Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

*К ЗАЩИТЕ ДОПУСТИТЬ*

*\_\_\_\_\_\_\_\_\_\_Е.В. Богдан*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

СИСТЕМА КОНТРОЛЯ ЗНАНИЙ

БГУИР КП 1–40 02 01 212 ПЗ

Студент: Зинович И.В.

Руководитель: Богдан Е.В.

Минск 2023

Учреждение образования

«Белорусский Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

––––––––––––––––––2023 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту     *Зиновичу Ивану Вячеславовичу –––––––––––––\_*

1. Тема проекта *Приложение «Система контроля знаний» –––––\_*

2. Срок сдачи студентом законченного проекта––––*15 декабря 2023 г.–––*

3. Исходные данные к проекту*- пЯзык программирования – С++, библиотека Qt, среда QT Creator------------------------------------------------------\_\_\_\_\_\_\_\_\_\_\_\_\_*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

1. Лист задания.

2. Введение.

3. Обзор литературы.

3.1. Обзор методов и алгоритмов решения поставленной задачи.

4. Функциональное проектирование.

4.1. Структура входных и выходных данных.

4.2. Разработка диаграммы классов.

4.3. Описание классов.

5. Разработка программных модулей.

5.1. Разработка схем алгоритмов (два важных метода).

5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов).

6. Результаты работы.

7. Заключение

8. Литература

9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов. ––––––––––––––––––––––––––––––*

*2. Схема алгоритма удаления странциы со слайда*

*3. Схема алгоритма чтения теста из файла*

6. Консультант по проекту (с обозначением разделов проекта) Е. В. Богдан

7. Дата выдачи задания –––––*15.09.2023г ––––––––––––––––––––––   –*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала к 06.10 – 15 %;––––––––––––––––––––––––––––\_\_+*

*разделы 2, 3 к 27.10 – 10 %;–––––––––––––––––––––––––––––––––––––    \_\_\_*

*разделы 4 к 06.11 –20 %;–––––––––––––––––––––––––––––––––––––––\_\_\_\_\_\_*

*разделы 5 к 20.11 – 35 %;––––––––––––––––––––––––––––––––––––––\_\_\_\_\_\_\_*

*разделы 6,7,8,9 к 27.11 – 10 %;––––––––––––––––––––––––––––––––––––––*

*оформление пояснительной записки и графического материала к 01.12.23 – 10 %*

*Защита курсового проекта с 21.12 по 28.12.23г.–––––––––––––––––––––––––*

РУКОВОДИТЕЛЬ *Богдан Е. В.*

(подпись)

Задание принял к исполнению *Зинович И.В.*

(дата и подпись студента)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc152809457)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc152809458)

[2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ 7](#_Toc152809459)

[3 ФУНКЦИАНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 8](#_Toc152809460)

[3.1 Диаграмма классов (смотреть приложение Б) 8](#_Toc152809461)

[3.2 Описание классов 8](#_Toc152809462)

[3.2.1 Класс для размещения динамического изображения 8](#_Toc152809463)

[3.2.2 Класс для создания или редактирования тестов 9](#_Toc152809464)

[3.2.3 Класс для генерирования диалогового окна, куда необходимо вводить название теста 11](#_Toc152809465)

[3.2.4 Класс для создания скролл-области каталога тестов 11](#_Toc152809466)

[3.2.5 Класс каталога тестов 11](#_Toc152809467)

[3.2.6 Класс для кнопки, поддерживающей нажатие shift + ПКМ, двойное нажатие ПКМ, нажатие ЛКМ 12](#_Toc152809468)

[3.2.7 Класс окна, в котором решают тест 13](#_Toc152809469)

[4 АЛГОРИТМЫ 15](#_Toc152809470)

[4.1 Алгоритмы по шагам 15](#_Toc152809471)

[4.1.1 Алгоритм по шагам для удаления слайда 15](#_Toc152809472)

[4.1.2 Алгоритм по шагам для чтения теста из файла 15](#_Toc152809473)

[5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ 17](#_Toc152809474)

[ЗАКЛЮЧЕНИЕ 24](#_Toc152809475)

[СПИСОК ЛИТЕРАТУРЫ 26](#_Toc152809476)

[ПРИЛОЖЕНИЕ А 27](#_Toc152809477)

[ПРИЛОЖЕНИЕ Б 28](#_Toc152809478)

[ПРИЛОЖЕНИЕ В 29](#_Toc152809479)

[ПРИЛОЖЕНИЕ Г 30](#_Toc152809480)

[ПРИЛОЖЕНИЕ Д 31](#_Toc152809481)

# ВВЕДЕНИЕ

Программирование на языках высокого уровня является ключевым элементом в разработке современного программного обеспечения. Одним из языков, пользующихся заслуженной популярностью среди разработчиков, является C++. Этот мощный и эффективный язык программирования высокого уровня обладает богатым функционалом, позволяющим создавать масштабируемые и производительные приложения.

Одним из инструментов, значительно обогащающих возможности C++, является библиотека Qt. Qt представляет собой кросс-платформенный фреймворк для разработки графических пользовательских интерфейсов, обладающий обширным инструментарием для работы с событиями, виджетами и многим другим. Одним из ключевых механизмов Qt являются сигналы и слоты, обеспечивающие эффективную коммуникацию между объектами в приложении. Механизм сигналов и слотов позволяет связывать события и обработчики событий в приложении, обеспечивая гибкость и масштабируемость кода.

Среда разработки Qt Designer предоставляет интуитивно понятный графический интерфейс для создания пользовательских интерфейсов на основе Qt. Она позволяет разработчикам быстро и эффективно создавать и настраивать графические интерфейсы приложений, упрощая процесс разработки.

Цель данной курсовой работы — изучить особенности языка C++, а также глубже понять и применить возможности библиотеки Qt, включая механизм сигналов и слотов, для разработки эффективных программных решений.

# 1 ПОСТАНОВКА ЗАДАЧИ

Целью данного проекта является разработка приложения, предоставляющего пользователю функциональные возможности по созданию, редактированию и управлению тестами. Приложение представляет собой удобное средство для организации тестовых структур с вопросами и возможностью добавления изображений к тестам.

Для разработки данного приложения мы будем использовать библиотеку Qt.

Библиотека Qt - это кроссплатформенный набор инструментов и библиотек для разработки программного обеспечения на C++. Она включает в себя широкий набор классов для создания графических интерфейсов, работы с файлами, сетями, базами данных и многих других задач.

Иерархия классов в Qt:

Qt предоставляет обширную иерархию классов. Основные модули включают классы для виджетов (`QWidget`, `QLabel`, `QPushButton`), графики (`QPainter`, `QPixmap`), работу с файлами (`QFile`, `QDir`), сетевые операции (`QTcpSocket`, `QUdpSocket`), а также структуры данных и контейнеры (`QList`, `QMap`, `QVector`), и многое другое.

Механизм сигналов и слотов:

Одна из ключевых особенностей Qt - это механизм сигналов и слотов. Он позволяет объектам взаимодействовать асинхронно. Сигналы генерируются объектами при определенных событиях, а слоты - это методы, которые реагируют на эти сигналы. Между собой объекты могут связываться, и при возникновении сигнала у одного объекта будут вызываться соответствующие слоты у другого.

Для "Системы контроля знаний студентов" этот механизм особенно полезен, поскольку позволяет реагировать на различные события в пользовательском интерфейсе (например, нажатие кнопки, изменение значения поля ввода) и обрабатывать их асинхронно, упрощая связь между различными компонентами вашего приложения.

Вы можете создавать свои собственные сигналы и слоты в классах, что позволяет гибко организовывать взаимодействие между различными частями вашего приложения.

Основные функции приложения включают:

1. Создание, удаление и редактирование тестов: Пользователь имеет возможность создавать новые тесты, удалять и редактировать уже существующие. Редактирование тестов происходит в специальном режиме, где можно добавлять вопросы и выбирать правильные варианты ответов. Также доступно добавление изображения к каждому тесту путем перетаскивания изображений на экран приложения в режиме создания тестов.

2. Динамическое сохранение результатов: Все созданные тесты и соответствующие им изображения записываются в специально созданную директорию. Каждый тест представлен текстовым файлом, содержащим его структуру и вопросы. Эти данные динамически обновляются в каталоге тестов, обеспечивая актуальность информации.

3. Управление приложением: Управление осуществляется при помощи интерфейса приложения и специальных горячих клавиш. Например, сочетание Shift + ЛКМ позволяет удалять выбранный тест, а двойное нажатие ПКМ открывает выбранный тест для редактирования из каталога тестов. Обычное нажатие ЛКМ выполняет действия согласно контексту интерфейса.

Особенности приложения:

Приложение разработано для операционной системы Windows. Для запуска на других устройствах необходимо пересобрать проект с учетом специфики и требований конкретной операционной системы.

Цель разработки приложения:

Целью данного проекта является предоставление пользователю интуитивно понятного и эффективного инструмента для создания и управления тестами с возможностью добавления изображений. Используя функциональность библиотеки Qt и языка C++, мы стремимся создать удобное приложение, которое обеспечит гибкость в работе с тестовыми структурами и обеспечит их хранение и доступность для пользователя.

# 2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ

Приложение использует следующую структуру данных для входных и выходных операций:

Таблица 1.1 – пример хранения данных в каталоге

|  |  |  |
| --- | --- | --- |
| Тип данных | Описание | Пример |
| Каталог тестов | Директория для хранения всех тестов | ..//alltests/ |
| Название теста | Директория, содержащая информацию о конкретном тесте | ..//alltests/test1/ |
| Текстовый файл | Файл с данными о тесте и его слайдах | ..//alltests/test1/test1.txt |
| Изображения | Папка, где хранятся изображения, используемые в тестах | ..//alltests/test1/image\_№n/ |

Таблица 1.2 – пример хранения одной строки в текстовом файле

|  |  |  |
| --- | --- | --- |
| Формат строки | Описание | Пример |
| Относительный путь к изображению | Путь к изображению, используемому в слайде теста | images/question1.png |
| Вопрос | Вопрос, представленный на данном слайде | What is the capital of France? |
| Количество вопросов | Общее количество вопросов в слайде | 4 |
| Номер правильного ответа | Номер правильного ответа в пределах общего количества вопросов (начиная с 1 как индекс) | 3 |
| Варианты ответов | Варианты ответов, разделенные символом /, их количество должно соответствовать указанному выше количеству вопросов | London/Paris/Berlin/Madrid |

# 3 ФУНКЦИАНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

## **3.1 Диаграмма классов (смотреть приложение Б)**

## **3.2 Описание классов**

### **3.2.1 Класс для размещения динамического изображения**

*class AspectRatioPixmapLabel : public QLabel*

Q\_OBJECT - макрос используется в классах QT для связывания сигналов и слотов.

public:

explicit AspectRatioPixmapLabel(QWidget \*parent = 0); - конструктор

virtual int heightForWidth( int width ) const; - функция для масштабирования

virtual QSize sizeHint() const; - перегрузка виртуальной функции

QPixmap scaledPixmap() const;

bool is\_changed = false; - информация, изменена ли картинка

public slots:

void setPixmap ( const QPixmap & ); - функция для установки изображения

void resizeEvent(QResizeEvent \*); - событие для масштабирования

void setClicable(bool x); - можем ли мы поменять динамически изображение

void clearImage(); - удалить изображение

QString getPath\_to\_IMG(); - получить директорию изображения

void setCurrentIMG(QString path\_to\_IMG); - установить конкретное изображение

private slots:

void dragEnterEvent(QDragEnterEvent \*event); - событие захвата

void dropEvent(QDropEvent \*event); - функция в Qt используется для обработки события "падения" (drop) объектов на виджет

void mousePressEvent(QMouseEvent\* e); - функция в Qt, которая вызывается при нажатии кнопки мыши на виджете.

private:

bool is\_clicable = false; - используется для отслеживания состояния, доступности клика на что-то в приложении.

QPixmap pix; - хранение изображение

QString path\_to\_IMG = "NONE"; - строка для хранения пути к изображению.

### **3.2.2 Класс для создания или редактирования тестов**

namespace Ui {

class CreatingTestForm; - отдельное пространство имен для копии класса Ui

}

*class CreatingTestForm : public QWidget*

Q\_OBJECT

public:

explicit CreatingTestForm(QWidget \*parent = nullptr); - конструктор класса

void set\_path\_to\_test(const QString& path); - путь до папки с тестом

void set\_name\_of\_test(const QString& name); - название текстового файла

~CreatingTestForm(); - деструктор

public slots:

void generate\_creating\_form(); - генерирует страницу для создания класса

void setCreatingFormForEditSlide(); - устанавлиет настройки страницы для создания страницы, для редактирования готового теста

private slots:

void cancelPushButton\_clicked(); - обработка события нажатия на кнопки Cancel из основного окна редактирования

void on\_editQuestionsPushButton\_clicked(); - обработка события кнопки Edit questions из основного окна редактирования

void on\_cancelPushButton\_2\_clicked(); - обработка события кнопки Cancel из addAnswersWidget ( QWidget в StackedWidgets )

void on\_addQuestionPushButton\_clicked(); - обработка события нажатия на кнопку Edit questions

void onRemovedButton(); - обработка события для удаления одного из вариантов добавленного ответа

void on\_savePushButton\_2\_clicked(); - обработка события сохранения вариантов ответа

void chooseRightAnswer(); - обработка события правильного варианта ответа

void on\_nextSlideButton\_clicked(); - обработка события нажатия на Next Slide

void on\_prevPushButton\_clicked(); - обработка события нажатия Prev Slide

void on\_savePushButton\_clicked(); - обработка события нажатия кнопки Save из creatingMenuWidget

void on\_addImagePushButton\_clicked(); - обработка события нажатия кнопки Edit Image

void generate\_image(); - генерирует страницу с виджетом

void returnPushButton\_clicked(); - обработка нажатия кнопки OK в addImageWidget

bool is\_need\_to\_delete\_IMG(const QString& path\_to\_img); - булева переменная, которая показывает, нужно ли удалять изображение

void keyPressEvent(QKeyEvent\* e); - обработка событий клавиш клавиатуры

void removeCurrentPage(int index\_to\_remove); - удаление текущей страницы в редакторе тестов

private:

void create\_current\_info\_map(QMap<int, QMap<QString, QString>>& tmp\_map,int num\_of\_slide); - устанавливает текущую информацию о слайде в редакторе теста

bool is\_avaiable\_nextSlidePushButton(); - булева переменная, определяющая можем ли мы переключиться на следующий слайд

Qstring get\_number\_of\_right\_answer(const QList<QRightClickButton\*>& list); - получить номер правильного ответа

signals:

void open\_menuwindow(); - сигнал об открытии каталога с тестом из класса редактора тестов

private:

QString num\_of\_right\_answer = "NONE"; - переменная, хранящая номер верного ответа

QString info\_in\_text\_file = "%1|%2|%3|%4|%5"; - хранение текущей информации о тесте

Ui::CreatingTestForm \*ui; - класс, для хранения визуала, сгенерированной средой QT Designer

QList<QRightClickButton\*> buttons\_list; - хранения кнопок текущего слайда

AspectRatioPixmapLabel\* imgLabel; - класс для изображения

QMap<int, QMap<QString, QString>> myTest; - хранение информации в структуре QMap для хранения информации обо всем тесте

QString path\_to\_test; - путь к папке с тестом

QString name\_of\_test; - название текстового файла с текстом

QString path\_to\_IMG; - путь к изображению слайда

bool is\_chosen = false; - выбрана ли картинка

bool is\_next = false; - можем ли перейти на следующую страницу

bool is\_editor = false; - режим редактирования или создания теста

int num\_of\_slide = 0; - текущий номер слайда

### **3.2.3 Класс для генерирования диалогового окна, куда необходимо вводить название теста**

*class DialogForCreatingForm : public QDialog*

Q\_OBJECT

public:

DialogForCreatingForm( QWidget\* parent = 0 ); - конструктор класса

~DialogForCreatingForm(); - деструктор класса

QString getInput() const; - геттер для получения строки ввода

signals:

void open\_creating\_tests\_page(); - сигнал для открытия редактора тестов ( от данного класса )

private:

QLineEdit\* m\_edit; - класс для ввода строки

### **3.2.4 Класс для создания скролл-области каталога тестов**

*class HorizontalScrollArea : public QScrollArea*

QWidget \*contentWidget; - виджет, на который располагают контент

QGridLayout \*grid; - класс, для компоновки виджетов

int nRows; - счетчик номера строки

int nColumns; - счетчик номера колонны

public:

HorizontalScrollArea(int rows, int cols, QWidget \*parent = Q\_NULLPTR) – конструктор класса

void addWidget(QWidget \*w, int row, int col); - добавление виджета в наш класс

int columnCount() const; - счетчик колонны

private:

void adaptSize(); - адаптирование размера нашего класса

protected:

void resizeEvent(QResizeEvent \*event); - обработка события изменения размера окна

### **3.2.5 Класс каталога тестов**

namespace Ui {

class MenuWindow; - пространство имен для хранения класса Ui, сгенерированного QT Designer

}

*class MenuWindow : public QMainWindow*

Q\_OBJECT

public:

explicit MenuWindow(QWidget \*parent = nullptr); - конструктор класса

~MenuWindow(); - деструктор класса

private slots:

void dynamicPushButton\_clicked(); - обработка события нажатия на кнопку

void on\_testsPushButton\_clicked(); - обработка события нажатия на кнопку Tests

void creating\_Tests\_page(); - генерация каталога тестов

void on\_createPushButton\_clicked(); - обработка события на нажатие кнопки Create Test

void shift\_right\_dynamicPushButton\_clicked(); - обработка нажатия на shift + ПКМ ( удаление теста )

void double\_right\_dynamicPushButton\_clicked(); - обработка двойного нажатия правой кнопки мыши ( открытие редактора теста )

signals:

void open\_creating\_tests\_form(); - сигнал для открытия редактора теста

void open\_menuwindow(); - сигнал для открытия каталога тестов

private:

Ui::MenuWindow \*ui; - хранение интерфейса, сгенерированного средой QT Designer

HorizontalScrollArea\* scroll; - хранение скролл-области

TestWidget\* testWidget; - хранение самого теста

CreatingTestForm\* testForm; - хранение редактора теста

QVector<QRightClickButton\*> dynamic\_buttons; - хранение адресов динамически добавленных кнопок

### **3.2.6 Класс для кнопки, поддерживающей нажатие shift + ПКМ, двойное нажатие ПКМ, нажатие ЛКМ**

*class QRightClickButton : public QPushButton*

Q\_OBJECT

public:

explicit QRightClickButton(QWidget \*parent = 0); - конструктор класса

private slots:

void mousePressEvent(QMouseEvent \*e); - событие нажатия мыши

void mouseDoubleClickEvent(QMouseEvent\* e); - событие двойного нажатия мыши

signals:

void shiftRightClicked(); - сигнал нажатия shift + ПКМ

void leftClicked(); - сигнал нажатия ЛКМ

void doubleRightClicked(); - сигнал двойного нажатия ПКМ

### **3.2.7 Класс окна, в котором решают тест**

namespace Ui {

class TestWidget; - пространство имен для хранения класса Ui, сгенерированного QT Designer

}

*class TestWidget : public QWidget*

Q\_OBJECT

public:

explicit TestWidget(QWidget \*parent = nullptr); - конструктор класса

~TestWidget(); - деструктор класса

void add\_answers(); - функция добавления ответов

private slots:

void action\_if\_right\_answer(); - изменения цветов кнопок, в зависимости от выбора ответа

void delete\_dynamic\_buttons(); - удаление кнопок со слайда

void generate\_slide(); - генерация слайда теста

void add\_image\_on\_slide(); - добавление изображения на слайд

void change\_style\_if\_right\_answer(); - изменить стиль, в зависимости от выбора ответа ( слот )

void make\_buttons\_unclickable(); - сделать кнопки недоступными для выбора

public:

void setPath(); - установить путь к тесту

signals:

void activate\_generating\_slide(); - сигнал о генерации слайда

void open\_menuwindow(); - сигнал об открытии каталога теста

void continue\_test(); - сигнал для переключения на следующий слайд теста

protected:

void keyPressEvent(QKeyEvent \*e) override; - обработка события нажатия на клавиши клавитуры

private:

Ui::TestWidget \*ui; - хранение интерфейса, сгенерированного средой QT Designer

QMap<int, QMap<QString, QString>> myTest; - структура данных QMap, где хранится весь тест, записанный из файла

QVector<QPushButton\*> button\_vector; - хранение адресов динамически сгенерированных кнопок

QString path\_to\_test; - путь к тесту

HorizontalScrollArea\* scroll; - скролл-область

AspectRatioPixmapLabel\* imgLabel; - хранение изображения

bool is\_image = false; - добавлена ли картинка

bool can\_we\_continue = false; - можем ли перейти к следующему слайду

int num\_of\_slide = 0; - номер текущего слайда

int score = 0; - количество верных ответов

# 4 АЛГОРИТМЫ

## **4.1 Алгоритмы по шагам**

### **4.1.1 Алгоритм по шагам для удаления слайда**

Этот алгоритм представляет собой процесс удаления слайда из структуры данных теста. Он принимает переменные, включая информацию о текущем тесте, индексе слайда для удаления и номере текущего слайда.

1. Начало
2. Входные данные:

myTest – переменная класса CreatingTestForm, которая содержит в себе структуру данных QMap<int, QMap<QString, QString>, в которой находится информация о структуре текущего теста;

int index\_to\_remove – переменная, содержащая информацию о номере слайда, который необходимо удалить ( переданная в функцию ) ;

int num\_of\_slide – переменная класса CreatingTestForm c информацией о номере текущего слайда теста

1. Если index\_to\_remove >= 0 && index\_to\_remove <= размера myTest,

то переходим в п.4, иначе в п.8

1. Начало цикла for от index\_to\_remove до размера myTest – 1
2. Если не конец цикла, то присваиваем элемент структуры Qmap под индексом i элемент под индексом i+1 и переходим опять в п.4, иначе переходим в п.6
3. Если index\_to\_remove == размеру myTest – 1 и index\_to\_remove >= 0, то вычитаем единицу из num\_of\_slide (для пролистывания слайда на предыдущий )
4. Удаляем элемент myTest с ключом равным размер (myTest – 1)
5. Конец

### **4.1.2 Алгоритм по шагам для чтения теста из файла**

Этот код читает текстовый файл с данными о тестах и парсит его в QMap<int, QMap<QString, QString>>. В каждом внутреннем QMap ключами являются строки (например, "img", "question", "amount\_of\_questions", и так далее), а значениями - соответствующие строки из файла.

1. Создать QMap<int, QMap<QString, QString>> result для хранения данных о тесте.
2. Создать QMap<QString, QString> test\_slide для хранения данных каждого слайда теста.
3. Открыть файл по заданному пути `path` с помощью QFile.
4. Если файл не открылся, завершить приложение (в данном случае, использован QApplication::exit()).
5. Создать QTextStream для чтения данных из файла.
6. Инициализировать счётчик `counter\_first` для отслеживания порядкового номера слайда теста.
7. Прочитать файл построчно с помощью `readLine()` и пройти по каждой строке, т.е начало цикла while
8. Инициализировать счётчик `counter` для отслеживания текущей позиции в строке.
9. Пройти по каждому символу в строке.
10. Если символ - вертикальная черта `'|'`, выполнить следующее:

- В зависимости от значения `counter` добавить содержимое `word` в `test\_slide` под соответствующим ключом ("img", "question", и так далее).

- Очистить `word` для чтения следующего значения.

- Увеличить `counter` на 1.

1. Если символ не является вертикальной чертой, добавить его в `word`.
2. После завершения чтения строки добавить `test\_slide` в `result` по текущему `counter\_first`.
3. Увеличить `counter\_first` на 1 для следующего слайда теста.
4. Повторять шаги 7-13 для каждой строки в файле до достижения конца файла.
5. Вернуть `result` - QMap<int, QMap<QString, QString>>, содержащий данные о тесте.

# 5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

На рисунке 5.1 изображено главное окно приложения. Окно имеет стандартные элементы управления, характерные для интерфейсов приложений на платформе Windows.

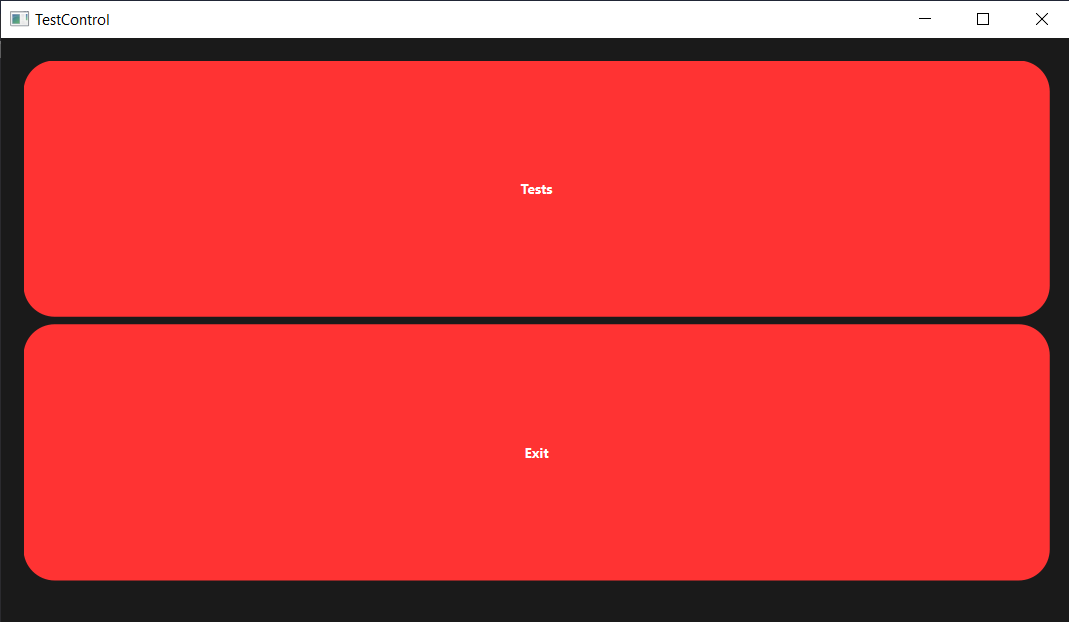


Рисунок 5.1 – главное окно приложения

На рисунке 5.2 изображен каталог тестов, представляющий собой интерфейсную часть приложения, отображающую доступные тесты для работы.

Каждый тест представлен в виде отдельного элемента, где пользователь имеет возможность взаимодействия с тестами посредством различных действий:

Выбор теста ЛКМ: Нажатие левой кнопкой мыши (ЛКМ) на тесте позволяет пользователю выбрать конкретный тест для дальнейшей работы с ним. Это действие запускает процесс загрузки теста для просмотра

Удаление теста Shift + ЛКМ: Путем нажатия и удержания клавиши Shift и последующего нажатия левой кнопки мыши (ЛКМ) пользователь может удалить выбранный тест из каталога. Это действие позволяет управлять списком тестов и удалить те, которые больше не требуются.

Вход в редактирование теста двойным нажатием ПКМ: Двойное нажатие правой кнопкой мыши (ПКМ) на выбранном тесте запускает режим редактирования этого теста. Это действие позволяет пользователю внести изменения в структуру или содержание теста.

Этот интерфейс обеспечивает удобное управление каталогом тестов и предоставляет пользователю гибкость в выполнении операций с тестами в зависимости от их потребностей.

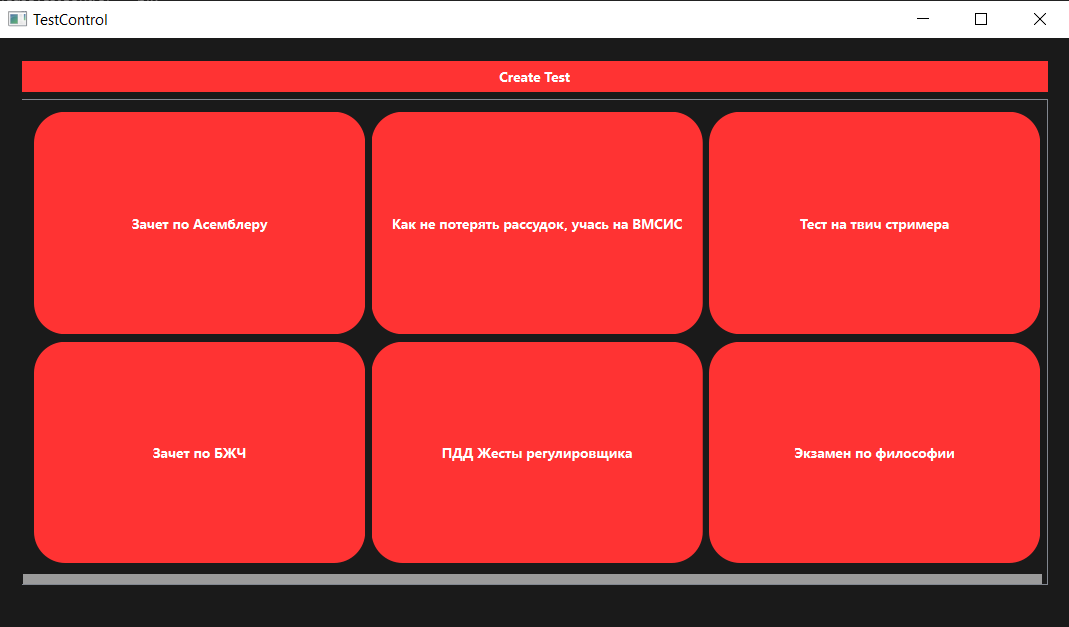


Рисунок 5.2 – каталог тестов

На рисунке 5.3 изображено диалоговое окно, которое появляется после нажатия на кнопку "Create Test". Это окно предоставляет пользователю возможность создать новый тест, заполнив необходимую информацию.

Диалоговое окно содержит следующие элементы:

Поле для ввода названия теста: Пользователь может ввести название нового теста в предусмотренное для этого текстовое поле.

Кнопки "Ok" и "Cancel": Кнопка "Ok" предназначена для подтверждения создания нового теста с введенным названием. При нажатии на нее данные о новом тесте будут сохранены. Кнопка "Cancel" предназначена для отмены создания теста и закрытия диалогового окна без сохранения введенных данных.

Это диалоговое окно обеспечивает удобный способ создания новых тестов, позволяя пользователям быстро указать название для нового теста и подтвердить или отменить создание.

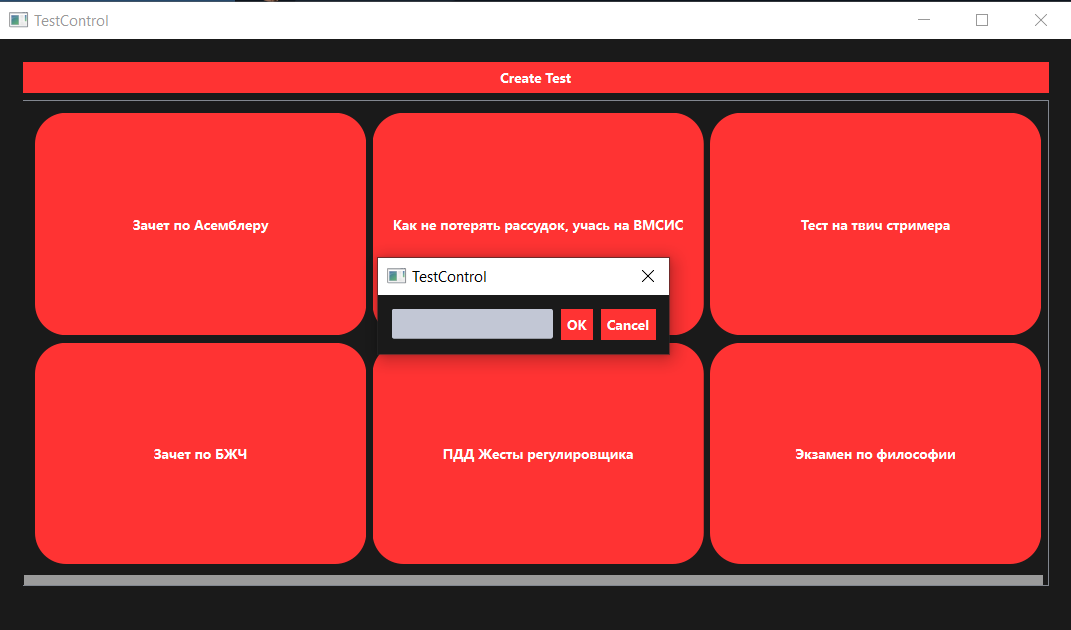


Рисунок 5.3 - диалоговое окно, которое появляется после нажатия на кнопку "Create Test"

На рисунке 5.4 изображен интерфейс редактора тестов, предоставляющий широкие возможности для редактирования содержимого тестовых структур. В данном интерфейсе представлены следующие элементы:

Кнопка "Edit Images" (Редактировать изображения): Позволяет пользователю добавлять новые изображения в слайды теста или удалять уже имеющиеся.

Кнопка "Edit Questions" (Редактировать вопросы): Позволяет пользователю добавлять вопросы к текущему слайду и выбирать правильные ответы на эти вопросы.

Кнопки "Save" (Сохранить) и "Cancel" (Отменить): Кнопка "Save" применяет внесенные изменения к текущему слайду теста. Кнопка "Cancel" отменяет все изменения и возвращает к предыдущему состоянию слайда.

Кнопки "Next Slide" (Следующий слайд) и "Prev Slide" (Предыдущий слайд):Позволяют переключаться между слайдами теста для редактирования следующего или предыдущего слайда соответственно.

Этот интерфейс обеспечивает удобный инструментарий для добавления изображений, вопросов, управления ответами и перехода между слайдами, предоставляя пользователю полный контроль над созданием и редактированием содержимого тестов.

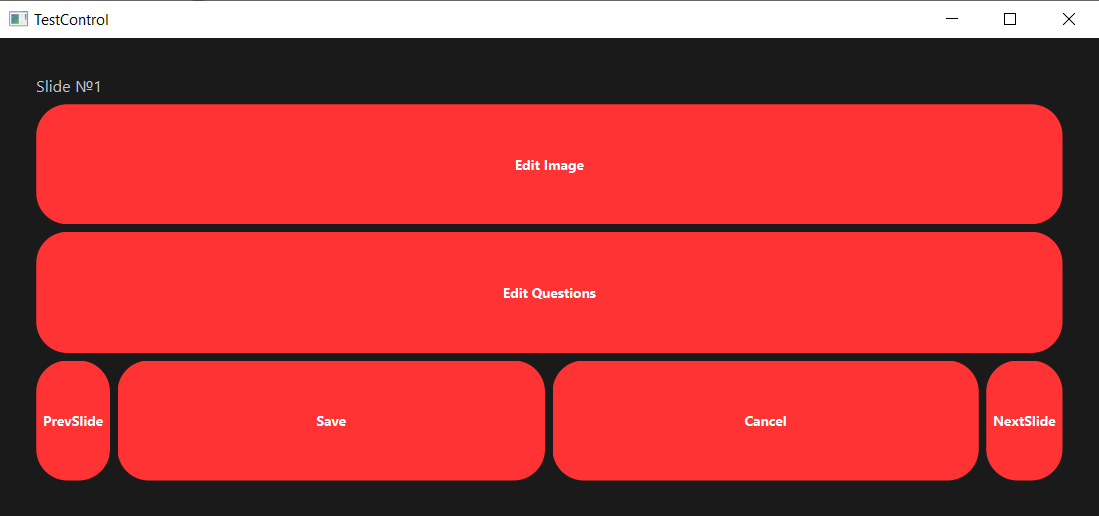


Рисунок 5.4 – редактор тестов

На рисунке 5.5 изображен процесс добавления изображения на слайд теста с использованием функционала перетаскивания.

Этот интерфейс включает в себя следующие элементы:

Место для перетаскивания изображения: Область окна, предназначенная для захвата и перетаскивания изображения. Пользователь может перетащить изображение с рабочего стола или другого окна прямо на это место для добавления на слайд.

Кнопка "OK": Позволяет пользователю подтвердить добавление изображения на слайд и сохранить изменения. После нажатия на эту кнопку изменения применяются, и пользователь возвращается к окну с редактором теста.

Этот интерфейс обеспечивает удобный способ добавления изображений на слайды теста с помощью простого перетаскивания. Кнопка

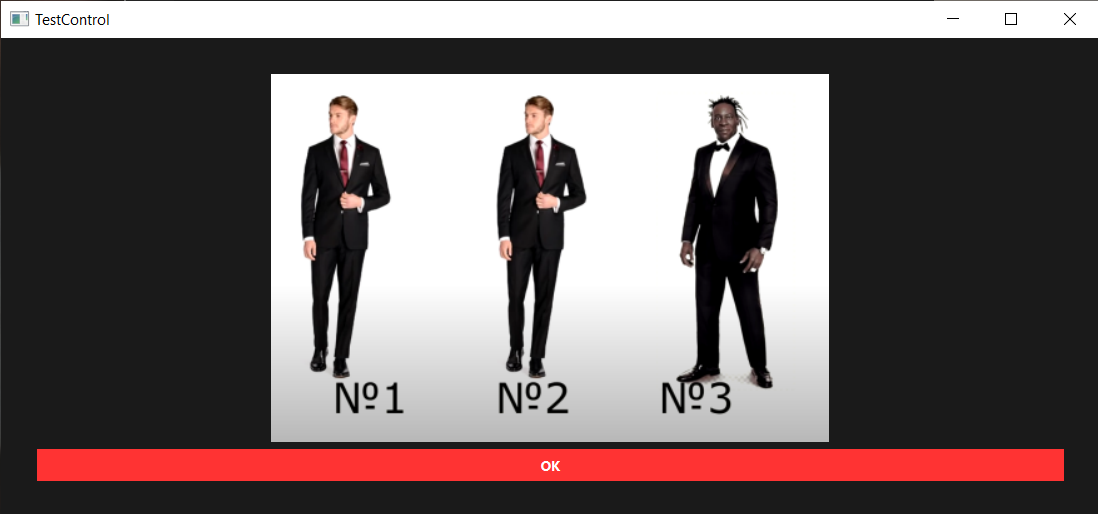


Рисунок 5.5 – окно с возможностью добавления картинки на слайд теста

На рисунке 5.6 представлено окно, предназначенное для добавления вопросов с вариантами ответов в тестовую структуру.

Интерфейс данного окна включает следующие элементы:

Первая строка для ввода вопроса: Здесь пользователь вводит текст вопроса для теста.

Вторая строка для ввода варианта ответа: Позволяет пользователю вводить текст варианта ответа на вопрос.

Кнопка "Add Question" (Добавить вопрос): При нажатии на эту кнопку введенный вопрос и вариант ответа добавляются в виде кнопки на экран. Эти кнопки представляют отдельные вопросы с вариантами ответов.

Выбор правильного ответа: Путем клика на кнопку сформулированного вопроса, пользователь подсвечивает выбранный вариант зеленым цветом, указывая его как правильный ответ. Повторное нажатие на эту кнопку снимает выделение, если пользователь передумал выбирать этот вариант.

Удаление вопроса Shift + ЛКМ: Нажатие и удержание клавиши Shift в сочетании с нажатием левой кнопки мыши (ЛКМ) на кнопке с вопросом позволяет пользователю удалить выбранный вопрос из тестовой структуры.

Кнопки "Save" (Сохранить) и "Cancel" (Отменить): Кнопка "Save" сохраняет введенные вопросы и ответы. Кнопка "Cancel" отменяет все изменения, сделанные в текущем окне.

Этот интерфейс обеспечивает удобный механизм для создания, редактирования и управления вопросами с вариантами ответов, предоставляя возможности добавления, удаления и выбора правильного ответа для каждого вопроса в тестовой структуре.



Рисунок 5.6 – окно с возможностью добавления вопроса, вариантов ответа и выбором правильного ответа

На рисунке 5.7 показано окно, предназначенное для прохождения теста. Интерфейс этого окна состоит из следующих элементов:

Изображение слева: В случае наличия изображения на текущем слайде теста оно отображается слева от вопроса и вариантов ответов.

Варианты ответов справа: Варианты ответов на поставленный вопрос представлены справа от изображения (если оно есть). Пользователь может выбрать один из вариантов ответа.

Подсветка вариантов ответов: Правильный вариант ответа подсвечивается зеленым цветом, остальные варианты отображаются серым цветом, а если пользователь выбрал неправильный вариант, то он подсвечивается красным цветом.

Управление тестом: Для выхода из теста пользователь может нажать на клавишу Esc. Чтобы перейти на следующий слайд после ответа на текущий вопрос, необходимо выбрать один из вариантов ответа и нажать кнопку Enter.

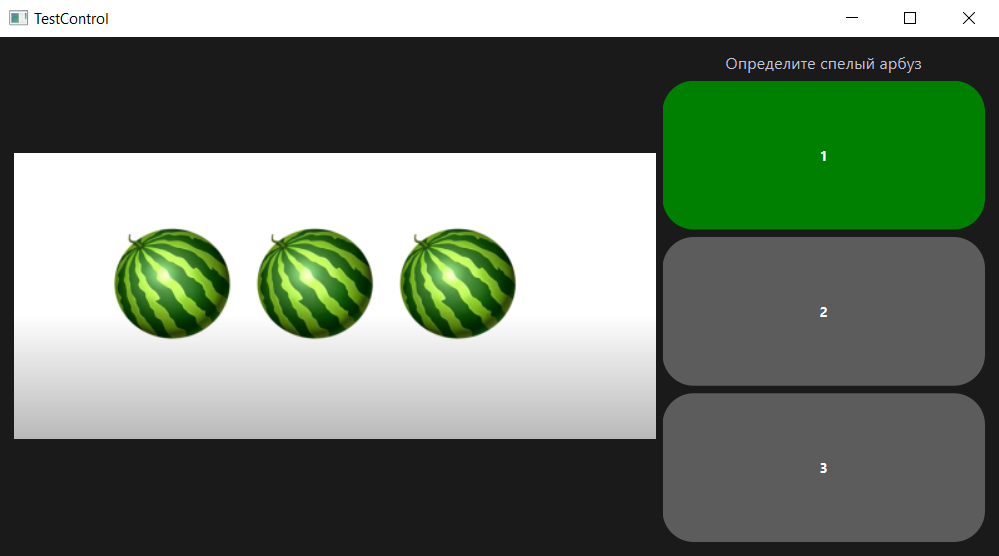


Рисунок 5.7 – окно для решения выбранного теста

# ЗАКЛЮЧЕНИЕ

Проведение тестирования и контроль знаний через приложение "Система контроля знаний" является важной составляющей в образовательном процессе. Тесты позволяют проверить уровень усвоения материала и помогают узнать, насколько глубоко и полно студент овладел изучаемой темой.

В ходе разработки "Системы контроля знаний" были использованы передовые технологии и возможности библиотеки Qt, что позволило создать эффективное приложение для управления тестами и контролем знаний.

Используемые технологии и инструменты:

В процессе разработки были использованы классы и функционал библиотеки Qt, такие как `QWidget`, `QString`, `QDir`, `QFile`, `QMap`, `QLineEdit`, `QList`, `QMessageBox`, `QPixmap`, `QHBoxLayout`, `QFileInfo`, `QDialog`, `QLayout`, `QPushButton`, `QKeyEvent` и многие другие. Эти инструменты обеспечили создание многооконного интерфейса, возможность хранения тестов в каталогах, а также обработку событий клавиатуры и другие функциональные возможности.

Работа с механизмом слотов и сигналов:

Одной из ключевых частей разработки было использование механизма слотов и сигналов библиотеки Qt. Этот мощный инструмент позволил эффективно организовать взаимодействие между элементами интерфейса и логикой приложения, обеспечивая отзывчивость и плавную работу системы контроля знаний.

Польза от приложения "Система контроля знаний":

Приложение создано с целью облегчения процесса создания, редактирования и управления тестами. Его интерфейс предоставляет удобное и интуитивно понятное окружение для пользователей, позволяя добавлять изображения, создавать вопросы с вариантами ответов, проходить тесты и анализировать результаты.

Интерфейс и хранение тестов:

Разработанный интерфейс представляет собой многооконную систему с интуитивными элементами управления. Тесты хранятся в каталогах, что обеспечивает удобное структурирование и доступ к ним.

Применение принципов ООП в разработке приложения:

Использование принципов ООП, таких как наследование, позволило создать собственные классы, унаследованные от базовых классов библиотеки Qt. Это позволило эффективно расширять функционал и адаптировать приложение для конкретных задач. Виртуальные методы, в свою очередь, обеспечили гибкость в работе с абстрактными классами Qt, позволяя создавать адаптивные решения.

Возможности пересборки под разные платформы:

Приложение успешно собрано для операционной системы Windows, но благодаря правильной настройке CMake файла, можно осуществить пересборку под мобильные устройства и MacOS. Это демонстрирует гибкость и переносимость разработанного приложения на различные платформы.

Значение знаний ООП для будущих задач:

Знания объектно-ориентированного программирования (ООП) являются ключевыми для дальнейших профессиональных задач. Они позволяют разрабатывать более структурированный и расширяемый код, а также более легко адаптировать и модифицировать приложения под различные потребности и платформы. Полученный опыт работы с ООП и библиотекой Qt будет ценным активом в дальнейшей разработке программного обеспечения.

"Система контроля знаний" является удобным инструментом для создания и прохождения тестов, обеспечивая эффективное управление знаниями и контролем обучения. Используемые технологии Qt, механизм слотов и сигналов, а также созданный интерфейс делают это приложение мощным инструментом для образования и тестирования знаний.

# СПИСОК ЛИТЕРАТУРЫ

[1] "Объектно-ориентированное программирование на С++" Бьярн Страуструп

[2] "Язык программирования С++" Герберт Шилдт

[3] "C++ Primer" Липман, Лажойе, Му, Хопкинс

[4] "Алгоритмы. Построение и анализ" Кормен, Лейзерсон, Ривест, Штайн

[5] "Введение в алгоритмы" Кормен, Лейзерсон, Ривест, Штайн

[6] "Алгоритмы на C++" Роберт Седжвик, Кевин Уэйн

[7] C++: эффективное программирование. 55 способов улучшения структуры программ и стиля кода" Scott Meyers

[8] "Алгоритмы. Построение и анализ" Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

[9] "Структуры данных и алгоритмы в C++" Robert Lafore

[10] "C++ for Game Programmers" Noel Llopis

# ПРИЛОЖЕНИЕ А

(Обязательное)

Листинг кода

# ПРИЛОЖЕНИЕ Б

(Обязательное)

Диаграмма классов

# ПРИЛОЖЕНИЕ В

(Обязательное)

Блок схема алгоритма для удаления слайда

# ПРИЛОЖЕНИЕ Г

(Обязательное)

Блок схема алгоритма для чтения теста из файла

# ПРИЛОЖЕНИЕ Д

(Обязательное)

Ведомость документов