButton\*> optionsRadioButtons;

protected:

QPair<bool, int> execute() override;

void plus() override;

void minus() override;

void switchAnswers(Answers answers) override;

public:

SingleChoice(QWidget \*parent = nullptr);

void read(QFile\* file) override;

void setWatchMistakesEnabled(bool) 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"

private:

Ui::TaskViewWidget \*ui;

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

};

#endif // TASKVIEWWIDGET\_H

**3.10. 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 "Task\TaskTypes\Task.h"

#include "Test.h"

#include "TestPassingWindow.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

// ГЛАВНОЕ ОКНО

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_openTestButton\_clicked();

void on\_startTestButton\_clicked();

void on\_name\_lineEdit\_textEdited(const QString &arg1);

void on\_surname\_lineEdit\_textEdited(const QString &arg1);

private:

Ui::MainWindow \*ui;

Test\* test; // Текущий тест

PassingTestWindow\* passingWindow; // Окно прохождения теста

bool testOpened; // Открыт ли тест

};

#endif // MAINWINDOW\_H

**3.11. Test.h**

#ifndef TEST\_H

#define TEST\_H

#include "Task\TaskTypes\Task.h"

#include "Task\TaskService.h"

#include "TestResult.h"

#include <QWidget>

#include <QFile>

#include <QUrl>

#include <QPair>

#include <QList>

// ТЕСТ

class Test : public QObject

{

Q\_OBJECT

private:

QVector<Task\*> taskList; // Задания

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

QUrl url; // Путь к тесту

public:

Test(QUrl);

// Добавить и удалить задание

void addTask(Task\*);

void removeTask(int);

// Открыть тест

void open();

// Активировать компоненты просмотра ошибок во всех заданиях

void setWatchMistakesEnabled(bool);

TestExecute execute(); // Результат прохождения теста

int size(); // Количество заданий

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

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

QUrl getUrl(); // Получить Url теста

// Время прохождения теста

void setPassingTime(bool, int);

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

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

};

#endif // TEST\_H

**3.12. TestPassingWindow.h**

#ifndef TESTPASSINGWINDOW\_H

#define TESTPASSINGWINDOW\_H

#include <QWidget>

#include <QTimer>

#include <QMessageBox>

#include "Task\TaskTypes\Task.h"

#include "Test.h"

#include "TestResultWindow.h"

namespace Ui {

class PassingTestWindow;

}

// ОКНО ПРОХОЖДЕНИЕ ТЕСТА

class PassingTestWindow : public QWidget

{

Q\_OBJECT

public:

explicit PassingTestWindow(Test\*, QPair<QString, QString>, QWidget \*parent = nullptr);

~PassingTestWindow();

signals:

void goToMainWindow();

private slots:

void on\_goToMainWindowButton\_clicked();

void on\_nextButton\_clicked();

void on\_prevButton\_clicked();

void on\_finishButton\_clicked();

void on\_showMistakes();

void updateTime(); // Обновить тикающее время

QString secondsToFormatStr(int); // Секунды в формат

private:

Ui::PassingTestWindow \*ui;

// Окно с результатом теста

TestResultWindow\* testResultWindow;

Test\* test; // Тест

QTimer\* timer; // Таймер

int currentTaskIndex; // Текущий индекс задания

int passedTime; // Прошедшее время (в секундах)

QPair<QString, QString> name; // Имя тестируемого

};

#endif // TESTPASSINGWINDOW\_H

**3.13. TestResult.h**

#ifndef TESTRESULT\_H

#define TESTRESULT\_H

#include <QPair>

#include <QString>

struct TestExecute

{

// Баллы <заработанные, возможные>

QPair<int,int> score = QPair<int,int>(0,0);

// Задания <верно решенные, всего>

QPair<int,int> tasks = QPair<int,int>(0,0);

};

struct TestResult

{

QPair<QString, QString> name; // Имя

TestExecute testExecute; // Подсчет теста

QString time; // Время прохождения

};

#endif // TESTRESULT\_H

**3.14. TestResultWindow.h**

#ifndef TESTRESULTWINDOW\_H

#define TESTRESULTWINDOW\_H

#include "TestResult.h"

#include <QWidget>

#include <QPair>

#include <QDialog>

namespace Ui {

class TestResultWindow;

}

class TestResultWindow : public QWidget

{

Q\_OBJECT

public:

explicit TestResultWindow(TestResult, QWidget \*parent = nullptr);

~TestResultWindow();

public slots:

void on\_goToMainWindowButton\_clicked();

void on\_watchMistakesButton\_clicked();

signals:

void goToMainWindow();

void showMistakes();

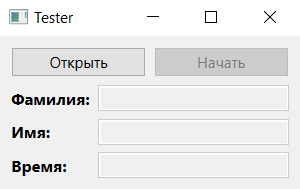
private:

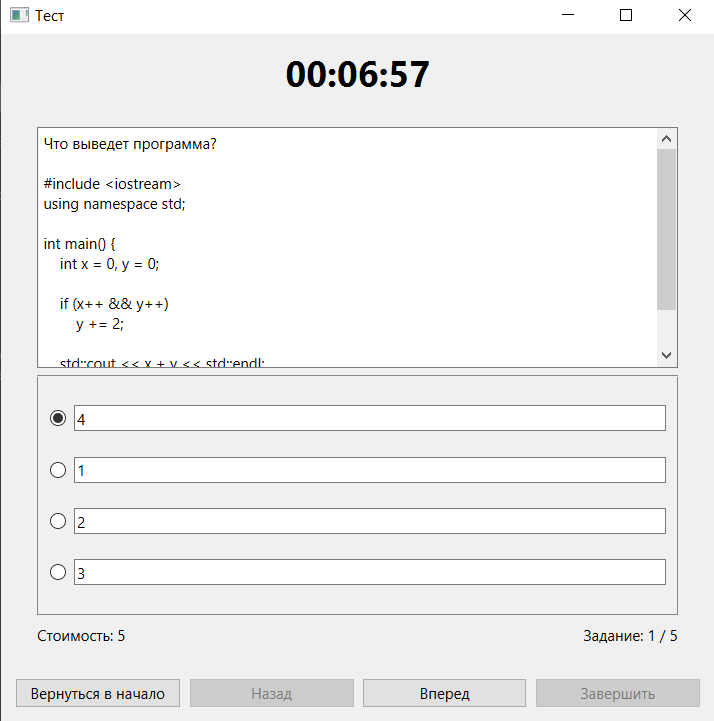
Ui::TestResultWindow \*ui;

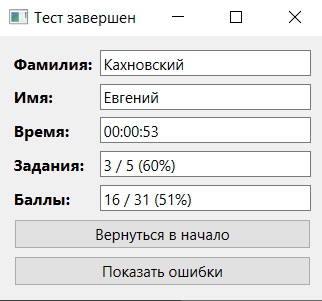
};

#endif // TESTRESULTWINDOW\_H

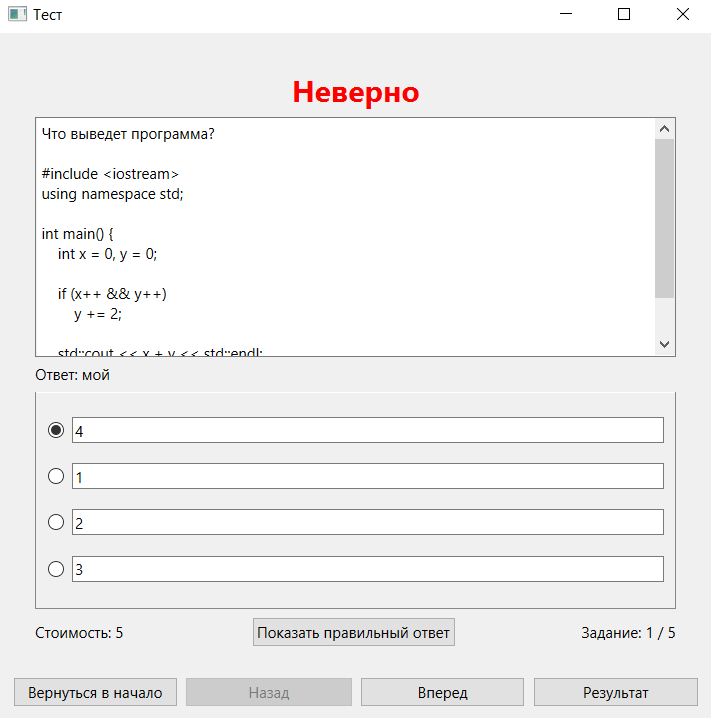
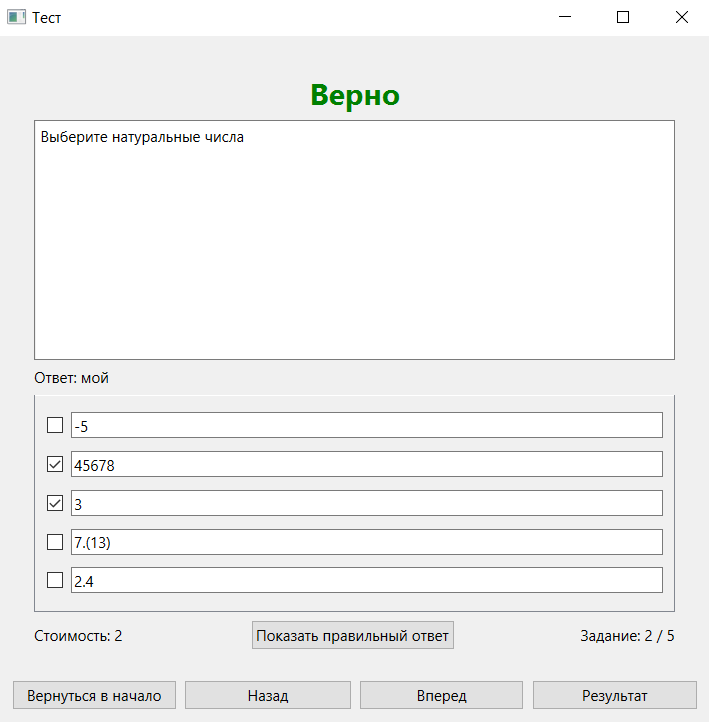
**4. РЕЗУЛЬТАТЫ**

 Интерфейс программы при ее запуске:

Процесс прохождения теста:

Результат теста:

Процесс просмотра ошибок:



**ЗАКЛЮЧЕНИЕ**

Был создан инструмент для прохождения тестов, созданных в конструкторе. Информирует пользователя о времени прохождения, количестве правильно выполненных заданий и полученных баллов в результате выполнения теста. Имеет возможность просмотра своих ошибок.

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)

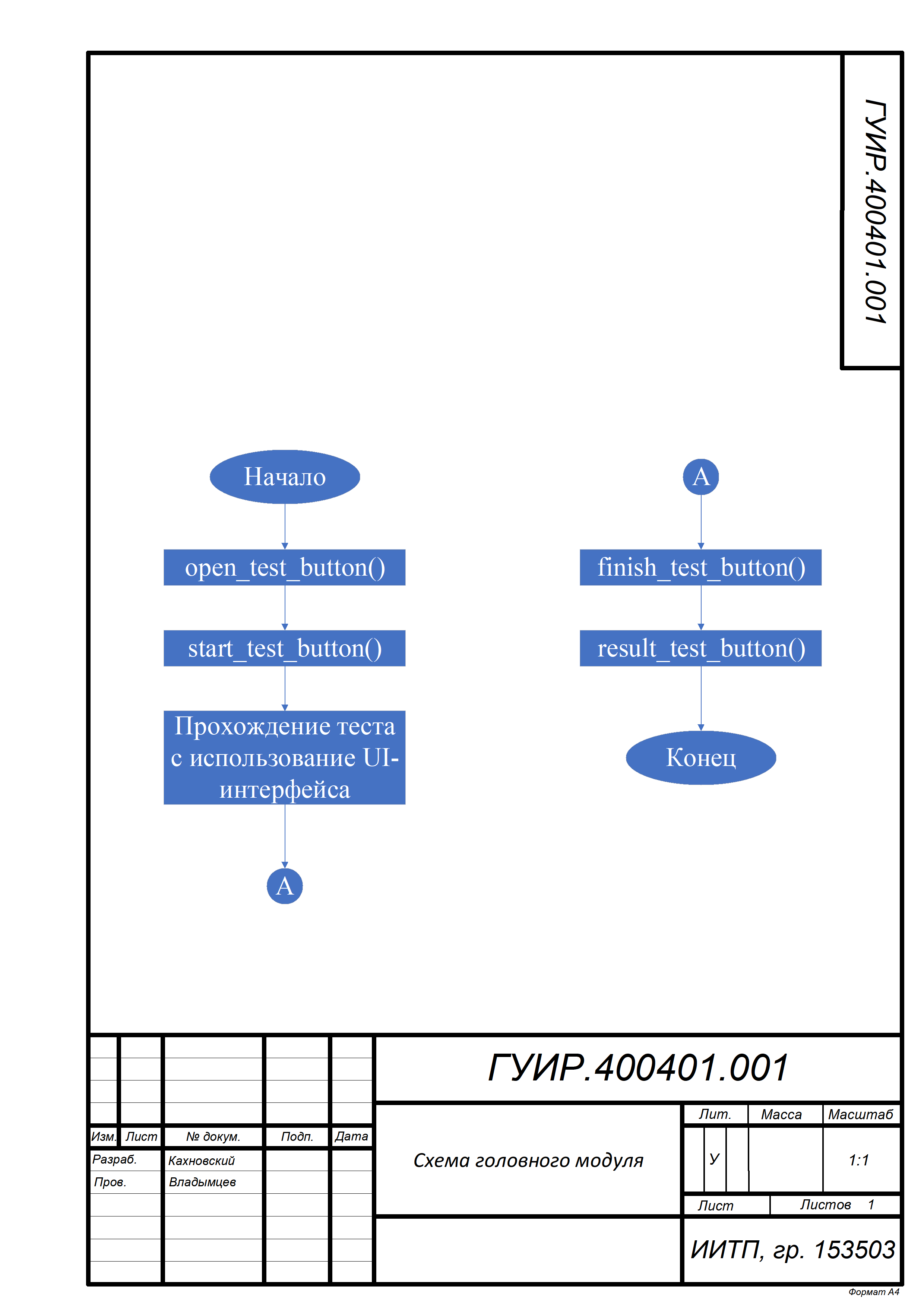
Qt Documentation [Электронный ресурс]. – Режим доступа: <https://doc.qt.io/>

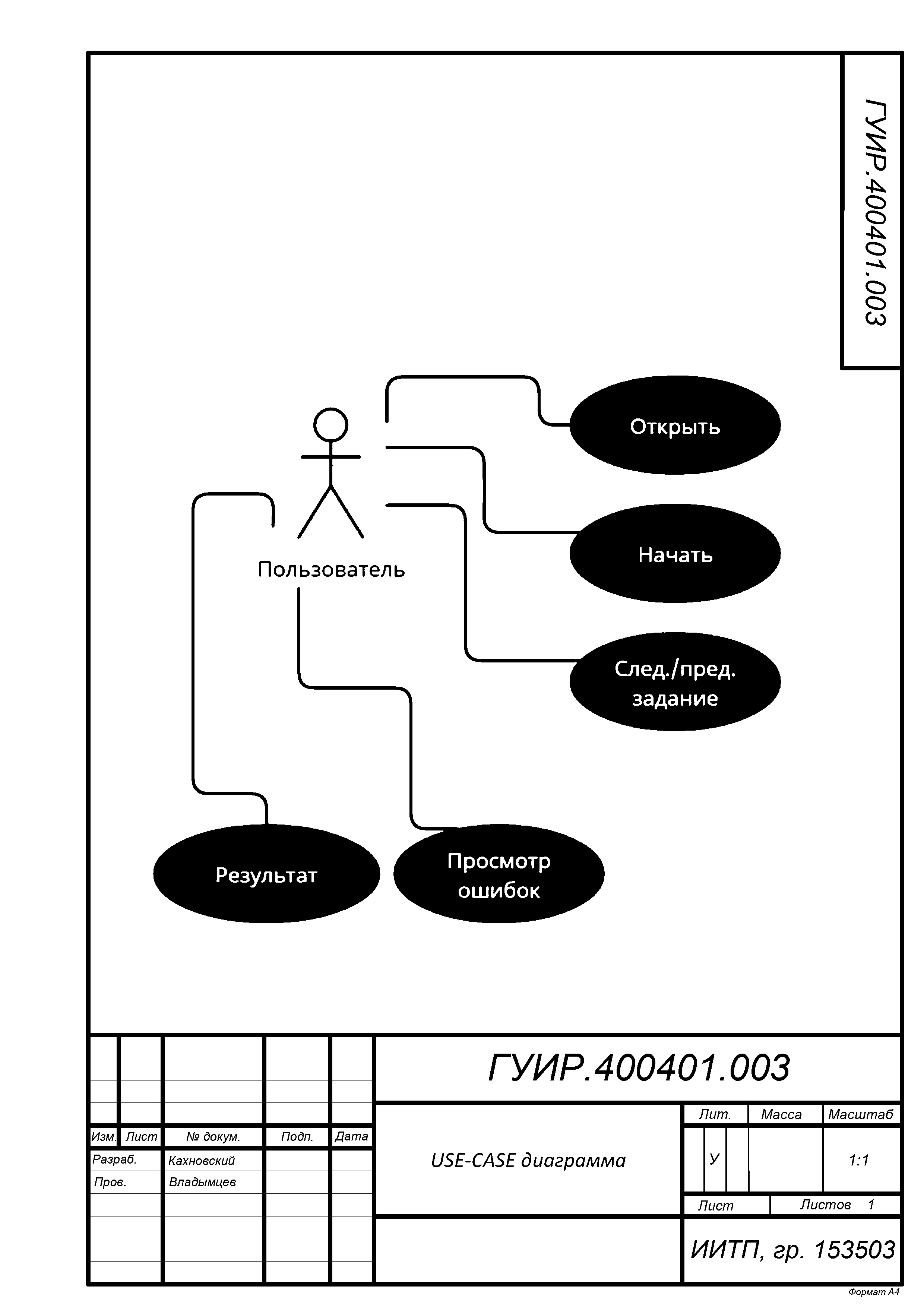
cppreference [Электронный ресурс]. — Режим доступа: <https://en.cppreference.com/w/>

**ПРИЛОЖЕНИЕ А.**

**

**ПРИЛОЖЕНИЕ Б.**



**ПРИЛОЖЕНИЕ В.**