СОДЕРЖАНИЕ

1. Реферта……………………………………………………………….2
   1. Общее число страниц, количество таблиц, рисунков, количество разделов
   2. Краткий обзор содержания основных разделов – расчетно пояснительной записки
2. Введение……………………………………………………………...3
3. Теоретическое введение (раздел 1)…………………………………….4
4. Проектная часть (раздел 2)…………………………………………....11
   1. Постановка задачи
   2. Проектное решение
   3. Описание структуры интерфейса пользователя
5. Эксперементальная часть (раздел 3)…………………………………22
   1. Постановка задачи или описание подхода к тестированию разработанной программы или типа, или приложения использующего тип………………………………………….22-23
   2. Тестирование……………………………………………...…24-30
6. Список литературы ……………………………………………………33
7. Приложение ……………………………………………………………34

РЕФЕРАТ

Общее число страниц: 34

Количество таблиц: 2

Рисунков: 33

Количество разделов: 3

Краткий обзор содержания разделов расчетно пояснительной записки

Язык программирования C++ является одним из самых популярных и широко используемых для разработки GUI приложений. В данной работе нам предстоит разработать приложение с графическим интерфейсом, работать с файловыми потоками ввода/вывода и различными функциями стандартных библиотек С++. Объем работы обеспечит закрепление и углубление практических навыков по программированию на языке С++.

ВВЕДЕНИЕ

Цель данной курсовой работы состоит в разработке GUI приложения с использованиаем языка программирования С++ и знаний полученных на лекциях по ОПП .

Реализация программной части – на языке С++ в среде Qt.

В ходе разработку были спроектированы модули, реализующие функции, необходимые для работы с объектами класса – массива и классов – вариантов.

Данная курсовая работа преследует такие цели, как:

1. Обучение элементам технологии создания программного продукта
2. Использование знаний по технологии структурного программированя с использованием современных средств модуль и объект
3. Обучение качественному оформлению документации на программное обеспечение

О своевременности и актуальности рассматриваемой проблемы говорит тот факт, что большую часть своего времени люди тратят на оформление различной документации, на пересчет различных банковскиъ операций в банке в ручную. Создание GUI приложения позволяет автоматизировать некоторые дейсвтия человека и решить поставленную задачу в кротчайшие сроки .

Дейсвительно, GUI приложения намного упрощают жизнь человека и позволяют сократить время разработки какого-либо программного продукта или же во многом помогают в решении поставленной заадачи, на которую без нее ушло бы дастаточно много времени .

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

Объектно-ориентированное проектирование

Объектно-ориентированное проектирование — это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления как логической и физической, так статической и динамической моделей проектируемой системы.

В данном определении содержатся две важные части:

1) объектно-ориентированное проектирование ведет к объектно- ориентированной декомпозиции;

2) используется многообразие приемов представления моделей, отражающих логическую (структуры классов и объектов) и физическую (архитектура модулей и процессов) структуру системы.

Именно поддержка объектно-ориентированной декомпозиции отличает объектно-ориентированное проектирование от структурного проектирования.

Подходы к сбору и анализу требований к ПО

По мере изучения поставленных целей, я производил декомпозицию задачи, что помогло быстрее разобраться над чем мне предстоит работать. В начале, для решения задач, я использовал среду разработки VisualStudio2017 от Microsoft, но вскоре столкнулся с проблемой, которая не позволяла воспроизводить , созданную мной , программу на устройствах не относящихся к Windows. Данная проблема ограничивала круг пользователей, которые могли бы воспользоваться моим приложением. Так же одной из проблем данной среды разработки была сложность к переносу данного приложения на мобильные устройства. В качестве решения было принято создать данное приложение в среде разработки Qt Creator 4.5.2. (см. Рисунок 1)

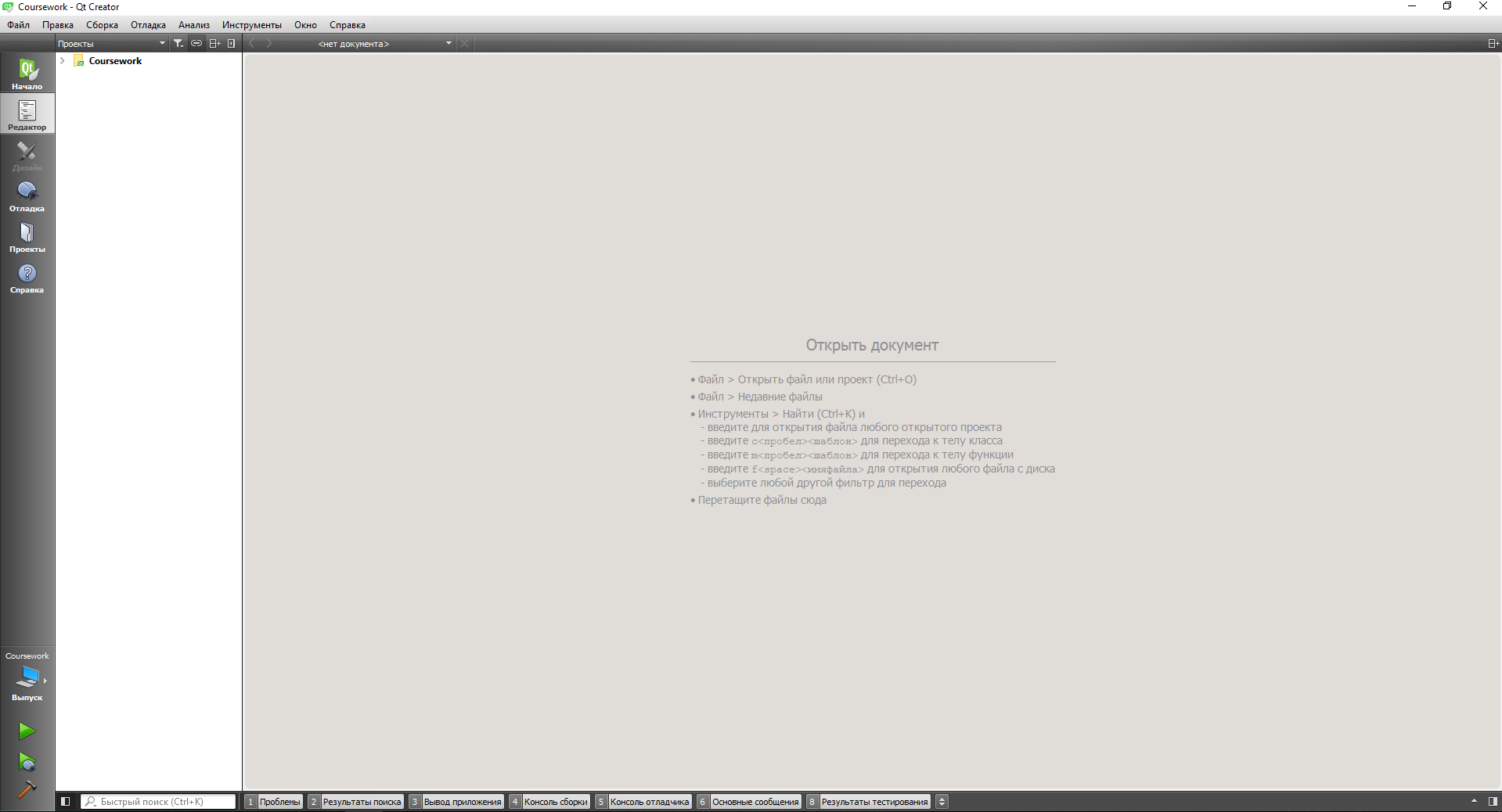


Рисунок 1

Qt- кроссплатформенный фреймворк, то есть он предоставляет возможность воспроизведения одного и того же кода под разные устройства без изменения их программистом в ручную, в следствии чего мой выбор закончился на данной среде разработки. При разработке приложения использовались библеотеки среды разработки Qt: QDebug, QModelindex, QMessageBox, QTableWidget, QFile, QString, QTextEdit, QTextStream, QUrl, QDesktopServices, и QApplication.

Предметной областью задания является «Формирование расписания экзаменов группы». Реальная база данных «расписаний экзаменов группы» должна содержать большое количество полей данных, но в данной работе были выбраны 2, с моей точки зрения, наиболее значимых:

1. Группа (Groupe.txt)
2. Аудитория (Auditor.txt)

Методы создания интерфейса

Приложение содержит в себе главное окно предоставляющая возможность пользователю выбрать нужную ему группу. Выбор группы осуществляется с помощью поля ввода LineEdit и нажатием на кнопку выбрать (PushButton).

В левом нижнем углу главного окна находится ссылка на «Создателя проекта», при нажатии на которую , пользователя перебрасывает на страницу создателя приложения в социальной сети VK.

Также для пользователя приложение предоставляет 2 способа выйти из приложения:

1. Нажатием в MenuBar на «Файл» и выборе в выпадающем окне кнопки «Выйти»
2. Нажатием значка крестика в правом верхнем углу приложения

Для создания интерфейса были использованы использованы инструменты среды разработки (см. Рисунок 2)

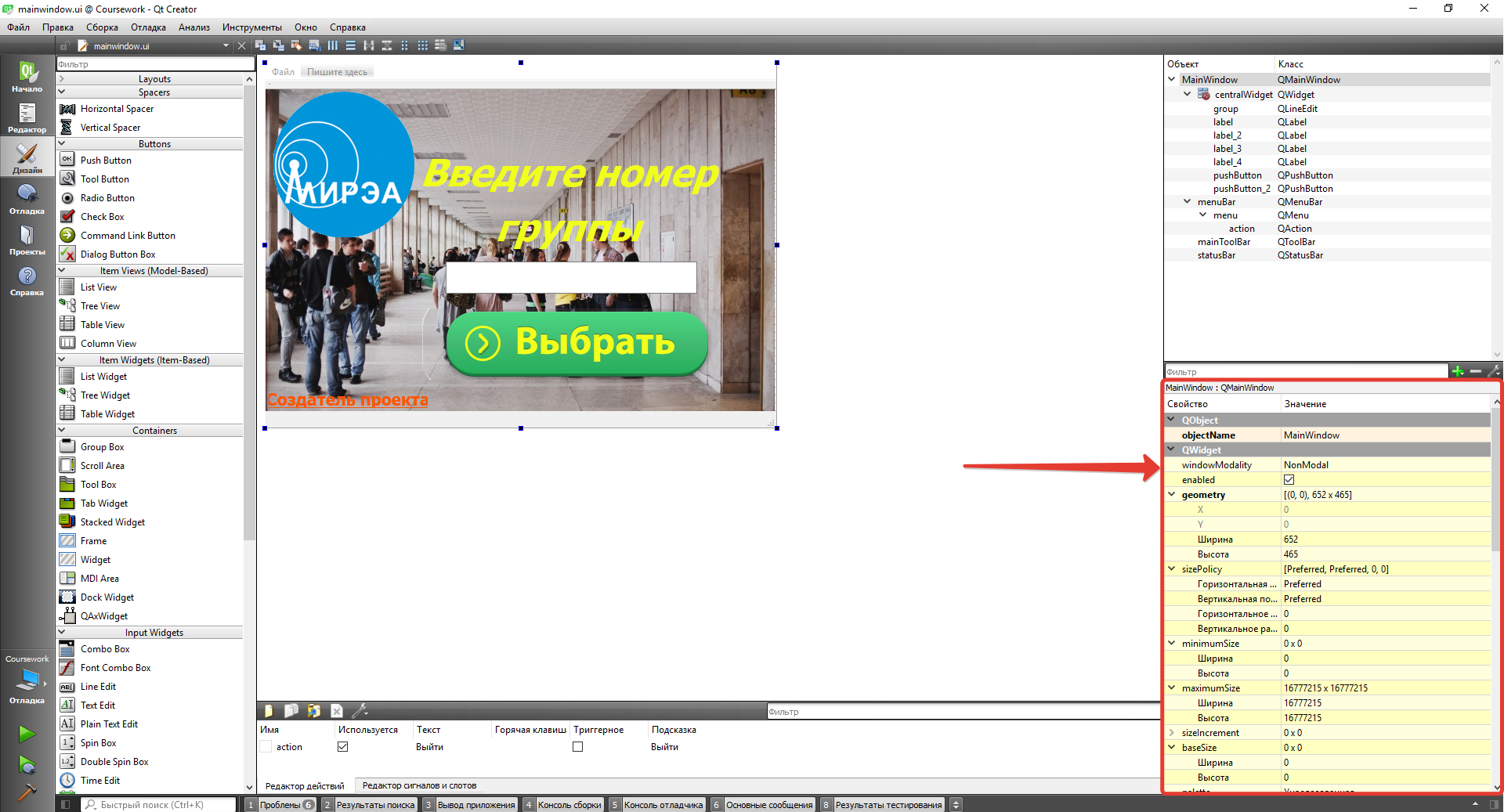


Рисунок 2

На втором окне, которое отрывается после ввода существующей в БД группы, появляется форма с пустыми таблицами расписаний экзаменов. Каждая таблица соответствует отдельному номеру недели экзаменационной сессии. Числа в 1 недели нумеруются с 28 по 31 мая, а все остальные дни нумеруются днями экзаменационной сессии в июне.

В заголовке окна записывается номер группы который блы введен при запуске приложения, дабы пользователь не забывал к какой группе данное расписание относится.(см. Таблица 1)

Причиной создания именно такого вида таблиц послужило легкость и доступность всей информации в целом. Благодаря данному представлению таблиц, пользователю не нужно пролистывать страницы, дабы узнать расписание на той или иной день, теперь же у него все будет как на ладони.

Таблица 1 – Расписание экзаменов за 1 неделю



Стоит заметить, что расписание экзаменов выводится в таблицу не сразу, а лишь только по нажатию на кнопку. Это было сделано с целью дать пользователю для начала разобраться в интерфейсе, а уже потом использовать его по своему усмотрению. (см. Рисунок 3)



Рисунок 3

Если вдруг у пользователя окажется желание разузнать для своих целей расписание экзаменов другой группы, то специально для того создана кнопка в этом окне с названием «Сменить группу», после нажатия на которую пользователя вернут в главное окно для ввода нужной группы.(см. Рисунок 4)



Рисунок 4

В университете не исключены случаи, когда аудитории, в которых по плану должны были проводится экзамены, вдруг оказываются недоступны. Причины бывают разные, но в частности это происходит во время внепланнового ремонта аудитории. Чтобы как можно быстрее найти аудиторию, в которой не проходит в данный момент экзамен, была предоставлена возможность пользователю найти не занятую аудиторию в заданный момент времени. Для использования данной возможности достаточно в данном окне выбрать интересующий вас промежуток времени и нажать на кнопку «Найти свободную аудиторию в указанное время». (см. Рисунок 4)



Рисунок 4

После нажатия на данную кнопку высветится новое окно со списком пыстых аудиторий. Учитывая , что пользователь явно не будет использовать данную кнопку только ради разового использования, была предоставлена возмонжость учитывать изменения , который производил пользователь в окне с расписанием экзаменов и на основе его, после нажатия на кнопку, инфомация о пустых аудиториях подстраивается под эти изменения. (см. Рисунок 5)



Рисунок 5

Для удобства пользователя, помимо 3 окна, информация о пустых аудиториях еще обновляется и в окне расписаний экзаменов. Данные аудитории подствечиваются синим цветом, чтобы пользователю было с легкостью их определить. (см. Рисунок 6)

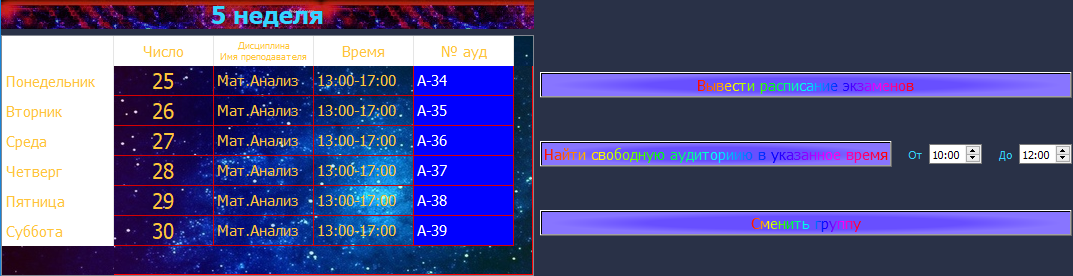


Рисунок 6

ПРОЕКТНАЯ ЧАСТЬ

Задачи курсовой работы: Разработать систему Формирование расписания экзаменов группы.

Приложение должно содержать в себе следующий функционал:

1. Изменять расписание экзаменов
2. Выбор расписания выбранной группы
3. Определение свободной аудитории для задданного промежутка времени

Для решения задачи были созданы следующие функции:

void Menu::setTableText(QString text) - *для* *вставвки* *текста* *с* *1* *формы*

int new\_copy\_time(QString copy\_time, int i, int j) - *возвращает* *значение* *вермени* *(часы* *или* *минуты)*

void Menu::SetInformationTable() - *установление* *информации* *в* *таблицы*

void Menu::Line\_memorization(QString str)- *запоминание* *строки* *в* *массив*

void Menu::Find\_exame(int i, int start\_examen\_min, int start\_examen\_heur, int end\_examen\_min, int to\_time\_min) - *находит* *свободную* *аудиторию* *в* *заданное* *время*

void Menu::Update\_Color\_Table(QTableWidgetItem \*item, int i) - *раскрашивает* *ячейки* *таблицы,* *которые* *обозначают* *свободную* *аудиторию* *в* *зааднный* *промежуто*

void Menu::Start\_Color\_Table(QTableWidgetItem \*item, int i) - *раскрашивает* *ячейку* *таблицы* *в* *базовый* *цвет*

short Menu::verification\_time(int hour\_to, int hour\_from, int minute\_to,int minute\_from) - *проверка* *на* *коректность* *ввода* *времени*

void List::set\_list(QString stroka) - *выводит* *список* *аудиторий* *на* *экран*

void Find::Void\_Audit(int from\_time\_heur, int from\_time\_minute,int to\_time\_heur, int to\_time\_min) - *устанавливает* *знак* *-* *говорящий* *о* *том,* *что* *аудитория* *будет* *занята*

QString Find::set\_list\_audit() - *в* *list\_audit* *выводит* *список* *не* *занятых* *в* *указанное* *время* *аудиторий*

bool Find::Now\_exame(int from\_time\_heur, int from\_time\_minute,int to\_time\_heur, int to\_time\_min,QString stroka) - *овтечает* *на* *вопрос* *"* *в* *выбранное* *время* *есть* *экзамен* *в* *данной* *аудитории* *?"*

void Find::clear\_file() - *очищает* *файл* *с* *пустыми* *аудиториями*

Также были созданы функции, которые обрабатывали события нажатия на какую-либо кнопку:

void MainWindow::on\_pushButton\_clicked() – подтверждение группы

void MainWindow::on\_action\_triggered() – закрытие окна

void MainWindow::on\_pushButton\_2\_clicked() – переход по ссылке создателя проекта

void Menu::on\_pushButton\_clicked() – чтение из файла информации о расписании экзаменов для заданной группы

void Menu::on\_find\_aud\_clicked() – открывает 3 форму с пустыми аудиториями в указанное время

void Menu::on\_action\_triggered() *- закрываем* *окно с расписанием экзамена*

void Menu::on\_pushButton\_2\_clicked() *- -закрываем* *окно* *списка* *экзаменов* *и* *сменяем* *группу* *в* *главном* *окне*

void List::on\_action\_triggered() – закрывает окно с пустыми аудиториями

Список классов-кандидатов, полученный на основе данного анализа, выглядит следующим образом:

• класс Find - объекты данного класса должны создаваться каждый раз, когда пользователь желает найти свободную аудиторию в указанное время;

• класс List — объекты данного класса должны создаваться при выводе списка пустых аудиторий на экран;

• класс MainWindow — объекты данного класса должны создаваться при запуске программы;

• класс Menu— объекты данного класса должны создаваться при успешном вводе пользователем номера группы

Диаграмма классов



Список выявленных понятий

В графическом приложении Qt обязательно должен быть создан один объект QApplication. QApplication управляет ресурсами приложения, например, задавая тип шрифта или курсора.

Функция main() является точкой входа в вашу программу. В большинстве случаев при написании приложения Qt, только main() нуждается в определённых, связанных с инициализацией действиях до передачи управления библиотеке Qt, которая уже потом, в свою очередь, сообщает программе о действиях пользователя.  
Параметр argc — число аргументов командной строки, а argv — массив этих аргументов. Это — стандартная особенность C++.

Объект app является экземплятором QApplication. В этой строчке мы его создаем. Мы передаём параметры argc и argv в конструктор QApplication, чтобы он мог обработать стандартные аргументы командной строки (такие как -display при работе с X11). Все переданные аргументы, обработанные Qt, будут удалены из argv, argc уменшится в соответствии с этим. Для получения подробной информации о работе с этими параметрами посмотрите документацию на метод QApplication::argv().  
Объект QApplication должен быть создан прежде, чем вы начнете работать с графическим интерфейсом пользователя Qt.

В этом месте main() передаёт управление Qt. QCoreApplication::exec() вернёт значение, когда приложение закончит работу. (QCoreApplication является базовым классом для QApplication. Он определяет ядро QApplication, функциональность, не связанную с GUI, и может быть использован при разработке приложений без графического интерфейса пользователя).  
В методе QCoreApplication::exec() происходит обработка всех пользовательских и системных событий и передача управления соотвествующим виджетам.

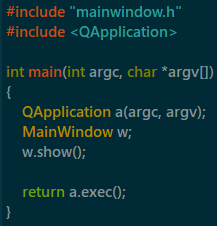


Рисунок 7

С помощью метода show объекта класса MainWindow, вызывается главное окно в котором пользователю требуется ввести номер группы. Для того, чтобы узнать расписание экзаменов группы, пользователю требуется вести верно номер группы.

В целях защиты приложения от неудачных попыток пользовательского ввода, были обработаны следующие исключения:

1. Проверка открытия текстового файла
2. Проверка доступности ссылки на страницу создателя проекта

Помимо исключений могут возникнуть проблемы с тем, что программа не сможет найти расписание данной группы и тогда выводить в таблицу расписаний экзаменов будет просто нечего. В качесвте решения были использованы условные операторы if-else.

Чтобы сообщить пользователю, что была найдена ошибка в вводе или программой было поймано исключение – используется класс QMessageBox, который в моем приложении использует метод для вывода сообщение об ошибке и метод для вывода сообщения об удачном вводе данных. Пример см. на Рисунке 8.

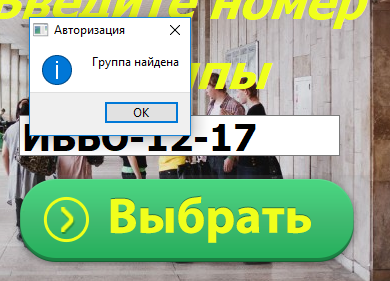


Рисунок 8

После ввода номер группы и нажатия на кнопку «Выбрать» программа» создается объект класса QFile, вызывая конструктор с параметром. В качесвте параметра в контсруктор передается относительный адрес текстового файла «Groupe.txt». Затем программа пытается открыть данный файл по этому адресу и в случаи успешного открытия создает объект класса QTextStream с вызывом коструктора с параметром адреса данного файла. С помощью этого объекта программе предоставляется возможность считывать данные с файла, благодаря этому мы проверяем «есть ли введенная пользователем группа в файле ?».

В файле для удобства поиска, группы обозначены спецаильный символом перед их номером, с помощью этого программа работает более эффективно по времени. (см. Рисунок 9)

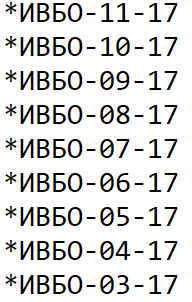


Рисунок 9

Когда программа находит нужную группу, то она создает указатель на класс Menu и закрывает открытый файл. Но так как класс Menu ялвяется наследником класса QMainWindow, то использую метод show базового класса мы вызываем новое окно с расписанием экзаменов. Чтобы изобразить номер группы в новом окне, через указатель menu передаем текст, который ввел пользователь в LineEdit, используя механизм параметров в метод класса Menu, где и происходит установка текста на новое окно.

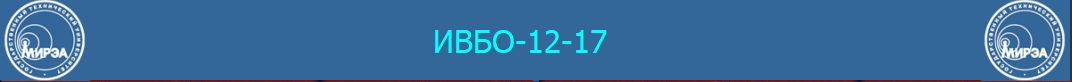


Рисунок 10

Вывод расписания

Даллее в новом окне , по нажатию на кнопку «Вывести расписание экзаменов», снова открывается тот же файл и по той же схеме находит группу. Найдя нужную группу начинается считывание информации об экзаменах для данной группы. Чтобы программа не тратила лишнее время на считывание не нужной информации, она будет считывать ее до тех пор, пока не встретит новый символ «\*», обозначающий, что в этой строке содержится номер другой группы. Каждая строка с символом «/» содержит в себе 4 пробела. Считывание происходит таким образом, что каждая строка ( каждый символ «/» ) означает отдельную строку двумерного массива, а каждый пробел в документе означает номер столбца в этом массиве.

После записи всех данных в двумерный массив, начинается вывод этих данных уже в таблицы, причем благодаря изначальному выбору записи данных в массив, программе удобно находить нужные ячейки в таблицах, в которые нужно вставить соответствующую информацию.



Рисунок 11

Нахождение свободной аудитории в указанное время

На рисунке 4 можно увидеть 2 виджета QTimeEdit. Они предназначены для указания пользователем времени «От» и «До» какого времени ему потребуется найти свободную аудитория. С помощью помощью SpinBox, расположенных на данных виджетах (см. Рисунок 11) , пользователь может менять время, причем в целях безопасности установлены ограничения:

1. Время «От» как минимум должно быть равно «10:00» и максимум «20:00»
2. Время «От» не должно быть позже чем время «До»

Первым делом по нажатию на данную кнопку программа создаст объект

Объект класса QTime, что позволит нам получить доступ к значениям на виджете QTimeEdit. Пройдя проверку на корректность веденных данных в виджеты – программа создаст объект класса Find через который буде отрыто окно с тображением списка пустых аудиторий, после чего, используя цикл for, будет происходить выделение синим цветом в таблицах окна с расписанием экзаменов тех ячеек, которые подходят по требованиям пользователя и выделение в стандартный цвет тех ячеек, которые под эти требования не подходят. (см. Рисунок 6)

Вывод списка пустых аудиторий

После успешного ввода данных в QTimeEdit по нажатию на кнопку «Найти свободную аудиторию» начнется процесс формирования списка пустых аудиторий.

Для формирования этого списка был создан специалньый текстовый документ «Auditor.txt». Первый делом программа очищает данный файл, а уже после создает 2 разных объекта класса QFile, одни из которых имеет адрес файла «Auditor.txt», а другой адрес файла «Groupe.txt». В данном случаи файлы открываются с разным назначением:

1. Файл «Auditor.txt» открывается для записи данных
2. Файл «Groupe.txt» открывается для чтениях данных

После этого начинается считывание файла «Groupe.txt», причем, когда при считывании с помощью метода класса QTextStream – indexOf в считанной строке программа находит символы «/», «А-», то тут же проводится запись данной аудитории в файл «Auditor.txt».

Примечание: данные записываются в файл «Auditor.txt» с разделяющими множествами пробелов и после каждой 20-ой записанной в файл аудитории производится перенос строки записи и процесс продолжается до тех пор все 500 аудиторий не будут проверены на выполнения требований пользователя (см. Рисунок 12).

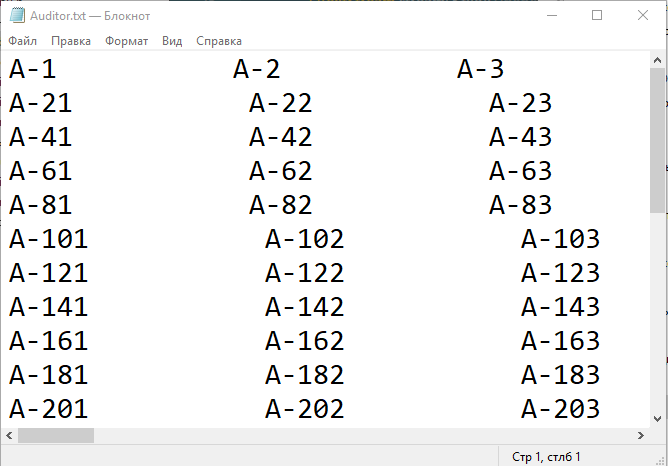


Рисунок 12

Данное представление данных было сделано с целью удобного вывода списка пустых аудиторий в QTextEdit. После сформированного списка пустых аудиторий программа создает объект класса List и передает в его метод строковую переменную, которая содержит в себе все содержимое текстового файла «Auditor.txt», а уже в этом методе происходит установка списка пустых аудиторий в QTextEdit. (см. Рисунок 5)

ЭКСПЕРЕМЕНТАЛЬНАЯ ЧАСТЬ

Формирование набора тестов имеет большое значение, поскольку тестирование является одним из наиболее трудоемких этапов (от 30 до 60 % общей трудоемкости) создания программного продукта. Причем доля стоимости тестирования в общей стоимости разработки имеет тенденцию возрастать при увеличении сложности программного обеспечения и повышении требований к их качеству.

Существуют два принципиально различных подхода к формированию тестовых наборов: структурный и функциональный.

1. Структурный подход базируется на том, что известка структура тестируемого программного обеспечения, в том числе его алгоритмы («стеклянный ящик»). В этом случае тесты строят так, чтобы проверить правильность реализации заданной логики в коде программы.
2. Функциональный подход основывается на том, что структура программного обеспечения не известна («черный ящик»). В этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также подходом, управляемым данными, так как при его использовании тесты строят на базе различных способов декомпозиции множества данных.

В качестве формирования тестов я выбрал Функциональный подход. В этом случае программа рассматривается как «черный ящик», и целью тестирования является выяснение обстоятельств, в которых поведение программы не соответствует спецификации. Для обнаружения всех ошибок в программе, используя управление по данным, необходимо выполнить исчерпывающее тестирование, т. е. тестирование на всех возможных наборах данных. Для тех же программ, где исполнение команды зависит от предшествующих ей событий, необходимо проверить и все возможные последовательности. Очевидно, что проведение исчерпывающего тестирования для подавляющего большинства случаев невозможно. Поэтому обычно выполняют «разумное» или «приемлемое» тестирование, которое ограничивается прогонами программы на небольшом подмножестве всех возможных входных данных. Этот вариант не дает гарантии отсутствия отклонений от спецификаций. Правильно выбранный тест должен уменьшать, причем более чем на единицу, число других тестов, которые должны быть разработаны для обеспечения требуемого качества программного продукта. При функциональном тестировании различают следующие методы формирования тестовых наборов:

• эквивалентное разбиение - область всех возможных наборов входных данных программы по каждому параметру разбивают на конечное число групп;

• анализ граничных значений - основан на исследовании входных данных;

• анализ причинно-следственных связей - использует алгебру логики и оперирует понятиями «причина» и «следствие»;

• предположение об ошибке - в значительной степени основана на интуиции;

Тестирование

Тест 1

Нажатие на кнопку в toolbar «Файл» и выбор команды «Закрыть»

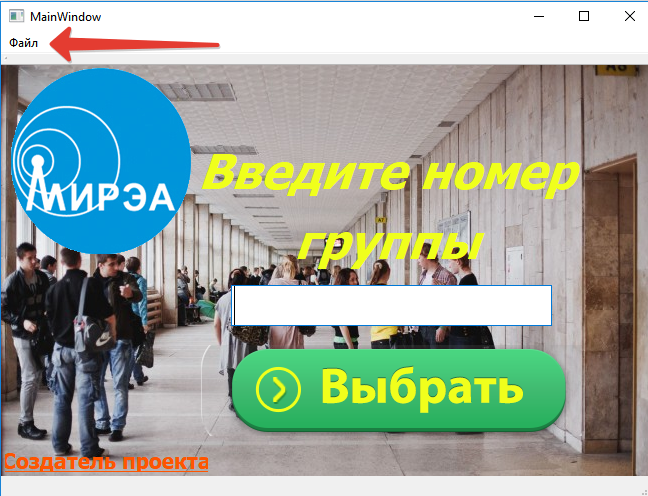


Рисунок 13

Результат: приложение закрылось.

Тест 2

Переход по ссылке «Создатель проекта»

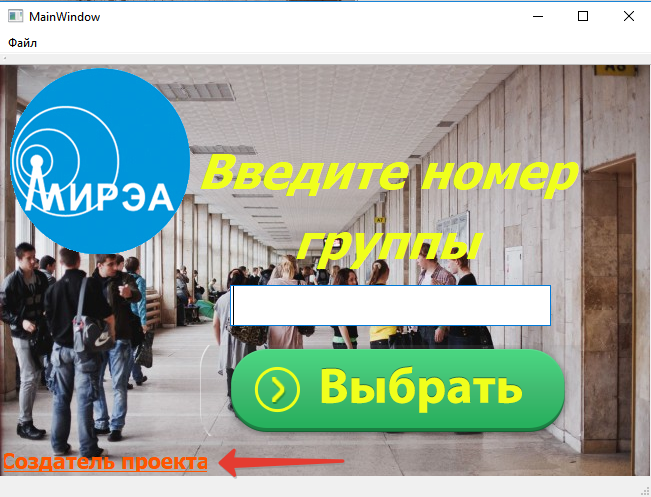


Рисунок 14

Результат: произошел вход на страницу создателя проекта .



Тест 3

Нажатие в главном окне на кнопку «Выбрать» при не заполннном поле ввода

Результат:

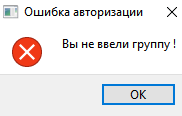


Рисунок 15

Тест 4

Ввод несуществующего номера группы

Результат:

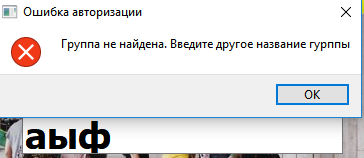


Рисунок 16

Тест 5

Ввод существующего номера группы

Результат:

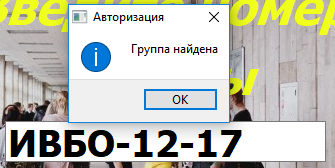


Рисунок 17

Тест 6

Нажатие на кнопку в toolbar «Файл» и выбор команды «Закрыть» в окне меню

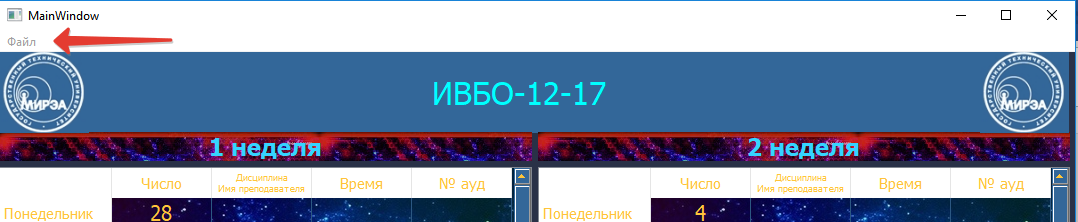


Рисунок 18

Результат: приложение закрылось.

Тест 7

Нажатие на кнопку «Вывеси расписание экзаменов»



Рисунок 19

Результат:



Рисунок 20

Тест 8

Нажатие на кнопку «Найти свободную аудиторию в указанное время» с не верно указанным промежутком времени



Рисунок 21

Результат:

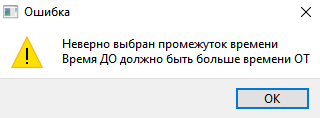


Рисунок 22

Тест 9

Нажатие на кнопку «Найти свободную аудиторию в указанное время» с верно указанным промежутком времени , не попадающий на время проведеннеие всех экзаменов



Рисунок 23

Результат:

1)Все свободные аудитории в указанное время сменили свой цвет ячейки на синий;



Рисунок 24

2)В новом окне представлен список свободных аудиторий в указаное время

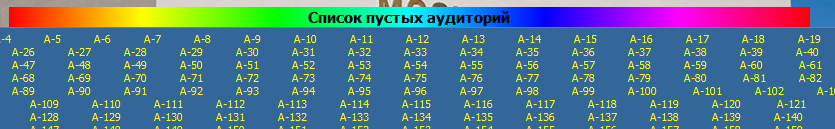


Рисунок 25

Тест 10

Нажатие на кнопку «Найти свободную аудиторию в указанное время» с верно указанным промежутком времени , попадающий на время проведеннеие всех экзаменов в таблице



Рисунок 26

Результат:

1)Все занятые аудитории сменили цвет своей ячейки на красный;



Рисунок 27

2) В новом окне представлен список свободных аудиторий в указаное время

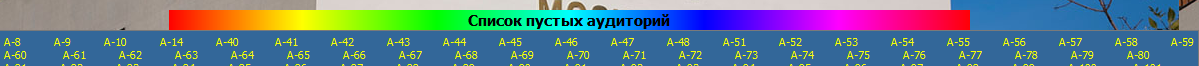


Рисунок 28

Тест 11

Нажатие на кнопку «Найти свободную аудиторию в указанное время» с верно указанным промежутком времени , частично попадающий на время проведеннеие экзаменов в таблице



Рисунок 29

Результат:

1)Все свободные аудитории в указанное время сменили свой цвет ячейки на синий, а остальные в красный;



Рисунок 30

2)В новом окне представлен список свободных аудиторий в указаное время



Рисунок 31

Тест 12

Нажатие на кнопку в toolbar «Файл» и выбор команды «Закрыть» в окне со списками свободных аудиторий

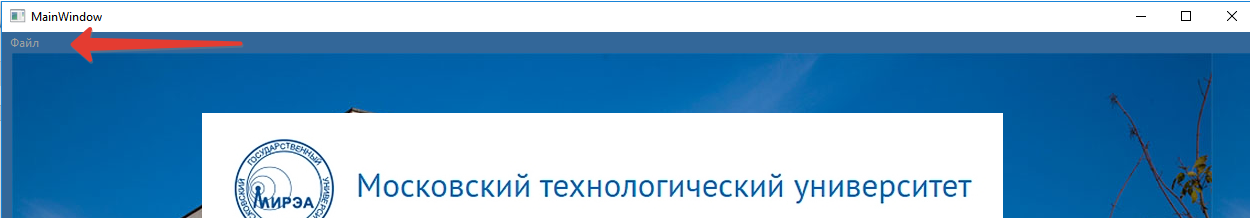


Рисунок 32

Результат: приложение закрылось.

Тест 13

Нажатие на кнопку «Сменить группу»



Рисунок 33

Результат: приложение возвращает в окно изображенный на рис.

Таблица 2 – Тестирование курсовой работы

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тест\Вариант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1 | + | + | + | + | + | + | + | + | + | + | + |
| 2 | + | + | + | + | + | + | + | + | + | + | + |
| 3 | + | + | + | + | + | + | + | + | + | + | + |
| 4 | + | + | + | + | + | + | + | + | + | + | + |
| 5 | + | + | + | + | + | + | + | + | + | + | + |
| 6 | + | + | + | + | + | + | + | + | + | + | + |
| 7 | + | + | + | + | + | + | + | + | + | + | + |
| 8 | + | + | + | + | + | + | + | + | + | + | + |
| 9 | + | + | + | + | + | + | + | + | + | + | + |
| 10 | + | + | + | + | + | + | + | + | + | + | + |
| 11 | + | + | + | + | + | + | + | + | + | + | + |
| 12 | + | + | + | + | + | + | + | + | + | + | + |
| 13 | + | + | + | + | + | + | + | + | + | + | + |

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы проводилась разработка алгоритмов, тестирование и отладка, как самого приложения, так и отдельных его модулей.

СПИСОК ЛИТЕРАТУРЫ

1)Документация Qt

URL: <http://doc.qt.io/>

2) Бьерн Страуструп «Язык программирования С++» 2-е дополненное издание «Языки программирования / С++» 2013 г.

3)Генри Уоррен – «Алгоритмические трюки для программистов» 2-е изадние 2014 г.

4)Элайн А. – «С++ от ламера до программера» 2015 г.

5)Алексей Васильев – «Программирвоание на С++ в примерах и задачах» 2017 г.

ПРИЛОЖЕНИЕ

Код модулей приводится в приложениях.