|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | ВМСС |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(бакалаврская работа)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Направление** | | | | 09.03.01 – Информатика и вычислительная техника | | | | | |
| (код и наименование) | | | | | | | | | |
| **Направленность (профиль)** | | | | | | | | | Вычислительные машины, комплексы, |
| системы и сети | | | | | | | | | |
|  | | | | | | | | | |
| **Форма обучения** | | | | | | | Очная | | |
|  | | (очная/очно-заочная/заочная) | | | | | | | |
| **Тема:** | | Разработка программы обработки результатов модельных | | | | | | | |
| экспериментов | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| **Студент** | | | А-08-17 Полушин А.И. | | | | | | |
| группа подпись фамилия и инициалы | | | | | | | | | |
| **Научный руководитель** | | | | | | д.т.н. профессор Абросимов Л.И. | | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | | | | |
| **Консультант** | | | | | | к.т.н. зам. нач. СКБ-230 НТЦ “НИИРП” ПАО “НПО” Алмаз”Фащев В.В. | | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | | | | |
| **«Работа допущена к защите»** | | | | | | | | | |
| **Зав. кафедрой** | | | | | к.т.н доцент Вишняков С.В. | | | | |
| уч. степень звание подпись фамилия и инициалы | | | | | | | | | |
| **Дата** |  | | | | | | |  | |
|  | | | | | | | |  | |
| **Москва, 2019** | | | | | | | | | |

# АННОТАЦИЯ

В данной выпускной квалификационной работе выполнены проектирование и разработка программы обработки результатов модельных экспериментов для **ПАО «НПО «Алмаз»**. Структура программы состоит из загрузки данных в базу данных, редактирования данных, моделирования данных по начальным значениям, обработки данных и построении графика по обработанным данным. Для реализации описанной программы использовался язык программирования C++ и фреймворк Qt. Для хранения данных использовалась реляционная система управления базами данных SQLite. Работа состоит из четырех- разделов, 146 страниц, 20 рисунков, 3 таблиц и 8 приложений.

# ABSTRACT

This final qualification paper describes the design and development of a program for processing the results of model experiments for the JSC NPO Almaz. The structure of the program consists of loading data into a database, editing data, modeling data based on the initial values, processing data, and construction of graph based on the processed data. To implement the described program, were used the C ++ programming language and the Qt framework. The relational database management system SQLite was used to store the data. The work consists of 3 sections, 146 pages, 20 figures, 3 tables and 8 applications.

Содержание

[ВВЕДЕНИЕ 6](#_Toc74680266)

[1. МОДЕЛЬ ОБЪЕКТА И МОДЕЛЬНЫЙ ЭКСПЕРИМЕНТ 8](#_Toc74680267)

[1.1. Модельный эксперимент 8](#_Toc74680268)

[1.2. Модель объекта 9](#_Toc74680269)

[2. ПРОЕКТИРОВАНИЕ ПРОГРАММ 10](#_Toc74680270)

[2.1. Структура базы данных 10](#_Toc74680271)

[2.2. Программа заполнения базы данных 11](#_Toc74680272)

[2.3. Программа обработки результатов модельных экспериментов 14](#_Toc74680273)

[3. ПРОЦЕСС РАЗРАБОТКИ 18](#_Toc74680274)

[3.1. База данных 18](#_Toc74680275)

[3.2. Программа заполнения базы данных 19](#_Toc74680276)

[3.2.1. Заполнение базы данных вручную 21](#_Toc74680277)

[3.2.2. Заполнение базы данных из файла 24](#_Toc74680278)

[3.2.3. Моделирование траектории полета воздушного объекта 26](#_Toc74680279)

[3.3. Реализация программы обработки результатов модельных экспериментов 35](#_Toc74680280)

[3.3.1. Обработка результатов 38](#_Toc74680281)

[3.3.2. Фильтр базы данных 41](#_Toc74680282)

[3.3.4. Построение графиков 41](#_Toc74680283)

[4. ТЕСТИРОВАНИЕ ПРОГРАММ 54](#_Toc74680284)

[ЗАКЛЮЧЕНИЕ 59](#_Toc74680285)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 61](#_Toc74680286)

[*Приложение А. Техническое задание* 63](#_Toc74680287)

[*Приложение Б. Листинг заголовочного файла mainwindow1.h.* 67](#_Toc74680288)

[*Приложение В. Листинг файла mainwindow1.cpp.* 70](#_Toc74680289)

[*Приложение Г. Листинг файла prognoz.h.* 88](#_Toc74680290)

[*Приложение Д. Листинг заголовочного файла mainwindow2.h.* 97](#_Toc74680291)

[*Приложение Е. Листинг файла mainwindow2.cpp.* 101](#_Toc74680292)

[*Приложение Ж. Листинг заголовочного файла mygraphicview.h.* 131](#_Toc74680293)

[*Приложение И. Листинг файла mygraphicview.cpp.* 136](#_Toc74680294)

**Обозначения и сокращения**

|  |  |  |
| --- | --- | --- |
| БД | - | база данных |
| СУБД | - | Система управления базами данных |
| ТЗ | - | Техническое задание |
| ЭПР | - | Эффективная площадь рассеяния |

# ВВЕДЕНИЕ

Современная геополитическая обстановка в мире ставит перед Российской Федерацией задачу создания максимально эффективной системы обороны от любых средств нападения как из воздушной среды, так и из космического пространства [1]. Решение этой задачи требует непрерывного наблюдения за воздушной обстановкой и быстрой обработки информации с последующей передачей её вышестоящим пунктам для принятия решения о дальнейших действиях. Обработкой информации занимаются специально обученные люди. До появления в производствах компьютеров работа была сложной и занимала много времени. Теперь же решение задач обработки информации не требует таких усилий. Однако, появилась необходимость постоянно обновлять и улучшать программное обеспечение.

**Для ПАО «НПО «Алмаз», занимающейся разработкой программного обеспечения и техники для Вооруженных Сил Российской Федерации, необходимо разработать программу, которая будет производить селекцию движущейся в трехмерном пространстве цели. Для этого необходимо обработать результаты модельных экспериментов. Суть обработки состоит в усреднении параметров с привязкой к другому параметру. Программа будет являться компонентом для тестирования. В ней будет происходить селекция движущейся цели по заданным параметрам, и моделирование параметров. Для наглядности параметры модельных экспериментов необходимо выводить на графики. Все параметры хранятся в базе данных SQLite.**

В соответствии с техническим заданием (см. приложение А) были определены следующие задачи:

– Определить параметры воздушных объектов[[1]](#footnote-1).

– Составить структуру базы данных.

– Разработать программу заполнения базы данных.

– Разработать программу обработки информации путем привязки и усреднения заданных параметров.

– Сделать вывод по результатам работы.

Для решения поставленных задач использованы следующие методы:

– Теоретический анализ и обобщение литературных данных.

– Статистическая обработка данных.

– Моделирование объекта в трехмерном пространстве.

– Математическая обработка данных.

Исходными данными для разработки являются:

– Фреймворк Qt C++.

– Система управления базами данных SQLite.

Ожидаемыми результатами являются: программа заполнения базы данных и программа обработки полученной информации с удобными и понятными интерфейсами.

В данной работе необходимо использовать Qt – фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++. Преимущества Qt в его открытости и доступности, кроссплатформенности, наличии хорошей и полной документации [2]. Также для хранения и структурирования данных необходимо использовать СУБД SQLite. SQLite – это система управления базами данных, отличительной особенностью которой является ее встраиваемость в приложения [3].

# 1. МОДЕЛЬ ОБЪЕКТА И МОДЕЛЬНЫЙ ЭКСПЕРИМЕНТ

В первой части данной главы рассматриваются понятия модели и модельного эксперимента. Вторая часть посвящена формальному способу описания модели объекта.

## **Модельный эксперимент**

Рассмотрим термин «модель» как способ отображения объекта, средство для получения информации и знаний. Модель — это объект, сохраняющий только важные для исследования свойства настоящего или существующего объекта.

Под термином «эксперимент» понимается деятельность, выполняемая в целях научного познания [4]. По характеру и разнообразию средств и способов их использования эксперименты различают на натурный и модельный. Их отличие состоит в том, что в модельном эксперименте необходимо обосновать отношение модели и оригинального объекта.

Преимущества модельного эксперимента в отличии от натурного:

– изоляция исследуемого объекта от влияния различных побочных явлений;

– возможность многократного повторения эксперимента в контролируемых условиях;

– возможность устанавливать начальные условия эксперимента.

Воздушный объект можно охарактеризовать различным набором параметров, многие из которых не представляют полезной информации. Таким образом, чтобы абстрагироваться от ненужных переменных, свойств и взаимосвязей воздушного объекта нужно создать его модель.

В качестве исследуемого модельного эксперимента будет выступать движущийся в пространстве воздушный объект. Его модель представляет собой набор параметров, описывающих движение объекта в пространстве.

## Модель объекта

Как уже было изложено выше экспериментом является движение объекта в трехмерном пространстве. Сразу можно выделить основные параметры модели – координаты объекта в трехмерном пространстве. Так как объект меняет свои координаты со временем, то логично выделить следующий параметр – время привязки информации. Несколько объектов могут фиксироваться радаром как один, поэтому уместен параметр – номер объекта. Всякий воздушный объект обладает свойством отражать падающую на него электромагнитную энергию. Чтобы оценить отражательную способность воздушного объекта вводится следующий параметр – ЭПР. ЭПР является дополнительным параметром для тестирования работоспособности программы.

Таким образом, сформирована модель с основными параметрами для решения задачи обработки результатов модельных экспериментов. По этой модели проектируется база данных.

# 2. ПРОЕКТИРОВАНИЕ ПРОГРАММ

В соответствии с ТЗ необходимо хранить все данные в базе данных *SQLite*. В первой части данной главы описывается структура базы данных.

Как уже отмечалось выше, одна из задач выпускной квалификационной работы (ВКР) заключается в создании программы или подпрограммы заполнения базы данных. Вторая часть данной главы содержит в себе основные этапы проектирования этой программы.

Обработку результатов модельных экспериментов выполняет вторая программа. Описание этапов проектирования этой программы содержатся в третьей части данной главы.

## 2.1. Структура базы данных

В главе 1.2. описана модель объекта, были выделены основные параметры: координаты в трехмерном пространстве, ЭПР, время привязки информации, номер объекта. Эти параметры будут основой структуры базы данных. Каждый эксперимент содержит в себе множества этих параметров. Эксперименты отличаются временем начала и названием с комментарием пользователя.

Таким образом, структура базы данных принимает следующий вид: (рис. 1), где, соответственно обозначены, таблица экспериментов с первичным ключом *expid*, названием эксперимента *name*, датой и временем начала эксперимента *date* и комментарием пользователя – *comm*. Таблица параметров с первичным ключом *paramid*, внешним ключом *expid*, временем привязки параметров – *date*, координатами *X, Y* и *Z,* ЭПР – *epr*, и номером объекта - *number*. На столбцах *expid* таблицы экспериментов и *paramid* таблицы параметров установлен автоинкремент[[2]](#footnote-2) для упрощения создания новых записей. Столбцы *date* двух таблиц по умолчанию добавляют текущую дату и время. Столбец *number* содержит проверку вводимых значений. Значение этого столбца должно быть больше 0.



Рис. 1. Структура базы данных

## 2.2. Программа заполнения базы данных

Основными функциями программы заполнения базы данных являются заполнение и редактирование базы данных. Для редактирования базы данных лучше всего подходит ручное редактирование, а для заполнения – моделирование траектории полета воздушного объекта. Исходя из этого реализуем два варианта: ручное заполнение с редактированием и моделирование траектории полета воздушного объекта.

Для удобства представим базу данных в программе в виде таблиц. Чтобы увидеть параметры конкретного эксперимента, пользователю необходимо выбрать его в таблице.

Для того чтобы пользователь мог удобно редактировать базу данных необходимо реализовать редактирование через выбор ячейки таблиц. Пользователь должен нажать два раза на ячейку, содержимое которой хочет изменить, и ввести новое значение.

Пользователь может допустить ошибки при вводе значений, поэтому необходимо реализовать функции проверки содержимого измененных и добавленных ячеек таблиц. Также необходимо добавить функцию обновления таблиц.

Необходимо предусмотреть импорт параметров эксперимента из файла с расширением txt. В первой строчке txt файла содержатся названия параметров через пробел в следующем порядке: *date, x, y, z, epr, number*. Во всех остальных строках значения параметров в этом же порядке. Значения *date* должны быть в формате: ГГГГ-ММ-ДД чч:мм:сс. Так как значение параметра *date* проставляется автоматически, то его можно не указывать (вместо значения ставится пробел, в случае если в первой строке есть *date*). Примеры txt файлов приведены на рис. 2 и рис. 3.

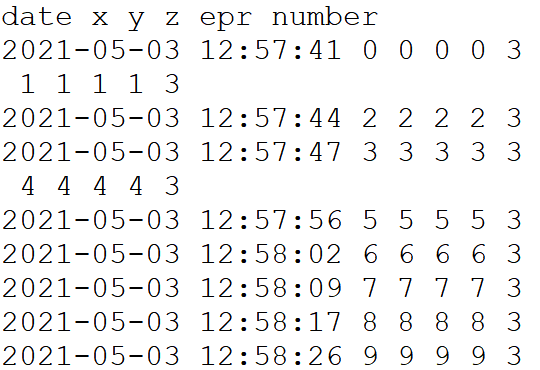


Рис. 2. Пример txt файла

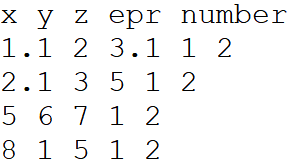


Рис. 3. Пример txt файла (без date)

Для моделирования параметров положения объекта в пространстве необходимо знать следующее: вектор начального положения объекта, вектор начальной скорости объекта, шаг времени и количество параметров. Для моделирования ЭПР необходимо знать тип объекта и угол локации. ЭПР является случайной величиной, а тип объекта влияет на функцию распределения ЭПР. В открытом доступе нет информации о типе и параметрах функций распределения. Поэтому за основу возьмем две функции распределения: экспоненциальную и равномерную. В качестве параметров функций распределения будут средние значения ЭПР для первого типа, и минимальное с максимальным значением ЭПР для второго типа. Типы делятся на подтипы, которые влияют на параметры функций распределения. Угол локации также влияет на параметры функции распределения. Исходя из этого, для моделирования ЭПР необходимо выбрать тип объекта, подтип и угол локации. Все параметры ЭПР являются секретными и поэтому указаны случайными величинами.

Координаты объекта рассчитываются в системе координат радиолокационной станции (РЛС), где ноль и есть положение РЛС. Векторы положения и скорости вводятся проекциями на оси в этой системе координат.

## 2.3. Программа обработки результатов модельных экспериментов

Обработка результатов модельных экспериментов заключается в том, что пользователь должен указать несколько экспериментов (минимум два) и параметр, который нужно найти во всех таблицах этих экспериментов и усреднить по всем объектам, и привязывается к другому параметру, то есть указывает диапазон значений, в пределах которого нужно найти и усреднить заданный параметр. Диапазон значений задается с помощью дельты. Дельта может быть «плавающей»: изменяться на каждом шаге усреднения в зависимости от разности параметров усреднения. Выбранные эксперименты должны добавляться в отдельную таблицу, где можно отменить выбор, очистить выбор и добавить новый эксперимент.

Так же, как и в программе заполнения базы данных, для удобства, представим базу данных в виде таблиц. После того, как пользователь выберет эксперимент, во второй таблице отобразятся параметры этого эксперимента. Для удобства поиска нужного эксперимента добавим фильтр по дате и названию.

Обработанную информацию проще и удобнее хранить в таблице. Заголовки столбцов содержат названия параметров и количество параметров, соответствующих диапазону привязки (сколько значений попало в диапазон изменения). Пример таблицы обработанной информации приведен в табл. 2.2.1. В данном примере пользователь выбрал привязку по координате X и усреднил параметр ЭПР. До обработки пользователь задал несколько экспериментов.

Таблица 1

Пример вывода обработанной информации

|  |  |  |  |
| --- | --- | --- | --- |
|  | Координата X | ЭПР | Количество экспериментов |
| 1 | 10 | 1,1 | 2 |
| 2 | 11 | 3,6 | 2 |
| 3 | 11,5 | 5,7 | 3 |
| 4 | 11,9 | 4,2 | 1 |
| 5 | 11,4 | 3,5 | 3 |
| 6 | 11 | 0,5 | 2 |
| 7 | 12 | 4,8 | 1 |
| 8 | 13 | 1,1 | 1 |
| 9 | 12 | 2,4 | 2 |
| 10 | 12 | 4,1 | 3 |

Следует предусмотреть сохранение результатов обработки информации. Для этого хорошо подходит CSV – формат, предназначенный для представления табличных данных, который читается широко распространенной программой Excel, разработанной корпорацией Microsoft.

В качестве дополнительной обработки необходимо усреднить выбранный пользователем параметр в выбранных экспериментах по всем объектам. Для этого сначала необходимо выбрать эксперименты и указать параметр. Как и в случае обработки результатов модельных экспериментов, выбранные эксперименты хранятся в таблице, где есть возможность добавить новый, отменить выбор и очистить таблицу. Выходным параметром является число – усреднение заданного параметра.

Пользователь не должен изменять базу данных, следовательно, необходимо не допускать изменения ячеек таблиц. Не обойтись и без обработки ошибок, а также ввода некорректных значений. Пользователь должен выбрать эксперименты, параметры усреднения и привязки, а также указать значение дельты или установить «плавающую» дельту.

В программе требуется предусмотреть построение графиков. Графики должны показывать изменение выбранного параметра от времени, а также изменение параметра усреднения от параметра привязки. Гораздо удобнее предусмотреть вывод сразу нескольких графиков, а также удаление или очистка их. Графики должны быть масштабируемыми, а также подстраиваться под размер главного окна программы. Оси на графиках должны автоматически подбирать масштаб.

**В Qt нет стандартных классов или библиотек для работы с графиками. В открытом доступе можно найти средства для построения графиков в Qt, рассмотрим наиболее популярные их них:**

* ***QCustomPlot –* это библиотека для построения графиков предназначена для создания красивых 2D-графиков, графиков и диаграмм. Недостатком *QCustomPlot* является невозможность работы с типом даты и времени. Его достоинства: простота, удобство и множественные виды графиков и диаграмм [5].**
* ***Qwt* – графическая библиотека, позволяющая значительно упростить процесс визуализации данных в программе. Достоинствами этой библиотеки является то, что нет необходимости вручную прописывать элементы отображения, такие как оси, сетки, кривые данных и т.д. Недостаток *Qwt* такой же, как и в предыдущей рассмотренной библиотеке [6].**

**При использовании этих библиотек придется производить конвертацию и настраивать каждую точку на оси на вывод как минимум двух значений (например, часы и минуты), что невозможно. Исходя из этого, необходимо реализовать собственный класс построения графиков.**

# 3. ПРОЦЕСС РАЗРАБОТКИ

В этой главе представлены методы и алгоритмы, используемые в программах. В первой части данной главы описан процесс создания базы данных в соответствии с её структурой. Вторая и третья часть содержат описания программ заполнения базы данных и обработки результатов модельных экспериментов соответственно.

В Qt реализованы сигналы и слоты. Сигнал вырабатывается, когда происходит какое-либо действие или событие, например, нажатие кнопки. Слот – это метод, который присоединяется к сигналу, он вызывается, когда генерируется связанный с ним сигнал [7]. На основе этого реализуются некоторые алгоритмы.

## 3.1. База данных

Исходя из технического задания, выбранная база данных – SQLite. Как уже было сказано, структура базы данных имеет вид рис. 1. Создадим эти две таблицы.

Таблица экспериментов:

CREATE TABLE experiment (

expid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

name VARCHAR(255) NOT NULL,

date DATETIME DEFAULT (DATETIME('now','localtime')),

comm TEXT);

Таблица параметров, связанная с таблицей экспериментов по внешнему ключу – *expid*:

CREATE TABLE param (

paramid INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

date DATETIME DEFAULT (DATETIME('now','localtime')),

x DOUBLE NOT NULL,

y DOUBLE NOT NULL,

z DOUBLE NOT NULL,

epr DOUBLE NOT NULL,

number INT CHECK (number>0) NOT NULL,

expid INT NOT NULL,

FOREIGN KEY (expid) REFERENCES experiment(expid));

Таким образом, получилась реляционная база данных, с двумя таблицами, связанными между собой связью «один ко многим».

## 3.2. Программа заполнения базы данных

Для вывода и редактирования таблиц будет использоваться стандартный класс библиотеки *Qt* – *QTableView*, за которым закрепляется модель – экземпляр класса *QSqlTableModel*. Модель связывается с таблицей базы данных путем вызова методов:

– QSqlTableModel(QObject \*parent = nullptr, QSqlDatabase db = QSqlDatabase()) – конструктор, в который передается открытое соединение с базой данных;

– void QSqlTableModel::setTable(const QString &tableName) – метод, который ставит в соответствие модель и таблицу в базе данных. Его аргумент принимает название таблицы;

– void QSqlTableModel::setFilter(const QString &filter) – метод, который устанавливает фильтр на модели. В него передается строка, содержащая часть запроса в базу данных после слова «WHERE»;

– bool select() – метод выборки таблицы из базы данных.

Модель закрепляется за *QTableView* посредством вызова метода void QTableView::setModel(QAbstractItemModel \*model), в который передается указатель на модель.

Применение изменений осуществляет слот bool QSqlTableModel::submitAll().

После того, как пользователь нажимает на ячейку «эксперимент», номер эксперимента сохраняется в переменной типа *QString* и на второй модели *QSqlTableModel* устанавливается фильтр по номеру эксперимента. Таким образом, после нажатия ячейки «эксперимент» в первой таблице, во второй таблице отображаются параметры соответствующего эксперимента. Нажать можно на любую ячейку, программа найдет строку этой ячейки и сохранит номер эксперимента.

Пользовательский интерфейс программы приведен на рис. 4. На этом рисунке видно, что пользователь выбрал эксперимент «Modeling» в первой таблице и параметры этого эксперимента отобразились во второй таблице.

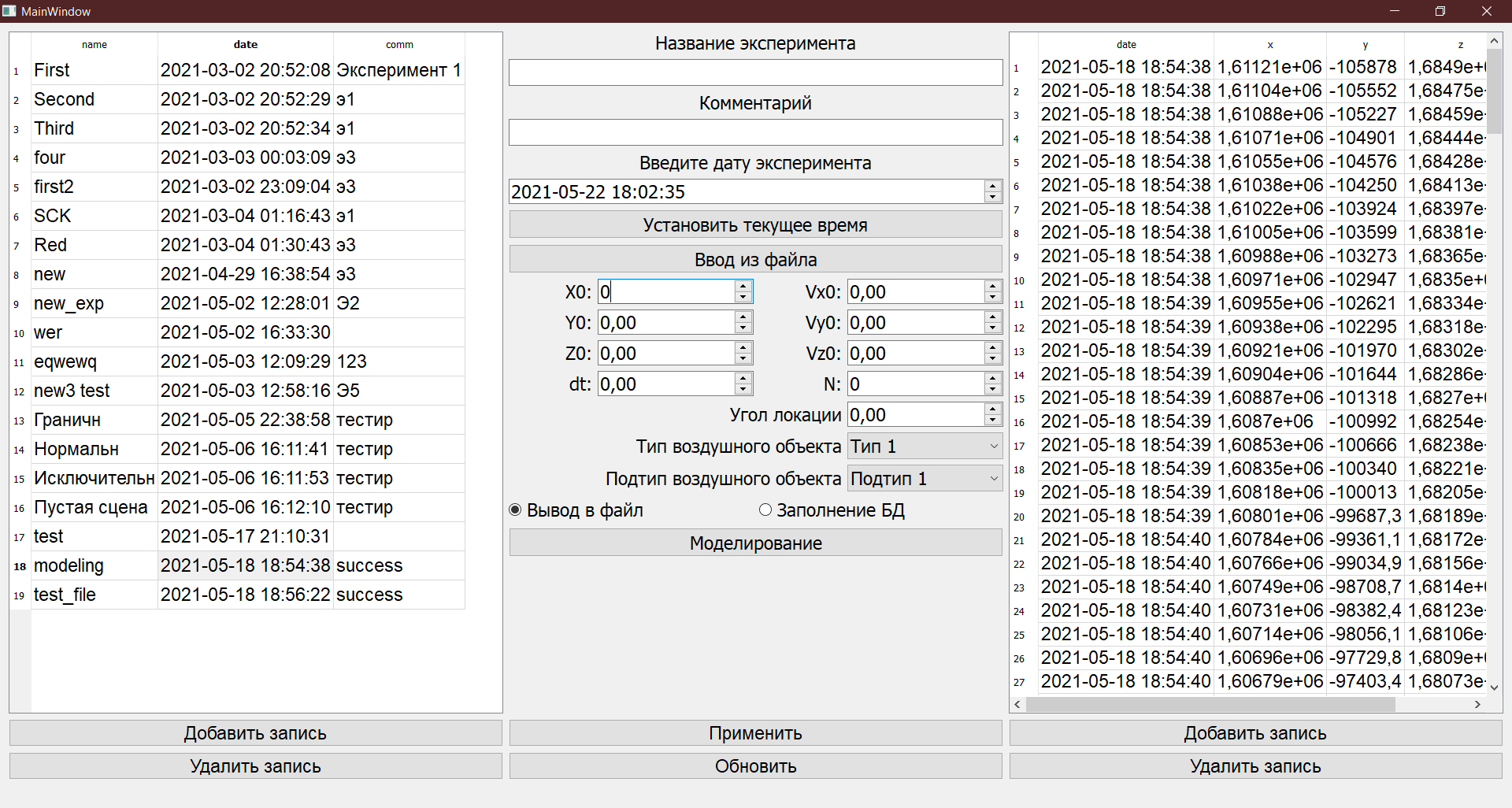


Рис. 4. Пользовательский интерфейс программы (Выбран эксперимент “Modeling”)

### 3.2.1. Заполнение базы данных вручную

Алгоритм заполнения базы данных должен производить проверку вводимых пользователем значений (рис. 5). Для этого создан экземпляр класса *QSet.* Он предоставляет однозначное математическое множество с возможностью быстрого просмотра [8]. В этот экземпляр помещаются индексы измененных ячеек, и по ним производится проверка введенных значений. Алгоритм обращается к данным модели по индексам из множества, и, если находит пустые значения, выдаёт соответствующее сообщение об ошибке.



Рис. 5. Схема алгоритма проверки введенных значений

Заполнение таблиц происходит с помощью кнопки «добавить строку» под соответствующей таблицей и введением значений в добавленные строки. При добавлении строк в таблицы «эксперименты» автоматически проставляется дата. Индексы добавленных строк второго столбца помещаются в упомянутые выше множества. Нет смысла добавлять индексы всех столбцов, потому что дата и время автоматически проставляются в таблицу сразу, а первичный ключ генерируется функцией автоинкремента, и столбец первичного ключа скрыт от пользователя. Схема алгоритма добавления строк в первую таблицу представлен на рис. 6.



Рис. 6. Схема алгоритма добавления строки в 1-ю таблицу

При добавлении строки в таблицу «параметры» автоматически проставляется дата и номер выбранного эксперимента, а также нули в столбцах: координата X, координата Y, высота и ЭПР. Номер объекта по умолчанию равен единице. Внешний ключ таблицы параметров также скрыт от пользователя. Схема алгоритма добавления строк во вторую таблицу представлена на рис. 7.



Рис. 7. Алгоритм добавления строки во 2-ю таблицу

Реализована функция удаления строк у двух таблиц. В случае, если пользователь захочет удалить строку в таблице экспериментов, то ему откроется диалоговое окно с предупреждением об удалении записей, которые связаны с этим экспериментом в таблице «параметры».

По окончании всех изменений необходимо нажать кнопку «применить».

### 3.2.2. Заполнение базы данных из файла

Чтобы добавить новый эксперимент из файла необходимо ввести название эксперимента, комментарий (не обязательно) и дату со временем в соответствующие поля и нажать на кнопку «Ввод из файла». Поле даты и времени автоматически проставляется текущей датой и временем, также это можно сделать, нажав на кнопку «Установить текущее время».

Далее пользователю откроется диалоговое окно, где необходимо выбрать файл с расширением txt. Если структура файла соответствует структуре, описанной в п. 2.2, то в первой таблице отобразится новый эксперимент. Схема алгоритма заполнения базы данных из файла представлен на рис. 8. Алгоритм построчно читает файл и составляет динамический запрос вставки данных в базу данных. Вначале происходит добавление нового эксперимента, и получение его *idexp*. Дальше происходит чтение названий столбцов и составление части запроса вставки. После этого в запрос добавляются значения. Если какой-либо запрос не выполнился, или не удалось считать файл или данные, то переменная логического типа *success* устанавливается в логический ноль. В конце происходит проверка значения этой переменной. Если она равна логическому нулю, то все добавленные данные удаляются, и пользователю выводится сообщение об ошибке.



Рис. 8. Схема алгоритма заполнения БД из файла

### 3.2.3. Моделирование траектории полета воздушного объекта

Для моделирования параметров эксперимента потребуется ввести в соответствующие поля следующее: название эксперимента, комментарий, дату и время, проекция векторов начального положения и начальной скорости объекта на оси координат РЛС, шаг времени, количество параметров, тип объекта, подтип объекта, угол локации. Затем необходимо выбрать куда выгружать данные: в базу данных или в *txt* файл, и нажать на кнопку «Моделирование».

Моделирование производится методом Рунге-Кутты четвёртого порядка по следующим формулам:

где xn+1, yn+1, zn+1 – значения координат в момент времени n+1;

xn, yn, zn – значения координат в момент времени n;

Vxn+1, Vyn+1, Vzn+1 – значения проекций вектора скорости в момент времени n+1;

Vxn, Vyn, Vzn – значения проекций вектора скорости в момент времени n;

h – величина шага;

tn+1 – время на n+1 шаге;

tn – время на n-ом шаге;

dt – шаг времени.

Для нахождения значений проекций вектора скорости в момент времени n+1 служат функции K1(x,y,z,Vx,Vy,Vz,dt), К2(x,y,z,Vx,Vy,Vz,dt), К3(x,y,z,Vx,Vy,Vz,dt). К1(x,y,z,Vx,Vy,Vz,dt) – функция нахождения коэффициентов k11, k21, k31, k41 для проекции вектора скорости на ось X. К2(x,y,z,Vx,Vy,Vz,dt) – функция нахождения коэффициентов k12, k22, k32, k42 для проекции вектора скорости на ось Y. К3(x,y,z,Vx,Vy,Vz,dt) – функция нахождения коэффициентов k13, k23, k33, k43 для проекции вектора скорости на ось Z. Остальные коэффициенты для xn+1, yn+1 и zn+1 находятся по рассмотренным ранее коэффициентам. Функции K1, K2 и K3 будут описаны ниже.

Рассмотрим нахождение коэффициентов на примере нахождения следующего значения координаты xn+1 и значения проекции вектора скорости в момент времени n+1 на ось X, формулы (8 – 23):

где *kx* – поправка к текущему значению координаты x;

*kvx* – поправка к проекции вектора скорости на ось X.

Аналогично находятся остальные значения yn+1, zn+1, Vyn+1 и Vzn+1.

Рассмотрим функции K1, K2, K3 [9]. Входными параметрами являются значения xn, yn, zn, Vxn, Vyn и Vzn. На выходе получается соответствующий коэффициент для Vxn+1,Vyn+1 или Vzn+1. Функция K1:

где Wz – угловая скорость вращения Земли, равная 0.7292115\*10-4 [c-1];

R – средний радиус Земли, равный 6371000 [м];

r – модуль вектора положения (25);

a1,2 – параметры для учета гравитации (26,27);

J2 – коэффициент 2-ой зональной гармоники, равный 108262.575\*10-8;

*BigSemiAxis* – экваториальный радиус Земли, равный 6378136 [м];

*GeocentricalGravityConst* – геоцентрическая гравитационная постоянная Земли, равная 39860044.18\*107 [м3/с2].

Функция K2:

где Wz – угловая скорость вращения Земли, равная 0.7292115\*10-4 [c-1];

R – средний радиус Земли, равный 6371000 [м];

r – модуль вектора положения (25);

a1,2 – параметры для учета гравитации (26,27);

J2 – коэффициент 2-ой зональной гармоники, равный 108262.575\*10-8;

*BigSemiAxis* – экваториальный радиус Земли, равный 6378136 [м];

*GeocentricalGravityConst* – геоцентрическая гравитационная постоянная Земли, равная 39860044.18\*107 [м3/с2].

Функция K3:

где Wz – угловая скорость вращения Земли, равная 0.7292115\*10-4 [c-1];

R – средний радиус Земли, равный 6371000 [м];

r – модуль вектора положения (25);

a1,2 – параметры для учета гравитации (26,27);

J2 – коэффициент 2-ой зональной гармоники, равный 108262.575\*10-8;

*BigSemiAxis* – экваториальный радиус Земли, равный 6378136 [м];

*GeocentricalGravityConst* – геоцентрическая гравитационная постоянная Земли, равная 39860044.18\*107 [м3/с2].

Для моделирования параметров эксперимента создан отдельный класс *prognoz*. Описание класса приведено в табл. 2.

Таблица 2

Описание класса *prognoz.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| prognoz | public | void\* | - | Конструктор |
| Wz | private | double const | - | Угловая скорость Земли |
| J2 | private | double const | - | Коэффициент 2-ой зональной гармоники |
| R | private | double const | - | Средний радиус Земли |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| a00 | private | double const | - | Параметр для учета гравитации |
| a20 | private | double const | - | Параметр для учета гравитации |
| cz | private | double const | - | Параметр для учета гравитации |
| K1 | private | double | const double &x, const double &y, const double &z, double &Vx, const double &Vy, const double &Vz, double &dt | функция нахождения коэффициентов k11, k21, k31, k41 для проекции вектора скорости на ось X |
| K2 | private | double | const double &x, const double &y, const double &z, double &Vx, const double &Vy, const double &Vz, double &dt | функция нахождения коэффициентов k12, k22, k32, k42 для проекции вектора скорости на ось Y |
| K3 | private | double | const double &x, const double &y, const double &z, double &Vx, const double &Vy, const double &Vz, double &dt | функция нахождения коэффициентов k13, k23, k33, k43 для проекции вектора скорости на ось Z |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| prop | public | double | double &t, double &x, double &y, double &z, &Vx, double &Vy, double &Vz, double &dt,const double angle,double &epr, int type,int subtype | Моделирование параметров на один шаг |
| density | private | double[2][35] | - | Массив соответствия высоты и плотности атмосферы |
| getDensity | private | double | const double &z | Расчет высоты и получение плотности |
| SRZNACH\_TYPE1 | private | double[3] | - | Массив средних значение ЭПР для типа 1 (экспоненциальное распределение) |
| MIN\_MAX\_TYPE2 | private | double[2][2][10] | - | Массив соответствия угла локации и минимального с максимальным значением ЭПР для типа 2 (равномерное распределение) |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| my\_uniform | private | double | - | Функция получения случайной величины |
| my\_uniformMINtoMAX | private | double | double min, double max | Функция получения случайной величины с равномерным законом распределения |
| my\_exponential | private | double | double lambda | Функция получения случайной величины с экспоненциальным законом распределения |
| calcEPR\_TYPE1 | public | double | int subtype | Функция получения ЭПР для типа 1 |
| calcEPR\_TYPE2 | public | double | double angle, int subtype | Функция получения ЭПР для типа 2 |

Экземпляр класса *prognoz* моделирует координаты и вектор скорости на один шаг, поэтому вызов метода *prop* происходит в циклеот 0 до N, где N – заданное количество параметров. Для расчета начальной ЭПР предназначены методы *calcEPR\_TYPE1* и *calcEPR\_TYPE2*. Последующий расчет значений ЭПР производятся в методе *prop*. Схема алгоритма метода *prop* (рис. 9) высчитывает значения по формулам (1 – 23).



Рис. 9. Схема алгоритма метода *prop*

## 3.3. Реализация программы обработки результатов модельных экспериментов

Обработки результатов модельных экспериментов состоит из нескольких этапов. Для начала нужно выбрать эксперименты, по которым будет производиться обработка. Для удобства нахождения нужного эксперимента предусмотрен фильтр записей. Как и в случае с программой заполнения базы данных, номер эксперимента определяется автоматически и параметры соответствующего эксперимента отображаются во второй таблице *QTableView*, за которой закрепляется модель *QSqlTableModel*. Чтобы выбрать эксперимент необходимо сначала нажать на него в таблице экспериментов, после этого нажать на кнопку добавить. Выбранный эксперимент отобразится в *QTableView* в нижней части экрана. Чтобы отменить выбор нужно выбрать его в *QTableView* в нижней части экрана и нажать на кнопку «удалить». Эксперимент удалится только из выбранных, изменение базы данных не предусмотрено в этой программе. Чтобы полностью очистить выбранные эксперименты необходимо нажать на кнопку «удалить все». Затем указать какой параметр усреднить и к какому параметру привязаться, а также задать значение дельты или выбрать «плавающую» дельту. После этого нужно нажать на кнопку «обработать». Затем происходит обработка и вывод информации в таблицу. Очистить таблицу можно с помощью кнопки «очистить». Чтобы сохранить таблицу на диске нужно нажать на кнопку «сохранить в файл» и задать расположение на диске в диалоговом окне выбора файлов.

Дополнительно можно построить графики параметров от времени нужного эксперимента или графики по результатам обработки. Графиков можно выводить любое количество. Размер графиков подстраивается под размер интерфейса. Увеличить изображение на графиках можно с помощью колесика мыши. Очистить графики возможно также по кнопке «очистить».

В программе не предусмотрено изменение базы данных, поэтому редактирование *QTableView* запрещено.

Интерфейс программы представлен на рис. 10 и рис. 11.

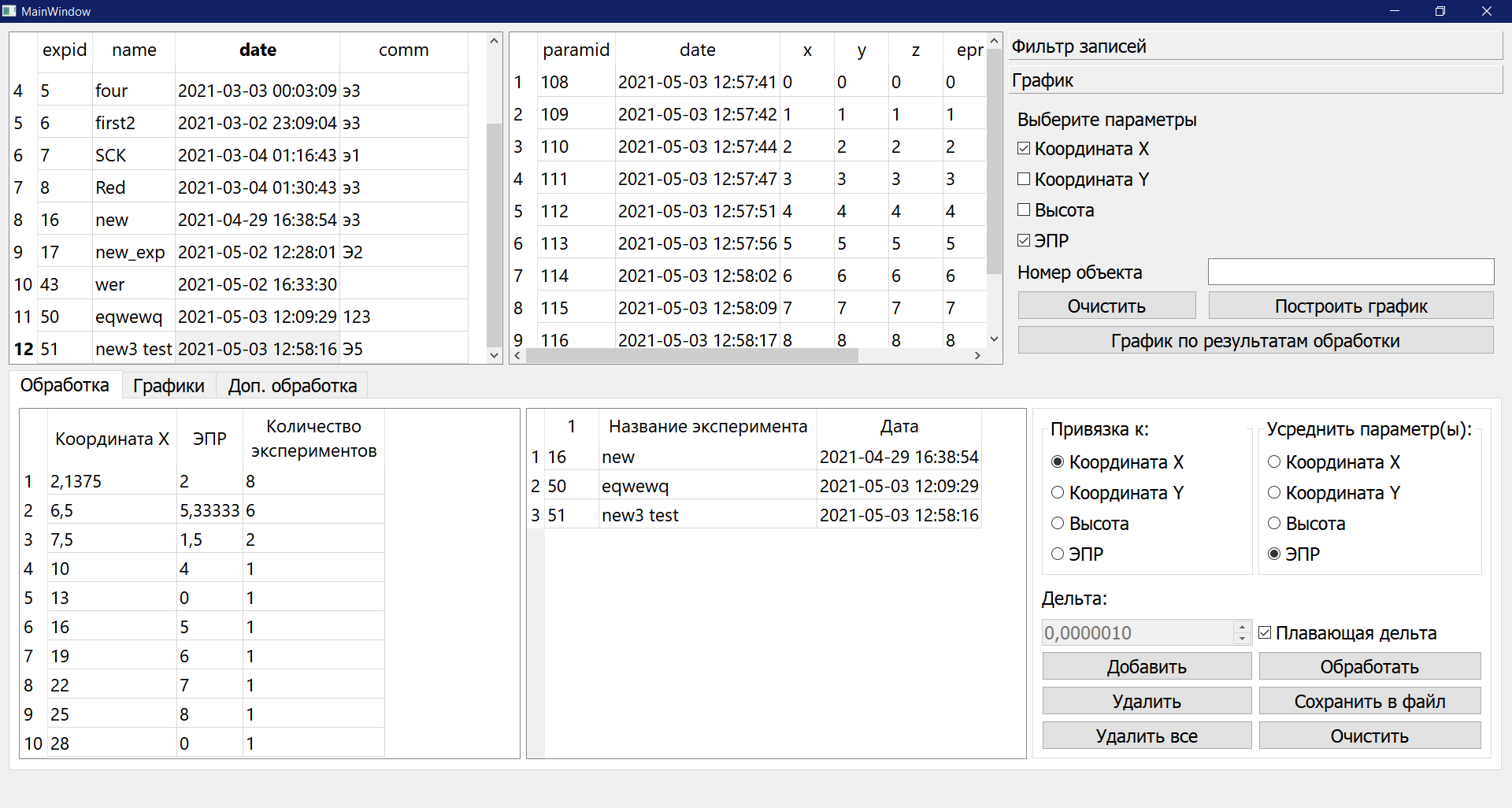


Рис. 10. Интерфейс программы (обработка результатов)

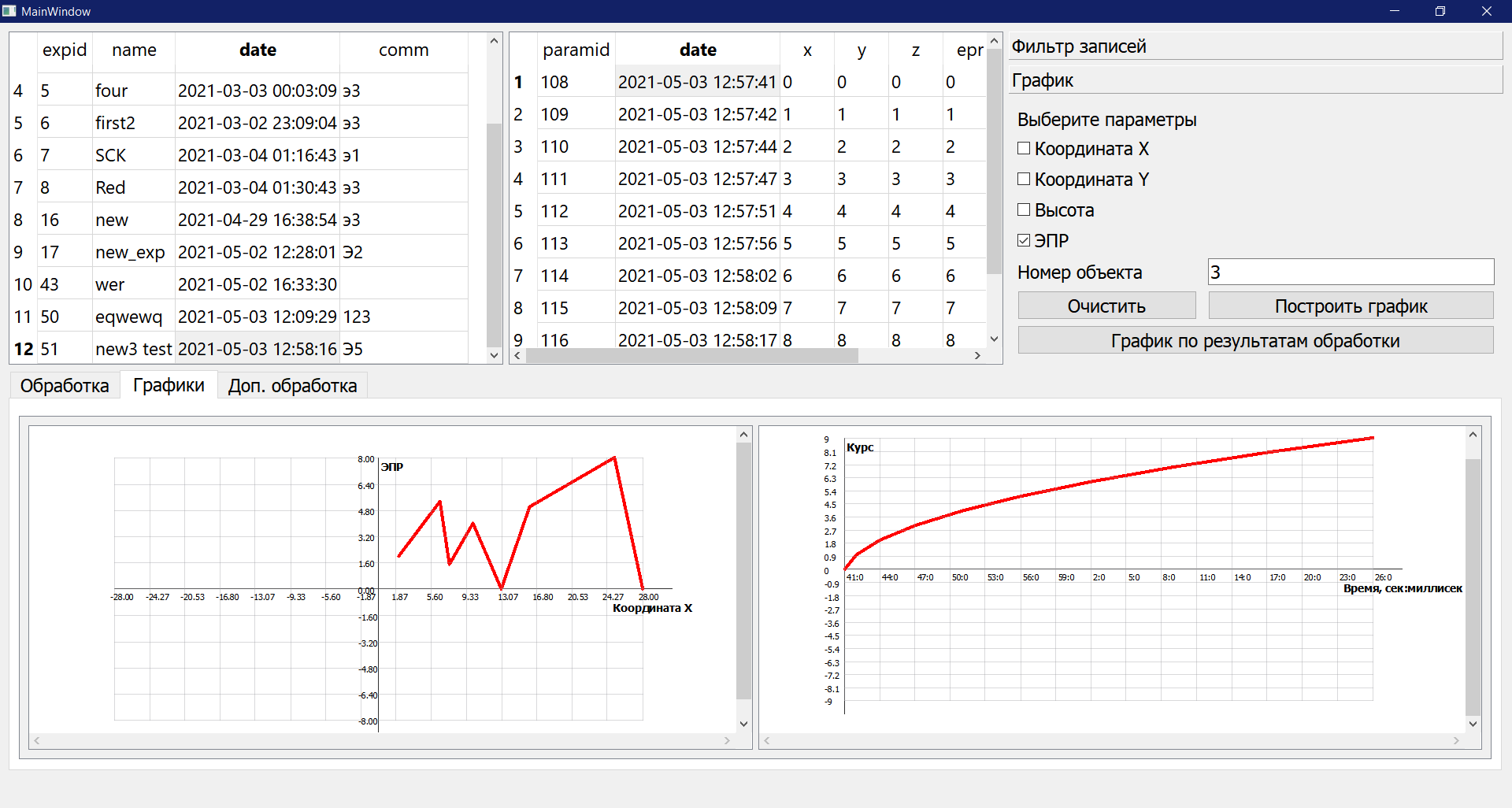


Рис. 11. Интерфейс программы (построение графиков)

### 3.3.1. Обработка результатов

После выбора экспериментов необходимо выбрать параметр усреднения и параметр привязки и задать значение дельты или указать «плавающую» дельту. Параметры выбираются с помощью *QRadioButton* – переключатель с текстом. Указать «плавающую» дельту можно с помощью *QCheckBox –* элемент графического интерфейса, у которого есть два состояния: включено и выключено (установлено и не установлено). При установке *QCheckBox* блокируется ввод значения дельты.

Для обработки пользователю нужно нажать на кнопку «обработать». После нажатия происходит проверка: выбраны ли минимум два эксперимента. По умолчанию выбран параметр «Координата X» для привязки и усреднения. Если проверка не прошла, то пользователь получает соответствующее сообщение.

Далее происходит создание динамического запроса в базу данных, и вывод информации в *QTableView*, за которым закреплена модель *QStandardItemModel*, которая связывается с *QTableView* посредством вызова методов:

– QStandardItemModel::QStandardItemModel(int rows, int columns, QObject \*parent = nullptr) – конструктор, в который передается количество строк и столбцов;

– void QTableView::setModel(QAbstractItemModel \*model) – метод, в который передается указатель на модель.

Все изменения в модели отображаются на связанном с ней экземпляром класса *QTableView.*

Алгоритм создания динамического запроса и выдачи результатов представлен на рис. 12 и рис. 13 [10 – 12].



Рис. 12. Схема алгоритм обработки и выдачи информации (стр.1)



Рис. 13. Алгоритм обработки и выдачи информации (стр.2)

Для усреднения выбирается основной эксперимент – первый эксперимент в списке выбранных экспериментов. Количество строк таблицы результата обработки равно количеству строк параметров основного эксперимента. Алгоритм проходится по каждой строке параметров основного эксперимента и находит соответствующие значения привязки, в зависимости от дельты, в таблице параметров следующего эксперимента в списке выбранных, затем складывает параметры усреднения и привязки, и увеличивает счетчик совпадений на единицу. После прохода по всем строкам параметров не основного эксперимента, сохраняет параметры в *model3* – модель таблицы обработанной информации. В конце, после прохода по всем параметрам всех выбранных экспериментов, происходит усреднение значений в *model3* и вывод результатов в табличном виде.

### 3.3.2. Фильтр базы данных

Для удобства нахождения нужного эксперимента в программе предусмотрен фильтр по дате и названию. Фильтр состоит из *QCalendarWidget* – виджет[[3]](#footnote-3), который предоставляет пользователю доступ к календарю, двух *QRadioButton*, с помощью которых можно задать интервалы даты, двух *QDateTimeEdit* для задания дат и времени вручную, *QLineEdit* для задания названия эксперимента и кнопки *QPushButton*. *QDateTimeEdit –* виджет, предназначенный для редактирования даты и времени.

При нажатии на кнопку «применить» формируется часть запроса в базу данных, которая содержится после слова «WHERE» и на модели таблицы устанавливается фильтр с помощью функции void QSqlTableModel::setFilter(const QString &filter), в которую передается эта строка. После этого обновляется таблица.

### 3.3.4. Построение графиков

Для построения графиков был создан отдельный класс *MyGraphicView,* унаследованный от *QGraphicsView –* предоставляет виджет для отображения содержимого *QGraphicsScene*. *QGraphicsScene –* предоставляет поверхность для управления большим числом графических 2D элементов. Описание класса приведено в табл. 3.

Таблица 3

Описание класса *MyGraphicView.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| MyGraphicView | public | QGraphicsView | QWidget\*parent | Конструктор |
| scene | private | QGraphicsScene\* | - | Указатель на сцену для рисования элементов |
| group\_1,  group\_2 | private | QGraphicsItemGroup\* | - | Указатель на группы элементов |
| timer | private | QTimer\* | - | Указатель на таймер |
| data\_time | private | QVector<QDateTime>\* | - | Динамический массив для хранения дат и времен |
| data\_param | private | QVector<QString> \* | - | Динамический массив для хранения значений параметров |
| data\_flag | private | bool | - | Флаг, который устанавливается в логическую единицу при вызове метода SetData |
| count\_data | private | int | - | Содержит количество данных |
| difDate | private | qint64 | - | Разница между самой ранней и поздней датами в секундах |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| zero, zero2 | private | int | - | Начальное значение времени |
| max, max2 | private | double | - | Максимальное значение параметров по модулю |
| param\_str, param\_str2 | private | QString | - | Название параметра |
| no\_time | private | bool | - | Флаг установки данных без зависимости от времени |
| vect\_par1, vect\_par2 | private | QVector <double>\* | - | Динамический массив для хранения значений параметров без зависимости от времени |
| resizeEvent | private | void | QResizeEvent \*event | Перехватывает событие изменение размера виджета |
| deleteItemsFromGroup | private | void | QGraphicsItemGroup \*group | Удаляет группу элементов |
| eventFilter | protected | bool | QObject \* obj , QEvent \* event | Перехватывает события от колесика мыши и изменяет масштаб |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод или свойство | Модификатор доступа | Тип | Аргументы | Описание |
| no\_time\_graph | private | void | - | Метод построения графика по значениям, полученным в результате обработки (не зависящим от времени) |
| setData | public | void | QVector<QDateTime> \*datetime,QVector<QString> \*param,QString par\_str | Устанавливает данные (зависимость параметра от времени) |
| setData (перегружен) | public | void | QVector<double> \*vect\_param1, QVector<double> \*vect\_param2,QString par\_str, QString par\_str2 | Устанавливает данные (зависимость параметра от другого параметра) |
| slotAlarmTimer | private | void | - | Слот для обработчика переполнения таймера. В нем производится перерисовка графика. |

Для того, чтобы построить график зависимости параметра от времени привязки пользователю необходимо выбрать параметр(ы), эксперимент и указать номер объекта. Затем нажать на кнопку «построить график».

Для указания параметров используется *QCheckBox*. При нажатии на *QCheckBox* заполняется массив логического типа *check\_graph[4]*, где: *true* – параметр выбран, *false* – параметр не выбран.

Алгоритм построения графиков сначала производит проверку: выбрал ли пользователь эксперимент, указал ли параметр(ы) и номер объекта.

После проверки производится запрос в базу данных «SELECT» выборки столбцов даты и выбранного параметра с фильтром по номеру эксперимента и номеру объекта, упорядоченные по дате. Затем создается новый элемент типа *QVector<MyGraphicView\*>,* в котором содержатся указатели на экземпляры класса *MyGraphicView*, и заполняются две структуры типа *QVector<QVector<QDateTime>\*>* и *QVector<QVector<QString>\*>*, в них содержатся указатели на динамические массивы *QVector<QDateTime>* и *QVector<QString>*, в которых содержатся значения даты и параметров, полученных после выполнения запроса в базу данных. Если нужно построить график по результатам обработки, то заполняются две структуры типа *QVector<QVector<double>\*>,* в них содержатся указатели на динамические массивы *QVector< double >*. Таким образом реализовано добавление любого количества графиков.

Метод *SetData* перегружен, это нужно для того, чтобы устанавливать параметры различных типов: первый метод принимает *QVector<QDateTime>\** и *QVector<QString>\** – зависимость выбранного параметра от времени, и аргумент типа *QString* – название параметра, второй принимает два аргумента типа QVector<double> \* – зависимость параметра от другого параметра и два аргумента *QString* – названия параметров.

Алгоритм метода *SetData,* кроме инициализации данных, определяет интервал времени и записывает его в свойство *difDate,* а также находит максимальное значение параметров по модулю, и устанавливает свойство *data\_flag* в логическую единицу, а *no\_time* в логический ноль. Если вызван перегруженный *SetData,* то алгоритм находит два максимальных значения по модулю, устанавливает флаг *no\_time* и *data\_flag* в логическую единицу.

Когда создается экземпляр класса *MyGraphicView,* начинает отчет таймер, установленный на 50 миллисекунд. После переполнения таймера вызывается слот *slotAlarmTimer()*, в котором происходит прорисовка графика. Если таймер переполнится раньше, чем установятся данные, то свойство *data\_flag* не позволит произвести прорисовку. Алгоритм прорисовки автоматически подбирает масштаб исходя из временных параметров, и определяет размерность времени: секунды, минуты или часы. Если флаг *no\_time* установлен в логическую единицу, то *slotAlarmTimer()* вызывает метод *no\_time\_graph().* Метод *no\_time\_graph()* аналогичен методу *slotAlarmTimer()*, с тем отличием, что *no\_time\_graph()* не делает поправку на время, и ось ординат находится посередине, так как значения на оси абсцисс могут быть отрицательными. Схема алгоритма метода *slotAlarmTimer()* представлена на рис. 14, рис. 15 и рис. 16. Вначале алгоритма идет инициализация сцены и делителя сцены. Делитель сцены нужен для упрощения построения графиков. Далее идет построение осей. После этого расчет точек на оси абсцисс и времени для каждой точки, одновременно с этим строятся вертикальные линии сетки. Время выбирается в зависимости от разности самой поздней и самой ранней даты и времени в секундах. Затем алгоритм рассчитывает точки на оси ординат и значения в этих точках. Расчет на оси ординат зависит от количества точек и максимального значения параметров по модулю. В конце проставляются названия осей, на оси абсцисс также в зависимости от разницы дат, на оси ординат в соответствии с принятым параметром [13, 14].



Рис. 14. Схема алгоритма метода *slotAlarmTimer()* (стр. 1)



Рис. 15. Схема алгоритма метода *slotAlarmTimer()* (стр. 2)



Рис. 16. Схема алгоритма метода *slotAlarmTimer()* (стр. 3)

Данный класс может строить графики не только с зависимостью параметра от времени, но и с зависимостью параметра от другого параметра, с помощью метода *no\_time\_graph()*. Этот метод аналогичен методу ***slotAlarmTimer()*, отличается построением оси абсцисс. Ось ординат находится посередине, так как значения на оси абсцисс могут быть отрицательными. Для того, чтобы установить зависимость параметра от другого параметра необходимо вызвать перегруженный метод *SetData ( QVector < double > \* vect\_param1 , QVector < double > \*vect\_param2, QString par\_str, QString par\_str2)*. Его аргументы принимают два вектора (динамических массива) типа double, и названия параметров. Перегруженный метод *SetData* также высчитывает разницу между максимальным и минимальным значением параметров и сохраняет в *max* и *max2* для одного вектора и другого. Этот метод устанавливает *no\_time* в логическую единицу, тем самым блокирует выполнение метода *slotAlarmTimer()* и вызывает метод *no\_time\_graph()*.**

**Расчет координат точек графика на сцене в методе *no\_time\_graph()* происходит следующим формулам:**

, где X1,2 – координата X первой и второй точек на сцене;

Y1,2 – координата Y первой и второй точек на сцене;

width – ширина сцены;

height – высота сцены;

n – делитель ширины сцены;

max, max2 – разности между максимальными и минимальными значениями параметров для vect\_par1 и vect\_par2 соответственно.

x1,2, y1,2 – соседние значения параметров на каждом шаге.

**Расчет координат точек графика на сцене в методе *slotAlarmTimer()* происходит следующим формулам:**

, где X1,2 – координата X первой и второй точек на сцене;

Y1,2 – координата Y первой и второй точек на сцене;

width – ширина сцены;

height – высота сцены;

n – делитель ширины сцены;

max – разность между максимальным и минимальным значениями параметров.

difDate\_res – разность между самой ранней и самой поздней датой со временем в секундах.

x1,2, y1,2 – соседние значения параметров на каждом шаге.

Здесь x1 и x2 предварительно делятся на коэффициент времени в зависимости от difDate\_res:

– если difDate\_res<60, то x1,2 = x1,2;

– если difDate\_res/60>60, то x1,2 = x1,2/360;

– если difDate\_res/60<60, то x1,2 = x1,2/60;

Таким образом происходит преобразование значений параметров в координаты линии на сцене. Линия графика рисуется красным цветом, линии осей – черным, линии сетки – серым. Результат построения графика представлен на рис. 17.

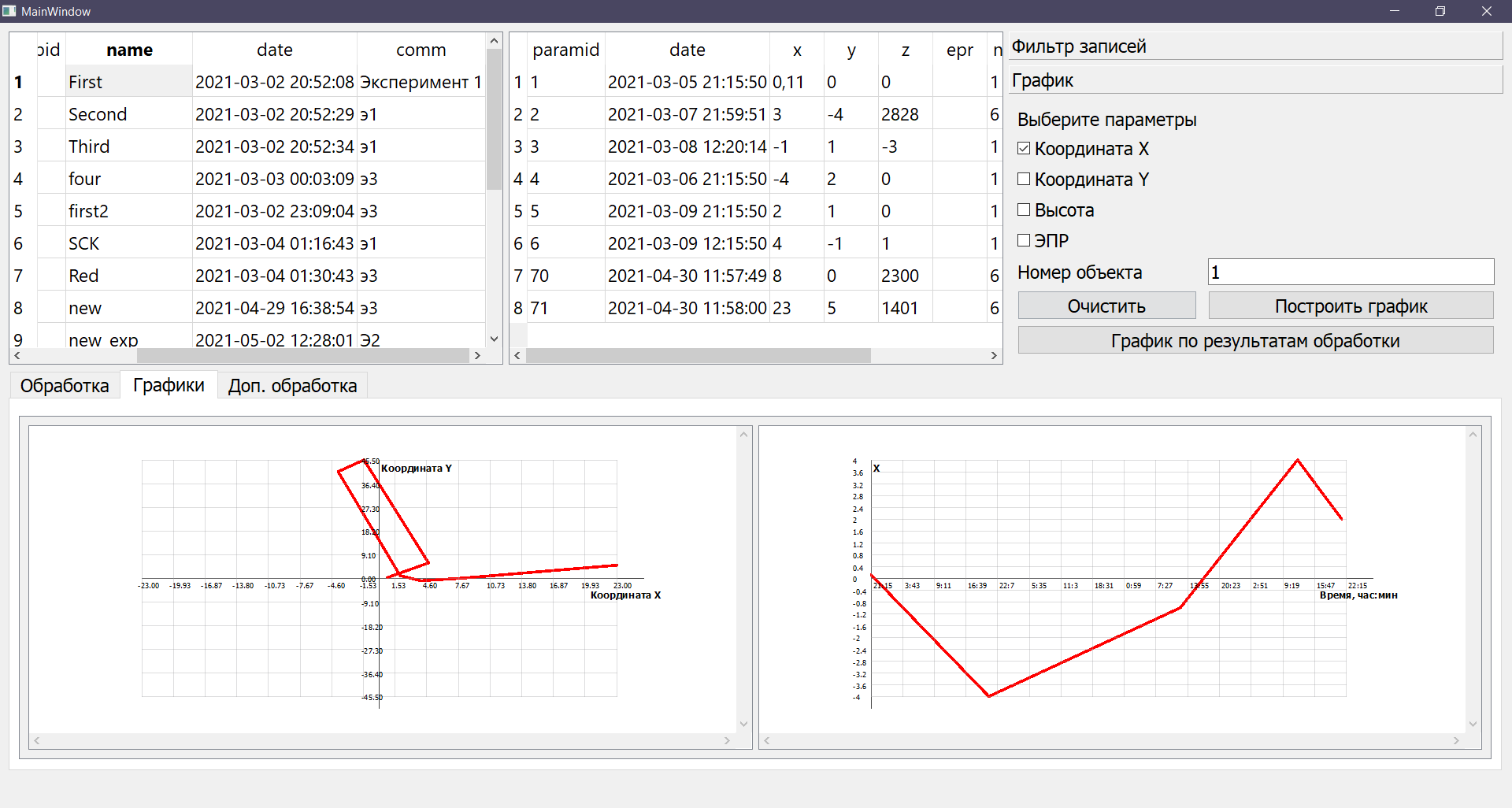


Рис. 17. Результат построения графика (Левый график – график по результатам обработки, правый – зависимость параметра “Координата X” от времени, эксперимент “First”, номер объекта 1)

Чтобы построить график по результатам обработки модельных экспериментов нужно сначала провести обработку, затем нажать на кнопку «График по результатам обработки».

Как только происходит изменение размера, то вызывается метод *resizeEvent(QResizeEvent \*event)*, который запускает таймер на 10 миллисекунд, и после переполнения происходит повторная прорисовка графика.

Когда происходит событие колёсика мыши, то вызывается метод *eventFilter(QObject \*watched, QEvent \*event)*, который изменяет масштаб сцены *QGraphicsScene*.

Графики добавляются в *QGridLayout* – класс, который выравнивает виджеты по сетке. Для удобства чтения информации с графиков в одной строке *QGridLayout* содержится не более двух графиков.

Чтобы удалить все графики необходимо нажать на кнопку «очистить». Алгоритм обработки нажатия этой кнопки удаляет содержимое структур типов QVector < MyGraphicView \*>, *QVector< QVector < QDateTime >\*>*, *QVector < QVector < QString > \*>* и *QVector<double>\**, и очищает занятую этими структурами память*.*

# 4. ТЕСТИРОВАНИЕ ПРОГРАММ

Для оценки быстродействия программ используется встроенная библиотека *QTimer*, которая позволяет засечь время моделирования и время обработки параметров. Результат отображается с помощью *QDebug* – класс, который предоставляет выходной поток для отладочной информации. Время засекается с момента нажатия на кнопку и до мометра выхода программы из слота обработки нажатия кнопки.

Согласно стандарта ГОСТ 19.301-79 испытания проводились по следующей схеме:

1. Объект испытаний – программа обработки результатов модельных экспериментов и программа заполнения базы данных.
2. Цель испытаний – выявление ошибок, проверка работоспособности программ, оценка быстродействия.
3. Средства испытаний.

Состав используемых во время испытаний технических средств:

**– Ноутбук Lenovo Y500 с процессором Intel Core i7;**

**– ОЗУ 16 Гбайт;**

**– 4 Гбайт видеопамяти;**

**– наличие свободного места на жестком диске более 200 Мбайт;**

1. Порядок испытаний. Сначала тестируется правильность вывода результатов моделирования, затем корректность введенной информации, и правильность вычислений с «плавающей» и константной (заданной пользователем) дельтой, затем производится оценка быстродействия программ: засекается время моделирования и время обработки результатов модельных экспериментов.
2. Методы испытаний. **Испытания проводятся методом функционального тестирования.**

**Тесты предназначены для отладки функций программ и выявления следующих ситуаций:**

**– пользователь не ввел название эксперимента при моделировании параметров;**

**– пользователь не ввел необходимые данные при ручном заполнении;**

**– пользователь выбрал меньше двух экспериментов для обработки;**

**– произведена попытка удалить запись без её выбора;**

**– произведена попытка построить график по недостаточным данным.**

**Каждый тест характеризуется своим набором входных данных. Когда происходит одна из вышеперечисленных ситуаций происходит информирование пользователя о соответствующей ошибке с помощью диалогового окна (пример на рис. 18). Все тесты проведены успешно, новых ошибок не обнаружено.**

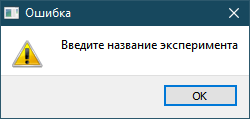


Рис. 18. Пример информирования пользователя об ошибке

Время моделирования параметров воздушного объекта зависит только от количества записей, поэтому оценка быстродействия моделирования производится на любом наборе начальных параметров, с изменением количества записей. Количество записей меняется от 1 до 10 с шагом 1, от 10 до 100 с шагом 10, от 100 до 1000 с шагом 100, от 1000 до 10000 с шагом 1000. Зависимость времени моделирования от количества записей представлена на рис. 19.

Рис. 19. Зависимость времени моделирования от количества записей

При моделировании параметров воздушного объекта количество записей обычно не превышает 1000. На основании полученных результатов можно сделать вывод, что моделирование параметров движения воздушного объекта происходит быстро для пользователя (меньше 10 секунд).

Время обработки результатов модельных экспериментов зависит от нескольких параметров: количество выбранных экспериментов, количество записей в каждом эксперименте, количество записей в основном эксперименте. При обработке результатов модельных экспериментов чаще всего выбирается меньше 10 экспериментов и меньше 1000 записей в каждом эксперименте. Исходя из этого оценим время обработки на следующих входных данных: количество выбранных экспериментов меняется от 2 до 10 с шагом 1, количество записей во всех экспериментах равно 1000, параметр привязки – координата X, параметр усреднения – ЭПР, дельта равна 100. Зависимость времени обработки от количества выбранных экспериментов представлена на рис 20.

Рис. 20. Зависимость времени обработки от количества выбранных экспериментов

На основании полученных результатов можно сделать вывод, что обработка результатов модельных экспериментов происходит практически мгновенно для пользователя (меньше 1 секунды).

В ходе использования и тестирования программы обработки результатов модельных экспериментов можно выделить часто повторяющийся ряд действий, а именно задание фильтра по экспериментам, выбор экспериментов и обработка информации. Для выполнения этих действий потребуется установить параметры фильтра во вкладке «Фильтр записей» (при запуске программы открыта именно эта вкладка) верхнего окна, нажать кнопку «применить» в той же вкладке, выбрать эксперимент, нажать на кнопку «Добавить» во вкладке «Обработка» нижнего окна (при запуске открыта), повторить выбор и нажатие кнопки «Добавить» для выбора остальных экспериментов, затем указать параметры обработки и нажать кнопку «Обработать» во вкладке «Обработка» нижнего окна. Все эти действия не требуют перехода на другие вкладки программы. В случае, если необходимо провести дополнительную обработку, или построить график необходимо переключиться на другие вкладки. В этом случае пользователю станут открытыми именно те блоки данных, с которыми он будет взаимодействовать, и не будет мешать лишняя информация. В программе заполнения базы данных все действия происходят в одном окне, и информация из базы данных всегда доступна пользователю. Таким образом, достигается максимальное удобство использования программ.

# ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы была поставлена цель произвести селекцию движущейся цели по модельным экспериментам.

Для достижения поставленной цели разработана база данных с соответствующей структурой. Разработана программа заполнения базы данных. Заполнять базу данных можно, как и вручную, так и посредством моделирования по начальным параметрам векторов положения и скорости воздушного объекта. Также реализован вывод в файл модельных параметров, и заполнение базы данных из файла. Разработана программа для обработки результатов модельных экспериментов. Объём? В неё входит сама обработка модельных экспериментов – усреднение значений параметров по нескольким экспериментам в заданном диапазоне, дополнительная обработка – усреднение одного параметра по выбранным экспериментам, построение графиков различных зависимостей (параметра от даты и времени и параметра от другого параметра). Для программ разработан интуитивно понятный пользовательский интерфейс. Он предназначен для сотрудников ПАО «НПО «Алмаз».

Интерфейс даёт возможность быстро и с легкостью обработать результаты экспериментов, оценить обработку, произвести моделирование по заданным параметрам, а также произвести дополнительную обработку.

Произведено тестирование разработанных программ, результаты которого показали, что разработанный пользовательский интерфейс отзывчив и корректно обрабатывает данные, вводимые пользователем.

Так как разработанная система использует кроссплатформенное и свободно

распространяемое программное обеспечение, то она может быть развернута на

различных платформах.

Сформированная в ТЗ цель достигнута. Разработанная программа позволит быстро и удобно производить селекцию движущейся цели – произвести обработку результатов модельных экспериментов и наглядно увидеть результат.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ПАО «НПО «Алмаз».: сайт. – URL: [www.raspletin.com](http://www.raspletin.com) (дата обращения 01.06. 2021). – Текст: электронный.
2. Qt (software) – Wikipedia.: сайт – URL: [www.wikipedia.org/wiki/Qt\_(software)](http://www.wikipedia.org/wiki/Qt_(software)) (дата обращения 31.05.2021). – Текст: электронный.
3. SQLite. сайт: – URL [www.younglinux.info/sqlite/sqlite](http://www.younglinux.info/sqlite/sqlite) (дата обращения 31.05.2021). – Текст: электронный.
4. Штофф В.А. Моделирование и философия. – М. Л.: Наука, 1966. 295 с.
5. Qt Plotting Widget QCustomPlot – Introduction.: сайт. – URL: www.qcustomplot.com (дата обращения: 07.05.2021). – Текст: электронный.
6. Qwt User's Guide: Qwt – Qt Widgets for Technical Applications.: сайт. – URL: qwt.sourceforge.io (дата обращения 07.05.2021). – Текст: электронный.
7. Сигналы и слоты в Qt. – URL: www.habr.com/ru/post/50812 (дата обращения 07.05.20210). – Текст: электронный.
8. Базовые контейнеры. сайт: – URL: [www.doc.crossplatform.ru/qt/4.3.2/containers.html](http://www.doc.crossplatform.ru/qt/4.3.2/containers.html) (дата обращения 01.06.2021). – Текст: электронный.
9. Сихарулидзе Ю.Г. Баллистика летательных аппаратов. — М.: Наука, 1982. — 352 с.
10. Прата С. Язык программирования C++ (C++11). Лекции и упражнения. - 6 изд. — М.: Вильямс, 2012. — 1248 с.
11. Страуструп Б. Язык программирования С++. Специальное издание. Пер. с англ. — М.: Издательство Бином, 2011 г. — 1136 с.
12. Лафоре Р. Объектно-ориентированное программирование в С++. Спб.: Питер, 2018. – 928 с.
13. Шлее М. Qt 4.8. Профессиональное программирование на C++ . — Спб.: БХВ-Петербург, 2012. — 894 с.
14. Боровский А.Н. Qt4.7+. Практическое программирование на C++. — СПб.: БХВ-Петербург, 2012. — 496 с.

# *Приложение А. Техническое задание*

|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТ |
| **Кафедра** | ВМСС |

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (бакалаврскую работу)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Направление** | | | 09.03.01 Информатика и вычислительная техника | | | |
| (код и наименование) | | | | | | |
| **Направленность (профиль)** | | | | | | Вычислительные машины, комплексы, |
| системы и сети | | | | | | |
|  | | | | | | |
| **Форма обучения** | | | | | очная | |
|  | (очная/очно-заочная/заочная) | | | | | |
| **Тема:** | Разработка программы обработки результатов модельных экспериментов | | | | | |
|  | | | | | | |
| **Студент** | | А-08-17 Полушин А.И. | | | | |
| группа подпись фамилия и инициалы | | | | | | |
| **Научный руководитель** | | | | д.т.н. профессор  Абросимов Л.И. | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | |
| **Консультант** | | | | к.т.н. зам. нач. СКБ-230 НТЦ “НИИРП” ПАО “НПО”  Алмаз”Фащев В.В. | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | |
| **Зав. Кафедрой** | | | | к.т.н доцент Вишняков С.В. | | |
| уч. степень звание подпись фамилия и инициалы | | | | | | |
|  | | | | | | |
| **Место выполнения работы** | | | | | | “НИИРП” ПАО “НПО” Алмаз” |
|  | | | | | | |

**СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ**

Суть модельного эксперимента состоит в том, чтобы фиксировать параметры объекта в некоторой области трехмерного пространства.

Для выполнения выпускной квалификационной работы необходимо:

1. Разработать подпрограмму заполнения базы данных.

2. По каждому эксперименту создается своя таблица в БД. Разрабатываемая программа должна предоставлять пользователю возможность задания условий обработки результатов (возможность выбора таблиц с результатами экспериментов и задание оцениваемого параметра). Например:

а) для каждого из выполненных N экспериментов записывается набор данных ( номер объекта, время привязки информации, положение объекта в пространстве),

б) далее пользователь задает, что ему нужно получить усредненные оценки по каждому объекту с привязкой к высоте.

в) разрабатываемая программа должна найти для каждого объекта во всех таблицах одно и тоже положение объекта (положение различается на некую дельту) и усреднить заданный параметр.

В программе предусмотреть возможность проводить дополнительную обработку результатов, например, получить среднее значение высоты по какому-либо эксперименту, по группе экспериментов.

Дополнительно можно построить графики, показывающие изменение выбранной характеристики от времени.

В качестве базы данных использовать SQLite.

Саму БД можно заполнять из текстового файла, либо создать приложение, которое будет имитировать объекты и их характеристики.

Для алгоритмов обработки и создания приложения использовать фреймворк Qt C++.

**ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА**

1. Схема алгоритма выдачи результатов
2. Пользовательский интерфейс приложения
3. Структура базы данных
4. Схема алгоритма заполнения базы данных

|  |  |
| --- | --- |
| **Количество листов** | 4 |
|  |  |
| **Количество слайдов в презентации** | не менее 8 |
|  |  |

**РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА**

1. Прата С. Язык программирования C++ (C++11). Лекции и упражнения. - 6 изд. — М.: Вильямс, 2012. — 1248 с.
2. Страуструп Б. Язык программирования С++. Специальное издание. Пер. с англ. — М.: Издательство Бином, 2011 г. — 1136 с.
3. Шлее М. Qt 4.8. Профессиональное программирование на C++ . — М.: БХВ-Петербург, 2012. — 894 с.
4. Боровский А.Н. Qt4.7+. Практическое программирование на C++. — СПб.: БХВ-Петербург, 2012. — 496 с.

***Примечания:***

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

# *Приложение Б. Листинг заголовочного файла mainwindow1.h.*

**/\*Описание полей и методов класса MainWindow1**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

**/\*Описание импорта необходимых библиотек**

**\*Разработал студент А-08-17 Полушин А.**

**\*Дата написания 01.03.2021**

**\*Версия 1\*/**

#include <QMainWindow>

#include <QtSql>

#include <QDebug>

#include <QFileInfo>

#include <QSqlTableModel>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QMessageBox>

#include <prognoz.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = NULL);

~MainWindow();

**/\*Описание слотов обработки событий: нажатия на кнопки, \*нажатие на QTableView, изменение индекса в QTableView, \*нажатие на QComboBox**

**\*Разработал студент А-08-17 Полушин А.**

**\*Дата написания 01.03.2021**

**\*Версия 1\*/**

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_6\_clicked();

void on\_pushButton\_5\_clicked();

void on\_pushButton\_7\_clicked();

void on\_pushButton\_8\_clicked();

void on\_tableView\_activated(const QModelIndex &index);

void table1\_change(const QModelIndex &index);

void table2\_change(const QModelIndex &index);

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

void on\_pushButton\_9\_clicked();

void on\_pushButton\_10\_clicked();

void on\_comboBox\_2\_activated(int index);

private: //Объявление полей и методов класса MainWindow1 с модификатором доступа private

Ui::MainWindow \*ui;

QSqlDatabase db; //Соединение с базой данных

//Указатели на модели таблиц

QSqlTableModel \*model, \*model2;

//Методы обновления данных в таблицах

void update\_table1();

void update\_table2();

QString exp\_id;//Id выбранного эксперимента

//Множества индексов первой и второй таблицы

QSet<QModelIndex> sind1;

QSet<QModelIndex> sind2;

//Экземпляр класса prognoz(моделирование параметров)

prognoz p;

};

#endif // MAINWINDOW\_H

# *Приложение В. Листинг файла mainwindow1.cpp.*

**/\*Реализация методов класса построения графиков**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

//Подключение необходимых библиотек

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QSetIterator>

#include <QSqlError>

#include <QSqlRecord>

#include "QStandardItemModel"

#include "QStandardItem"

#include <QFileDialog>

**/\*Реализация конструктора.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Результат: инициализация полей класса построения графиков**

**\*Версия 1\*/**

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

db = QSqlDatabase::addDatabase("QSQLITE");

db.setDatabaseName("..//..//data//exp.db");

QDateTime time = QDateTime::currentDateTime();

ui->dateTimeEdit->setDateTime(time);

ui->progressBar->hide();

if(!db.open())

{

qDebug()<<db.lastError().text();

return;

}

exp\_id="0";

//Заполнение QComboBox

ui->comboBox\_2->addItem("Тип 1",1);

ui->comboBox\_2->addItem("Тип 2",2);

ui->comboBox->addItem("Подтип 1",1);

ui->comboBox->addItem("Подтип 2",2);

ui->comboBox->addItem("Подтип 3",3);

//Обновление таблиц

update\_table1();

update\_table2();

}

//Деструктор

MainWindow::~MainWindow()

{

db.close();

delete ui;

}

**/\*Обработка нажатия на кнопку "применить"**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные:**

**\*множества индексов двух таблиц sind1, sind2;**

**\*Результат: Если нет пустых ячеек, то применеие изменений в базе данных**

**\*если есть пустые ячейки, то вывод сообщения с ошибкой**

**\*Метод решения: перебор элементов.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_clicked()

{

bool flag1=false;//флаг состояния

QSetIterator<QModelIndex> inc1(sind1),inc2(sind2);

while (inc1.hasNext() && !flag1)

{

if(ui->tableView->model()->data(inc1.next())=="")

{

flag1=true;

}

}

while (inc2.hasNext() && !flag1)

{

if(ui->tableView\_2->model()->data(inc2.next())=="")

{

flag1=true;

}

}

if(flag1)

{

QMessageBox::warning(this, "Ошибка","Значения ячеек не должны быть пустыми");

update\_table1();

update\_table2();

} else

{

model->submitAll();

sind1.clear();

model2->submitAll();

sind2.clear();

update\_table1();

update\_table2();

}

}

**/\*Обработка нажатия на кнопку "обновить"**

**\*Разработал студент А-08-17 Полушин А.**

**\*Результат: обновление двух таблиц**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_2\_clicked()

{

update\_table1();

update\_table2();

}

**/\*Обработка события нажатия на таблицу 1 (Эксперименты)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные:**

**\*Id эксперимента exp\_id**

**\*Результат: изменение Id эксперимента и обновление таблицы 2 (параметры)**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_clicked(const QModelIndex &index)

{

int row = index.row();

int id = index.sibling(row, 0).data().toInt();

exp\_id=QString::number(id);

update\_table2();

}

**/\*Метод обновления таблицы 2 (параметры)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные:**

**\*Модель второй таблицы model2**

**\*Результат: обновление таблицы 2**

**\*Метод решения: установка фильтра и вызов метода select() у модели model2**

**\*Версия 1\*/**

void MainWindow::update\_table2()

{

model2 = new QSqlTableModel(this,db);

model2->setEditStrategy(QSqlTableModel::OnManualSubmit);

model2->setTable("param");

model2->setFilter("expid="+exp\_id);

model2->select();

ui->tableView\_2->setModel(model2);

ui->tableView\_2->resizeColumnsToContents();

ui->tableView\_2->resizeRowsToContents();

//Сопоставление сигнала изменения индекса и слота изменения таблицы

connect(ui->tableView\_2->selectionModel(),SIGNAL(currentChanged(QModelIndex,QModelIndex)),this,SLOT(table2\_change(QModelIndex)));

ui->tableView\_2->hideColumn(0);

ui->tableView\_2->hideColumn(ui->tableView\_2->model()->columnCount()-1);

}

**/\*Метод обновления таблицы 1 (эксперименты)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные:**

**\*Модель второй таблицы model**

**\*Результат: обновление таблицы 1**

**\*Метод решения: вызов метода select() у модели model**

**\*Версия 1\*/**

void MainWindow::update\_table1()

{

model = new QSqlTableModel(this,db);

model->setEditStrategy(QSqlTableModel::OnManualSubmit);

model->setTable("experiment");

model->select();

ui->tableView->setModel(model);

ui->tableView->resizeColumnsToContents();

ui->tableView->hideColumn(0);

//Сопоставление сигнала изменения индекса и слота изменения таблицы

connect(ui->tableView->selectionModel(), SIGNAL(currentChanged(QModelIndex,QModelIndex)), this,SLOT(table1\_change(QModelIndex)));

}

**/\*Обработка нажатия на кнопку "Добавить строку" (1 таблица)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов 1 таблицы sind1**

**\*Результат: добавление строки в 1-ю таблицу с заполненными по умолчанию \*значениями,**

**\*добавление индексов в sind1**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_6\_clicked()

{

model->insertRow(model->rowCount());

QModelIndex index = ui->tableView->model()->index(ui->tableView->model()->rowCount()-1,ui->tableView->model()->columnCount()-2);

QString time\_format = "yyyy-MM-dd HH:mm:ss";

QDateTime cur\_date = QDateTime::currentDateTime();

QString str\_cur\_date = cur\_date.toString(time\_format);

ui->tableView->model()->setData(index,str\_cur\_date);

QModelIndex ind = ui->tableView->model()->index(ui->tableView->model()->rowCount()-1,1);

sind1.insert(ind);

}

**/\*Обработка нажатия на кнопку "Добавить строку" (2 таблица)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов 2 таблицы sind2**

**\*Результат: добавление строки во 2-ю таблицу с заполненными по умолчанию \*значениями,**

**\*добавление индексов в sind2**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_5\_clicked()

{

//Выбран эксперимент или нет?

if(QString::compare(exp\_id,"")!=0 && QString::compare(exp\_id,"0")!=0)

{

model2->insertRow(model2->rowCount());

QModelIndex index = ui->tableView\_2->model()->index(ui->tableView\_2->model()->rowCount()-1,ui->tableView\_2->model()->columnCount()-1);

ui->tableView\_2->model()->setData(index,exp\_id);

QDateTime time=QDateTime::currentDateTime();

index = ui->tableView\_2->model()->index(ui->tableView\_2->model()->rowCount()-1,ui->tableView\_2->model()->columnCount()-7);

ui->tableView\_2->model()->setData(index,time.toString("yyyy-MM-dd HH:mm:ss"));

for(int i=1;i<=ui->tableView\_2->model()->columnCount();i++)

{

index = ui->tableView\_2->model()->index(ui->tableView\_2->model()->rowCount()-1,ui->tableView\_2->model()->columnCount()-i);

if (i>2&&i<ui->tableView\_2->model()->columnCount()-1)

ui->tableView\_2->model()->setData(index,0);

if(i==2)

ui->tableView\_2->model()->setData(index,1);

//sind2.insert(index);

}

ui->tableView\_2->resizeColumnsToContents();

} else

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

}

**/\*Обработка нажатия на кнопку "Удалить строку" (1 таблица)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов 1 таблицы sind1**

**\*Результат: удаление строки из 1-ой таблицы,**

**\*удаление индексов из sind1**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_7\_clicked()

{

QMessageBox msgBox;

msgBox.setWindowTitle("Внимание");

msgBox.setText("При удалении эксперимента удалятся все параметры, связанные с ним");

msgBox.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);

msgBox.setIcon(QMessageBox::Information);

msgBox.setDefaultButton(QMessageBox::Ok);

int res = msgBox.exec();

if (res == QMessageBox::Ok) //нажата кнопка Ok

{

QSqlQuery query = QSqlQuery(db);

if(!query.exec("delete from param where expid="+exp\_id))

{

qDebug() << query.lastError().databaseText();

qDebug() << query.lastError().driverText();

} else

{

int selectrow = ui->tableView->currentIndex().row();

if(selectrow>=0)

{

model->removeRow(selectrow);

} else

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

}

}

}

**/\*Обработка нажатия на кнопку "Удалить строку" (2 таблица)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов 2 таблицы sind2**

**\*Результат: удаление строки из 2-ой таблицы,**

**\*удаление индексов из sind2**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_8\_clicked()

{

int selectrow = ui->tableView\_2->currentIndex().row();

if(selectrow>=0)

{

model2->removeRow(selectrow);

qDebug() << db.lastError().text();

} else

QMessageBox::warning(this, "Ошибка","Выберите запись");

}

**/\*Обработка активации (например, с помощью кнопок клавиатуры) первой \*таблицы**

**\*Разработал студент А-08-17 Полушин А.**

**\*Результат: вызов метода on\_tableView\_clicked(index)**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_activated(const QModelIndex &index)

{

on\_tableView\_clicked(index);

}

**/\*Обработка изменения выбора ячейки в 1 таблице**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов sind1**

**\*Результат: добавление индекса выбранной ячейки в sind1**

**\*Версия 1\*/**

void MainWindow::table1\_change(const QModelIndex &index)

{

if(index.column()!=3)

sind1.insert(index);

}

**/\*Обработка изменения выбора ячейки во 2 таблице**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: множество индексов sind2**

**\*Результат: добавление индекса выбранной ячейки в sind2**

**\*Версия 1\*/**

void MainWindow::table2\_change(const QModelIndex &index)

{

sind2.insert(index);

}

**/\*Обработка нажатия на кнопку "Ввод из файла"**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: строки ввода lineedit и lineedit2**

**\*(название и комментарий эксперимента)**

**\*Результат: добавление нового эксперимета и его параметров из файла**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_3\_clicked()

{

//Проверка на ввод названия эксперимента

if(ui->lineEdit->text()=="")

{

QMessageBox::warning(this, "Ошибка","Введите название эксперимента");

return;

}

//Открытие файла

QFile f(QFileDialog::getOpenFileName(this, "Импорт данных", "..", "TextFile(\*.txt)"));

QString file\_name = f.fileName();

if(file\_name=="")

return;

bool success = true;

//Строки динамических запросов и массива параметров

QString query\_str\_exp,query\_str\_param,column[6];

//Если есть комментарий, то добавление его в запрос

if(ui->lineEdit\_2->text()!="")

{

query\_str\_exp="INSERT INTO experiment (name,date,comm) VALUES (";

query\_str\_exp+="\""+ui->lineEdit->text()+"\",\""+ui->dateTimeEdit->text()+"\",\""+ui->lineEdit\_2->text()+"\")";

} else

{

query\_str\_exp="INSERT INTO experiment (name,date) VALUES (";

query\_str\_exp+="\'"+ui->lineEdit->text()+"\',\'"+ui->dateTimeEdit->text()+"\')";

}

//Инициализация запросов в БД

QSqlQuery query\_exp,query\_param;

query\_exp.prepare(query\_str\_exp);

if(!query\_exp.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

query\_exp.clear();

query\_exp.prepare("SELECT MAX(expid) FROM experiment");

if(!query\_exp.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

query\_exp.next();

QString id\_exp=query\_exp.value(0).toString();

//инициализация массива параметров

for(int i=0;i<column->length();i++)

{

column[i] = "";

}

//Инициализация регулярных выражения для поиска дат и значений параметров

QRegExp reg\_column("\\S+"),

reg\_date("\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}"),

reg\_param("[+-]?([0-9]\*[.])?[0-9]+");

if ((f.exists())&&(f.open(QIODevice::ReadOnly))&&success)

{

//Переменная для хранения строки из файла

QString str\_line="";

if(!f.atEnd() && success)

{

str\_line=f.readLine();

int last\_index = 0, inc = 0;

//Чтение первой строки, содержащей названия параметров

while( ( last\_index = reg\_column.indexIn( str\_line, last\_index ) ) != -1 && inc<6)

{

last\_index += reg\_column.matchedLength();

column[inc]=reg\_column.cap(0);

inc++;

}

reg\_column.indexIn(str\_line);

} else success = false;

int count\_row=0;

//Чтение файла, инициализация динамических запросов и их выполнение

while(!f.atEnd() && success)

{

query\_param.clear();

query\_str\_param = "INSERT INTO param (";

for(int i =0;i<6;i++)

{

if(column[i]!="")

{

if(i!=0)

query\_str\_param +=",";

query\_str\_param += column[i];

}

}

query\_str\_param+=",expid) VALUES (";

str\_line=f.readLine();

QByteArray charArr = str\_line.toLocal8Bit();

char \*c = charArr.data();

//Если нет даты устанавливаем текущую

if(\*c==' ')

{

QDateTime time = QDateTime::currentDateTime();

query\_str\_param+="\'"+time.toString("yyyy-MM-dd HH:mm:ss")+"\'";

int last\_index = 0, inc = 0;

while( ( last\_index = reg\_param.indexIn( str\_line, last\_index ) ) != -1 && inc<6 && success)

{

last\_index += reg\_param.matchedLength();

query\_str\_param+=",";

query\_str\_param+=reg\_param.cap(0);

inc++;

}

//EXPID

query\_str\_param+=","+id\_exp;

} else

{

int last\_index = 0, inc = 0;

if((last\_index = reg\_date.indexIn(str\_line,last\_index))!=-1&&success)

{

query\_str\_param+="\'";

last\_index += reg\_date.matchedLength();

query\_str\_param+=reg\_date.cap(0);

query\_str\_param+="\'";

} else last\_index = 0;

while( ( last\_index = reg\_param.indexIn( str\_line, last\_index ) ) != -1 && inc<6 && success)

{

if(last\_index!=0)

query\_str\_param+=",";

last\_index += reg\_param.matchedLength();

query\_str\_param+=reg\_param.cap(0);

inc++;

}

//EXPID

query\_str\_param+=","+id\_exp;

}

query\_str\_param+=")";

count\_row++;

qDebug()<<query\_str\_exp<<"\n";

qDebug()<<query\_str\_param<<"\n";

query\_param.prepare(query\_str\_param);

if(!query\_param.exec())

{

success=false;

}

qDebug()<<query\_str\_param;

}

qDebug()<<query\_str\_exp;

f.close();

}else//Если файл не существует или нельзя открыть

{

QMessageBox::warning(this, "Ошибка","Не удалось открыть файл");

success=false;

}

//Если произошла ошибка - откат изменений

if(!success)

{

QMessageBox::warning(this, "Ошибка","Импорт не удался");

QSqlQuery query\_delete;

query\_delete.prepare("DELETE FROM param WHERE expid="+id\_exp);

if(!query\_delete.exec())

qDebug()<<query\_delete.lastError().text();

query\_delete.clear();

query\_delete.prepare("DELETE FROM experiment WHERE expid="+id\_exp);

if(!query\_delete.exec())

qDebug()<<query\_delete.lastError().text();

query\_delete.clear();

}

//Обновление таблицы

update\_table1();

}

**/\*Обработка нажатия на кнопку "Установить текущее время"**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: строка ввода даты dateTimeEdit**

**\*Результат: установление текущей даты и времени в строку dateTimeEdit**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_4\_clicked()

{

QDateTime time = QDateTime::currentDateTime();

ui->dateTimeEdit->setDateTime(time);

}

**/\*Обработка нажатия на кнопку "Моделирование"**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: кнопки выбора radioButton\_2, radioButton;**

**\*шкала прогресса progressBar;**

**\*экземпляр класса prognoz - p;**

**\*строка ввода начальной координаты X doubleSpinBox**

**\*строка ввода начальной координаты Y doubleSpinBox\_3**

**\*строка ввода начальной координаты Z doubleSpinBox\_5**

**\*строка ввода проекции вектора начальной скорости на ось X doubleSpinBox\_2**

**\*строка ввода проекции вектора начальной скорости на ось Y doubleSpinBox\_4**

**\*строка ввода проекции вектора начальной скорости на ось Z doubleSpinBox\_6**

**\*строка ввода шага времени doubleSpinBox\_7**

**\*строка ввода количества параметров spinBox**

**\*строка ввода угла локации doubleSpinBox\_8**

**\*QComboBox выбор типа comboBox\_2**

**\*QComboBox выбор подтипа comboBox**

**\*Результат: моделирование параметров**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_10\_clicked()

{

//Инициализация начальных значений

ui->progressBar->setValue(0);

double x, y, z, Vx,Vy,Vz, dt,t,angle,epr;

int type,subtype,N;

x = ui->doubleSpinBox->value();

y = ui->doubleSpinBox\_3->value();

z = ui->doubleSpinBox\_5->value();

Vx = ui->doubleSpinBox\_2->value();

Vy = ui->doubleSpinBox\_4->value();

Vz = ui->doubleSpinBox\_6->value();

dt = ui->doubleSpinBox\_7->value();

N = ui->spinBox->value();

angle = ui->doubleSpinBox\_8->value();

type = ui->comboBox\_2->currentIndex()+1;

subtype = ui->comboBox->currentIndex()+1;

//Если выбрано заполнение БД

if(ui->radioButton\_2->isChecked())

{

QString exp\_name, exp\_comm;

//Если нет названия эксперимента

if(ui->lineEdit->text()=="")

{

//Вывод сообщения об ошибке

QMessageBox::warning(this, "Ошибка","Введите название эксперимента");

return;

}

ui->progressBar->show();

//Установление граничного значения прогресса

ui->progressBar->setRange(0,N);

//Получение названия эксперимента

exp\_name = ui->lineEdit->text();

if(ui->lineEdit\_2->text()!="")

{

exp\_comm = ui->lineEdit\_2->text();

} else exp\_comm ="";

//Создание и выполнение запроса добавления нового эксперимента в БД

QString query\_str\_exp;

query\_str\_exp = "INSERT INTO experiment (name,date";

if(ui->lineEdit\_2->text()!="")

{

query\_str\_exp+=",comm";

}

query\_str\_exp+=") VALUES ('"+ui->lineEdit->text()+"','"+ui->dateTimeEdit->text()+"'";

if(ui->lineEdit\_2->text()!="")

{

query\_str\_exp+=",'"+ui->lineEdit\_2->text()+"'";

}

QDateTime datetime\_const = ui->dateTimeEdit->dateTime(),

datetime = datetime\_const;

query\_str\_exp+=")";

QSqlQuery query;

query.prepare(query\_str\_exp);

query.exec();

query.clear();

query\_str\_exp = "SELECT MAX(expid) FROM experiment";

query.prepare(query\_str\_exp);

query.exec();

query.next();

//Получение Id нового эксперимента

QString id\_exp = query.value(0).toString();

QString query\_str\_param;

query.clear();

//Расчет первого значения ЭПР

if(type==1)

{

epr = p.calcEPR\_TYPE1(subtype);

} else

{

epr = p.calcEPR\_TYPE2(angle,subtype);

}

//моделирование значений параметров и заполнение БД

for(int i = 0; i<N;i++)

{

query\_str\_param = "INSERT INTO param (date,x,y,z,epr,number,expid) VALUES ('"+

datetime.toString("yyyy-MM-dd HH:mm:ss")+"',"+QString::number(x,'f',2)+","+QString::number(y,'f',2)+","+

QString::number(z,'f',2)+","+QString::number(epr,'f',2)+",1,"+id\_exp+")";

query.prepare(query\_str\_param);

query.exec();

query.clear();

qDebug()<<query\_str\_param;

p.prop(t,x,y,z,Vx,Vy,Vz,dt,angle,epr,type,subtype);

datetime = datetime\_const.addSecs((qint64)(i+1)\*dt);

ui->progressBar->setValue(i+1);

}

ui->progressBar->hide();

} else if(ui->radioButton->isChecked())

{ //Если выбрано сохранение параметров в файл

//Создание файла и выбор каталога

QFile f(QFileDialog::getSaveFileName(this, "Сохранить параметры эксперимента", "", "\*.txt"));

//Открытие файла и запись параметров

if( f.open( QIODevice::WriteOnly ) )

{

QTextStream ts( &f );

//QStringList strList;

QString str;

str = "date x y z epr number\n";

ts << str;

QDateTime datetime\_const = ui->dateTimeEdit->dateTime(),

datetime;

if(type==1)

{

epr = p.calcEPR\_TYPE1(subtype);

} else

{

epr = p.calcEPR\_TYPE2(angle,subtype);

}

str = datetime\_const.toString("yyyy-MM-dd HH:mm:ss")+" "+

QString::number(x,'f',2).replace(',','.')+" "+

QString::number(y,'f',2).replace(',','.')+" "+

QString::number(z,'f',2).replace(',','.')+" "+

QString::number(epr,'f',2).replace(',','.')+" "+

QString::number(1)+"\n";

ts << str;

ui->progressBar->show();

for(int i = 0; i<N;i++)

{

p.prop(t,x,y,z,Vx,Vy,Vz,dt,angle,epr,type,subtype);

datetime = datetime\_const.addSecs((qint64)(i+1)\*dt);

ui->progressBar->setValue(i+1);

str = datetime.toString("yyyy-MM-dd HH:mm:ss")+" "+

QString::number(x,'f',2).replace(',','.')+" "+

QString::number(y,'f',2).replace(',','.')+" "+

QString::number(z,'f',2).replace(',','.')+" "+

QString::number(epr,'f',2).replace(',','.')+" "+

QString::number(1)+"\n";

ts << str;

ui->progressBar->setValue(i+1);

}

ui->progressBar->hide();

}

f.close();

}

//Обновление таблицы 1

update\_table1();

}

**/\*Обработка изменения индекса QComboBox\_2 (тип)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Входные данные: строка ввода даты dateTimeEdit**

**\*Результат: Изменение значений QComboBox (подтип) в зависимости от QComboBox\_2 (тип)**

**\*Версия 1\*/**

void MainWindow::on\_comboBox\_2\_activated(int index)

{

ui->comboBox->clear();

if(index == 0)

{

ui->comboBox->addItem("Подтип 1",1);

ui->comboBox->addItem("Подтип 2",2);

ui->comboBox->addItem("Подтип 3",3);

} else

{

ui->comboBox->addItem("Подтип 1",1);

ui->comboBox->addItem("Подтип 2",2);

ui->comboBox->addItem("Подтип 3",3);

ui->comboBox->addItem("Подтип 4",4);

ui->comboBox->addItem("Подтип 5",5);

ui->comboBox->addItem("Подтип 6",6);

ui->comboBox->addItem("Подтип 7",7);

ui->comboBox->addItem("Подтип 8",8);

ui->comboBox->addItem("Подтип 9",9);

ui->comboBox->addItem("Подтип 10",10);

}

}

# *Приложение Г. Листинг файла prognoz.h.*

**/\*Реализация класса моделирования параметров prognoz**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#ifndef PROGNOZ\_H

#define PROGNOZ\_H

//Подключение нелюходимых библиотек

#include <iostream>

#include <math.h>

#include <QDebug>

class prognoz

{

public:

prognoz()

{

};

private:

//Инициализация константных переменных

double Wz = 0.7292115e-4, J2 = 108262.575e-8, cz = 0.5\*J2\*pow(6378136.,2.), R = 6371000.0,

a00 = 39860044.18e+7/R, a20 = -2 \* cz \* 39860044.18e+7/pow(R,3.);

**/\*Расчет коэффициента 1 для вычисления Vx**

**\*Входные параметры: координаты x, y, z**

**\*проекции вектора скорости на оси X, Y, Z - vx, vy, vz**

**\*шаг времени dt**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double K1 (const double &x, const double &y, const double &z,

const double &Vx, const double &Vy, const double &Vz,

const double &dt/\*, const double &bk)

{

double r = sqrt(pow(x,2.0)+pow(y,2.0)+pow(z,2.0)),

b = R/pow(r,3.0),

d = z/r;

d = 5.0 \* pow(d,2.);

double c = 1.5 \* a20 \* pow(R,2.)/pow(r,2.),

a = b \* (a00 + c \* (d - 1.)),

\_Wz = Wz;

return ((pow(\_Wz,2.0)-a)\*x+2.0\*\_Wz\*Vy)\*dt;

}

**/\*Расчет коэффициента 2 для вычисления Vy**

**\*Входные параметры: координаты x, y, z**

**\*проекции вектора скорости на оси X, Y, Z - vx, vy, vz**

**\*шаг времени dt**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double K2 (const double &x, const double &y, const double &z,

const double &Vx, const double &Vy, const double &Vz,

const double &dt)

{

double r = sqrt(pow(x,2.0)+pow(y,2.0)+pow(z,2.0)),

b = R/pow(r,3.0),

d = z/r;

d = 5.0 \* pow(d,2.);

double c = 1.5 \* a20 \* pow(R,2.)/pow(r,2.),

a = b \* (a00 + c \* (d - 1.)),

\_Wz = Wz;

return((pow(\_Wz,2.0) - a)\*y - 2\*\_Wz\*Vx)dt;

}

**/\*Расчет коэффициента 3 для вычисления Vz**

**\*Входные параметры: координаты x, y, z**

**\*проекции вектора скорости на оси X, Y, Z - vx, vy, vz**

**\*шаг времени dt**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double K3 (const double &x, const double &y, const double &z,

const double &Vx, const double &Vy, const double &Vz,

const double &dt/\*, const double &bk, const double &atmorange\*/)

{

double r = sqrt(pow(x,2.0)+pow(y,2.0)+pow(z,2.0)),

b = R/pow(r,3.0),

d = z/r;

d = 5.0 \* pow(d,2.);

double c = 1.5 \* a20 \* pow(R,2.)/pow(r,2.),

a = b \* (a00 + c \* (d - 1.)),

bk = 0.0001,

\_density = getDensity(z),

rV=sqrt(pow(Vx,2.0)+pow(Vy,2.0)+pow(Vz,2.0));

//qDebug()<<"K3="<<((2.\*b\*c - a)\*z- 0.5\*bk\*\_density\*Vz\*rV)\*dt<<"\n";

return((2.\*b\*c - a)\*z/\*- 0.5\*bk\*\_density\*Vz\*rV\*/)\*dt;

}

public:

**/\*Расчет параметров на один шаг**

**\*Входные параметры: координаты x, y, z;**

**\*проекции вектора скорости на оси X, Y, Z - vx, vy, vz;**

**\*шаг времени dt;**

**\*угол локации angle;**

**\*значение ЭПР epr;**

**\*тип type;**

**\*подтип subtype.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double prop (double &t, double &x, double &y, double &z,

double &Vx, double &Vy, double &Vz,

const double &dt,const double angle,double &epr, int type,int subtype)

{

//Рассчет значения ЭПР

if(type==1)

{

epr = calcEPR\_TYPE1(subtype);

} else

{

epr = calcEPR\_TYPE2(angle,subtype);

}

//Нахождение коэффициентов K1 для метода Рунге-Кутты

double K11 = K1(x, y, z, Vx, Vy,Vz, dt),

K12 = K2(x, y, z, Vx, Vy,Vz, dt),

K13 = K3(x, y, z, Vx, Vy,Vz, dt),

K14 = Vx\*dt,

K15 = Vy\*dt,

K16 = Vz\*dt;

//Нахождение поправок для коэффициентов К2

double kx = 0.5\*K14,

ky = 0.5\*K15,

kz = 0.5\*K16,

kvx = 0.5\*K11,

kvy = 0.5\*K12,

kvz = 0.5\*K13;

//Нахождение коэффициентов K2 для метода Рунге-Кутты

double K21 = K1(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K22 = K2(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K23 = K3(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy, Vz+kvz,dt),

K24 = (Vx + kvx)\*dt,

K25 = (Vy + kvy)\*dt,

K26 = (Vz + kvz)\*dt;

//Нахождение поправок для коэффициентов К3

kx = 0.5\*K24;

ky = 0.5\*K25;

kz = 0.5\*K26;

kvx = 0.5\*K21;

kvy = 0.5\*K22;

kvz = 0.5\*K23;

//Нахождение коэффициентов K3 для метода Рунге-Кутты

double K31 = K1(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K32 = K2(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K33 = K3(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K34 = (Vx + kvx)\*dt,

K35 = (Vy + kvy)\*dt,

K36 = (Vz + kvz)\*dt;

//Нахождение поправок для коэффициентов К4

kx = K34;

ky = K35;

kz = K36;

kvx = K31;

kvy = K32;

kvz = K33;

//Нахождение коэффициентов K4 для метода Рунге-Кутты

double K41 = K1(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K42 = K2(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K43 = K3(x + kx, y + ky, z + kz, Vx + kvx, Vy + kvy,Vz+kvz, dt),

K44 = (Vx + kvx)\*dt,

K45 = (Vy + kvy)\*dt,

K46 = (Vz + kvz)\*dt;

//Нахождение следующих значений параметров

t += dt;

x += (K14 + 2.\*K24 + 2.\*K34 + K44)/6.;

y += (K15 + 2.\*K25 + 2.\*K35 + K45)/6.;

z += (K16 + 2.\*K26 + 2.\*K36 + K46)/6.;

Vx += (K11 + 2.\*K21 + 2.\*K31 + K41)/6.;

Vy += (K12 + 2.\*K22 + 2.\*K32 + K42)/6.;

Vz += (K13 + 2.\*K23 + 2.\*K33 + K43)/6.;

return 0;

}

private:

//Массив плотности на определенной высоте

double density[2][35] =

{

//высота

{

0,500,1000,1500,2000,2500,3000,4000,5000,6000,7000,8000,9000,10000,

11000,12000,14000,16000,18000,20000,24000,28000,32000,36000,40000,50000,

60000,80000,100000,150000,200000,300000,500000,700000,1000000

},

//Плотность

{

1.2250, 1.1673,1.1117,1.0581,1.0065,0.9569,0.9093,0.8194,0.7365,0.6601,0.59,

0.5258,0.4671,0.4671,0.3648,0.3119,0.2279,0.1665,0.1216,0.0889,0.0469,0.0251,

0.0136,7.26e-3,4.00e-3,1.03e-3,3.0e-4,1.85e-5,5.55e-7,2.0e-9,2.52e-10,1.92e-11,

5.21e-13,3.07e-14,3.56e-15

}

};

**/\*Расчет плотности**

**\*Входные параметры: высота z**

**\*массив плотности density**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double getDensity(const double &z)

{

for(int i = 1;i<35;i++)

{

if(z>density[0][i-1]&&z<density[0][i])

{

return (density[1][i-1]+density[1][i])/2.;

}

}

return 0;

}

//Среднее значение ЭПР для типа 1

double SRZNACH\_TYPE1[3] =

{

0.42,0.43,0.2

};

//Минимальное и максимальное значение ЭПР для типа 2

//Зависит от угла локации

double MIN\_MAX\_TYPE2[2][2][10] =

{

//для угла 0-15

{

//минимальное

{

0.004, 0.0013, 0.0251, 0.0026, 0.021, 0.001, 0.001, 0.006, 0.005,0.024

},

//максимальное

{

0.12,0.201,0.35,0.22,0.34,0.3,0.4,0.2,0.19,0.7

}

},

//Для угла 15-30

{

//минимальное

{

0.004,0.003,0.01,0.032,0.01,0.001,0.001,0.0032,0.05,0.04

},

//максимальное

{

0.5,0.16,0.78,0.516,0.526,0.76,0.3,0.616,0.51,0.7

}

}

};

**/\*Метод получения случайного значения**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double my\_uniform()

{

int r = rand();

return (double)r/RAND\_MAX;

}

**/\*Реализация равномерной функции распределения**

**\*Входные данные: минимальное и максимальное значение - min, max**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double my\_uniformMINtoMAX(double min, double max)

{

return my\_uniform()\*(max-min)+min;

}

**/\*Реализация экспоненциальной функции распределения**

**\*Входные данные: лямбда lambda**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double my\_exponential(double lambda)

{

return -log(my\_uniform())/lambda;

}

public:

**/\*Расчет значения ЭПР на один шаг для типа 1**

**\*Входные данные: подтип subtype**

**\*Результат: случайное значение ЭПР**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double calcEPR\_TYPE1(int subtype)

{

double mo = SRZNACH\_TYPE1[subtype-1];

double lambda = 1/mo;

return my\_exponential(lambda);

}

**/\*Расчет значения ЭПР на один шаг для типа 2**

**\*Входные данные: угол локации angle;**

**\*подтип subtype.**

**\*Результат: случайное значение ЭПР**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

double calcEPR\_TYPE2(double angle,int subtype)

{

int \_angle = static\_cast<int>((double)angle/15.);

double min = MIN\_MAX\_TYPE2[\_angle][0][subtype];

double max = MIN\_MAX\_TYPE2[\_angle][1][subtype];

return my\_uniformMINtoMAX(min,max);

}

};

#endif // PROGNOZ\_H

# *Приложение Д. Листинг заголовочного файла mainwindow2.h.*

**/\*Описание полей и методов класса MainWindow2**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

**/\*Описание импорта необходимых библиотек**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#include <QMainWindow>

#include <QtSql>

#include <QDebug>

#include <QFileInfo>

#include <QSqlTableModel>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QMessageBox>

#include <QStandardItem>

#include <QGridLayout>

#include "mygraphicview.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

**/\*Описание слотов обработки событий: нажатия на кнопки,**

**\*нажатие на QTableView, изменение индекса в QTableView,**

**\*нажатие на QComboBox;**

**\*изменение состояния QCheckBox;**

**\*нажатие на календарь;**

**\*Разработал студент А-08-17 Полушин А.**

**\*Дата написания 01.05.2021**

**\*Версия 1\*/**

public:

MainWindow(QWidget \*parent = NULL);

~MainWindow();

private slots:

void on\_calendarWidget\_clicked(const QDate &date);

void on\_tableView\_clicked(const QModelIndex &index);

void on\_tableView\_activated(const QModelIndex &index);

void on\_pushButton\_clicked();

void on\_checkBox\_8\_stateChanged(int arg1);

void on\_checkBox\_7\_stateChanged(int arg1);

void on\_checkBox\_9\_stateChanged(int arg1);

void on\_checkBox\_4\_stateChanged(int arg1);

void on\_checkBox\_13\_stateChanged(int arg1);

void on\_pushButton\_2\_clicked();

void on\_checkBox\_10\_stateChanged(int arg1);

void on\_checkBox\_6\_stateChanged(int arg1);

void on\_checkBox\_5\_stateChanged(int arg1);

void on\_checkBox\_11\_stateChanged(int arg1);

void on\_pushButton\_3\_clicked();

void on\_tableView\_2\_clicked(const QModelIndex &index);

void on\_checkBox\_12\_stateChanged(int arg1);

void on\_checkBox\_16\_stateChanged(int arg1);

void on\_checkBox\_14\_stateChanged(int arg1);

void on\_checkBox\_15\_stateChanged(int arg1);

void on\_pushButton\_4\_clicked();

void on\_pushButton\_5\_clicked();

void on\_pushButton\_6\_clicked();

void on\_pushButton\_8\_clicked();

void on\_pushButton\_7\_clicked();

void on\_tableView\_4\_clicked(const QModelIndex &index);

void on\_pushButton\_11\_clicked();

void on\_pushButton\_9\_clicked();

void on\_pushButton\_10\_clicked();

void on\_checkBox\_stateChanged(int arg1);

void on\_pushButton\_12\_clicked();

void on\_pushButton\_13\_clicked();

void on\_pushButton\_14\_clicked();

void on\_tableView\_5\_clicked(const QModelIndex &index);

void on\_pushButton\_15\_clicked();

void on\_pushButton\_16\_clicked();

//Объявление полей и методов класса MainWindow2 с модификатором доступа private

private:

Ui::MainWindow \*ui;

QSqlDatabase db;//Соединение с базой данных

//Указатели на модели таблиц

QSqlTableModel \*model,\*model2;

//Id выбранного эксперимента

QString exp\_id;

//Методы обновления данных в таблицах

void update\_table1();

void update\_table2();

bool check[2][5];

//Массив выбранных параметров для построения графиков

bool check\_graph[4];

void proc();

//Метод обработки результатов модельных экспериментов

void proc2();

//Модели таблиц

QStandardItemModel \*model3,\*model4,\*model5;

//Динамические массивы для построения графиков

QVector<QVector<QDateTime>\*> vvdate;

QVector<QVector<QString>\*> vvparam;

QVector<QVector<double>\*> vvpar1,vvpar2;

//Динамический массив указателей на экзнмпляры класса MyGraphicView

QVector<MyGraphicView \*> graph;

//Текущая строка и столбец в Layout - вывод графиков

int graph\_row,graph\_column;

//Виджеты для вывода графиков

QWidget\* scrollAreaContent;

QScrollArea\* scrollArea;

QGridLayout\* grid\_layout;

//Выбранные строки на моделях

int row\_tmp,row\_tmp2;

//ID выбранного эксперимента для удаления

int id\_exp\_del;

//Множества номеров строк

QSet<int> sind2,sind;

};

#endif // MAINWINDOW\_H

# *Приложение Е. Листинг файла mainwindow2.cpp.*

**/\*Реализация класса моделирования параметров prognoz**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

//Подключение необходимых библиотек

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QSqlError>

#include <QSqlRecord>

#include <QFileDialog>

**/\*Реализация конструктора**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

//Соединение с БД

db = QSqlDatabase::addDatabase("QSQLITE");

db.setDatabaseName("..//..//data//exp.db");

//Инициализация полей класса

exp\_id="0";

graph\_row=0;

graph\_column=0;

row\_tmp2=-1;

row\_tmp=-1;

//Открытие соединения с БД

if (db.open())

ui->label->setText("");

model3 = new QStandardItemModel(0,3,this);

model4 = new QStandardItemModel(0,3,this);

model5 = new QStandardItemModel(0,3,this);

model5->clear();

ui->tableView\_5->setModel(model5);

ui->tableView\_5->resizeColumnsToContents();

ui->radioButton->setChecked(true);

ui->dateTimeEdit->setDateTime(QDateTime::currentDateTime());

ui->dateTimeEdit\_2->setDateTime(QDateTime::currentDateTime());

//Вызов методов обновления таблиц

update\_table1();

update\_table2();

//Инициализация полей класса

for (int i=0;i<2 ;i++)

{

for (int j=0;j<5 ;j++ )

{

check[i][j]=false;

}

}

for (int i=0;i<5;i++)

{

check\_graph[i]=false;

}

grid\_layout = new QGridLayout();

QWidget\* scrollAreaContent = new QWidget;

scrollAreaContent->setLayout( grid\_layout );

scrollArea = new QScrollArea;

scrollArea->setHorizontalScrollBarPolicy( Qt::ScrollBarAlwaysOff );

scrollArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);

scrollArea->setWidgetResizable(true);

scrollArea->setMinimumSize(0,0);

scrollArea->setSizePolicy(QSizePolicy::Preferred , QSizePolicy::Preferred );

scrollArea->setWidget( scrollAreaContent );

ui->gridLayout\_4->addWidget(scrollArea);

}

//Деструктор

MainWindow::~MainWindow()

{

db.close();

delete ui;

}

**/\*Обновление 1-ой таблицы**

**\*Входные данные: модель таблицы model**

**\*экземпляр класса QTableView - tableview**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::update\_table1()

{

model = new QSqlTableModel(this,db);

model->setEditStrategy(QSqlTableModel::OnManualSubmit);

model->setTable("experiment");

if(!model->select())

{

qDebug()<<model->lastError();

QMessageBox::critical(this, "Ошибка","Проверьте соединение с базой данных:\n"+model->lastError().text());

}

ui->tableView->setModel(model);

ui->tableView->resizeColumnsToContents();

ui->tableView->resizeRowsToContents();

}

**/\*Обновление 2-ой таблицы**

**\*Входные данные: модель таблицы model2**

**\*экземпляр класса QTableView - tableview**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::update\_table2()

{

model2 = new QSqlTableModel(this,db);

model2->setEditStrategy(QSqlTableModel::OnManualSubmit);

model2->setTable("param");

model2->setFilter("expid="+exp\_id);

if(!model2->select())

{

qDebug()<<model2->lastError();

QMessageBox::critical(this, "Ошибка","Проверьте соединение с базой данных:\n"+model2->lastError().text());

}

ui->tableView\_2->setModel(model2);

ui->tableView\_2->resizeColumnsToContents();

ui->tableView\_2->resizeRowsToContents();

}

**/\*Обработка нажатия на виджет календаря**

**\*Входные данные: кнопка radioButton;**

**\*строки ввода дат dateTimeEdit и dateTimeEdit2**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_calendarWidget\_clicked(const QDate &date)

{

if(ui->radioButton->isChecked())

{

ui->dateTimeEdit->setDate(date);

} else if(ui->radioButton\_2->isChecked())

{

ui->dateTimeEdit\_2->setDate(date);

}

}

**/\*Обработка нажатия на QTableView - 1-ю таблицу**

**\*Входные данные: ID эксперимента exp\_id.**

**\*Результат: обновление второй таблицы.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_clicked(const QModelIndex &index)

{

int row = index.row();

int id = index.sibling(row, 0).data().toInt();

exp\_id=QString::number(id);

update\_table2();

}

**/\*Обработка активации (например, с помощью кнопок клавиатуры) первой таблицы**

**\*Разработал студент А-08-17 Полушин А.**

**\*Результат: вызов метода on\_tableView\_clicked(index)**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_activated(const QModelIndex &index)

{

on\_tableView\_clicked(index);

}

**/\*Обработка нажатия на кнопку "применить"**

**\*Входные данные: модель 1-ой таблицы model;**

**\*строка ввода названия эксперимента lineEdit**

**\*Результат: обновление 1-ой таблицы (установка фильтра)**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_clicked()

{

if(ui->lineEdit->text()=="")

{

model->setFilter(QString("date between '%1' and '%2' ")

.arg(ui->dateTimeEdit->date().toString("yyyy-MM-dd hh:mm:ss"),

ui->dateTimeEdit\_2->date().toString("yyyy-MM-dd hh:mm:ss")));

} else

{

model->setFilter(QString("date between '%1' and '%2' and name like '%3'")

.arg(ui->dateTimeEdit->date().toString("yyyy-MM-dd hh:mm:ss"),

ui->dateTimeEdit\_2->date().toString("yyyy-MM-dd hh:mm:ss"),

ui->lineEdit->text()));

}

model->select();

ui->tableView->setModel(model);

ui->tableView->resizeColumnsToContents();

ui->tableView->resizeRowsToContents();

}

**/\*Обработка нажатия на кнопку "Обработать"**

**\*Входные данные: модель таблицы выбранных экспериментов;**

**\*состояние "плавающей дельты" QCheckBox - checkBox;**

**\*значение "плавающей дельты" - doubleSpinBox;**

**\*Кнопки выбора параметров усреднения (координаты X, Y, Z и ЭПР);**

**\*Кнопки выбора параметров привязки (координаты X, Y, Z и ЭПР);**

**\*Результат: проверка введенных значений и вывод обработки результатов**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_2\_clicked()

{

if(model5->rowCount()<2)

{

QMessageBox::warning(this, "Ошибка","Выберите минимум 2 эксперимента");

return;

}

if(!ui->checkBox->isChecked()&&ui->doubleSpinBox->value()==0)

{

QMessageBox::warning(this, "Ошибка","Введите корректную дельту");

return;

}

ui->tableView\_3->setModel(model3);

//ui->tableView\_3->resizeColumnsToContents();

proc2();

if(ui->radioButton\_7->isChecked())

{

model3->setHeaderData(0,Qt::Horizontal,"Координата X");

} else if(ui->radioButton\_8->isChecked())

{

model3->setHeaderData(0,Qt::Horizontal,"Координата Y");

}else if(ui->radioButton\_9->isChecked())

{

model3->setHeaderData(0,Qt::Horizontal,"Высота");

}else if(ui->radioButton\_10->isChecked())

{

model3->setHeaderData(0,Qt::Horizontal,"ЭПР");

}

if(ui->radioButton\_11->isChecked())

{

model3->setHeaderData(1,Qt::Horizontal,"Координата X");

} else if(ui->radioButton\_12->isChecked())

{

model3->setHeaderData(1,Qt::Horizontal,"Координата Y");

}else if(ui->radioButton\_13->isChecked())

{

model3->setHeaderData(1,Qt::Horizontal,"Высота");

}else if(ui->radioButton\_14->isChecked())

{

model3->setHeaderData(1,Qt::Horizontal,"ЭПР");

}

model3->setHeaderData(2,Qt::Horizontal,"Количество\nэкспериментов");

ui->tableView\_3->resizeColumnsToContents();

}

**/\*Метод обработки результатов модельных экспериментов**

**\*Входные данные: модель таблицы выбранных экспериментов;**

**\*состояние "плавающей дельты" QCheckBox - checkBox;**

**\*значение "плавающей дельты" - doubleSpinBox;**

**\*Кнопки выбора параметров усреднения (координаты X, Y, Z и ЭПР);**

**\*Кнопки выбора параметров привязки (координаты X, Y, Z и ЭПР);**

**\*Результат: усреднение заданного параметра по всем выбранным экспериментам**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::proc2()

{

//Установка названий столбцов привязки (bind) и усреднения (param)

QString bind,param;

if(ui->radioButton\_7->isChecked())

{

bind="x";

} else if(ui->radioButton\_8->isChecked())

{

bind="y";

}else if(ui->radioButton\_9->isChecked())

{

bind="z";

}else if(ui->radioButton\_10->isChecked())

{

bind="epr";

}

if(ui->radioButton\_11->isChecked())

{

param="x";

} else if(ui->radioButton\_12->isChecked())

{

param="y";

}else if(ui->radioButton\_13->isChecked())

{

param="z";

}else if(ui->radioButton\_14->isChecked())

{

param="epr";

}

//Создание запросов

QSqlQuery query1,query2;

double delta=0;

bool fl\_delta=false;

//Если выбрана "плавающая дельта"

if(ui->checkBox->isChecked())

fl\_delta=true;

else delta = ui->doubleSpinBox->value();

//Подготовка запроса выборки

query1.prepare("SELECT "+bind+","+param+" FROM param WHERE expid="+model5->data(model5->index(0,0)).toString());

//Выполнение запроса выборки

if(!query1.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

//Вызов слота обработки нажатия на кнопку "Очистить"

on\_pushButton\_5\_clicked();

//Переменная для подсчета числа строк

int inc=0;

//Индекс для перемещения по таблице

QModelIndex indm3;

//Перебор результатов выборки и установление их в таблицу

while (query1.next())

{

model3->insertRow(model3->rowCount());

indm3=model3->index(inc,0);

model3->setData(indm3,query1.value(0));

indm3=model3->index(inc,1);

model3->setData(indm3,query1.value(1));

inc++;

}

//Перейти в начало выборки

query1.first();

//Создание массива количества попадатий в заданный диапазон

int n[model3->rowCount()];

//Инициализация массива n

for (int i =0;i<model3->rowCount();i++)

{

n[i]=1;

}

//Перебор строк таблицы выбранных экспериментов

for(int j = 0;j<model5->rowCount()-1;j++)

{

//Подготовка запроса выборки не основного эксперимента

query2.prepare("SELECT "+bind+","+param+",paramid FROM param WHERE expid="+model5->data(model5->index(j+1,0)).toString());

//Промежуточные переменные для нахождения параметров, которые попадают в диапазон

double p1=0;

double p2=0;

//Промежуточные результаты

double result1=0;

double result2=0;

//Множество ID параметров, которые уже попали в диапазон

QSet<int> set\_id;

//Выполнение запроса

if(!query2.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

//Перебор строк таблицы результатов обработки

for(int i=0;i<model3->rowCount();i++)

{

indm3=model3->index(i,0);

p1=model3->data(indm3).toDouble();

result1=p1;

indm3=model3->index(i,1);

result2=model3->data(indm3).toDouble();

while(query2.next())

{

p2=query2.value(0).toDouble();

//Если дельта задана

if(!fl\_delta)

{

//Если значение попадает в диапазон

if(p2>=p1-delta&&p2<=p1+delta)

{

//Если ID параметра не содержится во множестве

if(!set\_id.contains(query2.value(2).toInt()))

{

//Добавить ID параметра во множество

set\_id.insert(query2.value(2).toInt());

n[i]++; //Увеличить число попаданий в диапазон

//Увеличить результаты

result1+=query2.value(0).toDouble();

result2+=query2.value(1).toDouble();

}

}

} else //Плавающая дельта

{

double tmp1;

indm3 = model3->index(i,0);

tmp1 = model3->data(indm3).toDouble();

if(i+1<model3->rowCount())

{

//Если следующего значения нет - delta = средние между текущим и предыдущим

indm3 = model3->index(i+1,0);

delta = (tmp1 - model3->data(indm3).toDouble())/2;

if(delta<0)

delta \*=-1;

}

else

{

//Если следующее значение есть - delta = средние между текущим и следующим

indm3 = model3->index(i-1,0);

delta = (tmp1 - model3->data(indm3).toDouble())/2;

if(delta<0)

delta \*=-1;

}

//Если значение попадает в диапазон

if(p2>=p1-delta&&p2<=p1+delta)

{

//Если ID параметра не содержится во множестве

if(!set\_id.contains(query2.value(2).toInt()))

{

//Добавить ID параметра во множество

set\_id.insert(query2.value(2).toInt());

n[i]++;//Увеличить число попаданий в диапазон

//Увеличить результаты

result1+=query2.value(0).toDouble();

result2+=query2.value(1).toDouble();

}

}

}

}

indm3 = model3->index(i,0);

model3->setData(indm3,result1);

indm3 = model3->index(i,1);

model3->setData(indm3,result2);

query2.first();

}

}

double tmp\_res;

//Проход по всем строкам таблицы результатов и усреднение параметров

for(int i = 0; i<model3->rowCount();i++)

{

indm3 = model3->index(i,0);

tmp\_res = model3->data(indm3).toDouble();

model3->setData(indm3,(double)tmp\_res/n[i]);

indm3 = model3->index(i,1);

tmp\_res = model3->data(indm3).toDouble();

model3->setData(indm3,tmp\_res/n[i]);

indm3 = model3->index(i,2);

model3->setData(indm3,n[i]);

}

}

**/\*Обработка нажатия на кнопку "Построить график"**

**\*Входные данные: модель таблицы выбранных экспериментов;**

**\*ID выбранного эксперимента;**

**\*массив выбранных параметров check\_graph[];**

**\*строка номера объекта lineEdit\_2;**

**\*динамические массивы даты и времени и параметров vvdate, vvparam;**

**\*динамический массив указателей на экзнмпляры класса MyGraphicView - graph;**

**\*Позиция графика в QGridLayout строка - graph\_row, колонка - graph\_column.**

**\*Результат: построение и добавление графика в QGridLayout.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_3\_clicked()

{

//Если не выбран эксперимент

if(exp\_id=="" || exp\_id=="0")

{

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

return;

}

//Флаг состояния выбора параметров

bool flag=false;

for(int i=0;i<4;i++)

{

if(check\_graph[i])

{

flag=true;

}

}

//Если не выбрано хотя бы одного параметра - сообщение об ошибке

if(!flag)

{

QMessageBox::warning(this, "Ошибка","Выберите параметры");

return;

}

//Номер обЪекта

int num=0;

//Корректность ввода номера объекта

bool check\_text;

//Если номер объекта задан

if(ui->lineEdit\_2->text()!="")

{

//Проверка корректности номера объекта

num=ui->lineEdit\_2->text().toInt(&check\_text,10);

if (!check\_text)

{

QMessageBox::warning(this, "Ошибка","Введите корректный номер объекта");

return;

}

//Если выбрана координата x

if(check\_graph[0])

{

//Создание запроса

QSqlQuery queryToSqlite = QSqlQuery(db);

//Выполнение запроса выборки

if(!queryToSqlite.exec("select date,x from param where expid = "+exp\_id+" and number="+QString::number(num)+" order by date"))

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с базой данных");

qDebug() << queryToSqlite.lastError().databaseText();

qDebug() << queryToSqlite.lastError().driverText();

} else

{

//Добавление новых значений в массивы

graph.append(new MyGraphicView(this));

vvdate.append(new QVector<QDateTime>);

vvparam.append(new QVector<QString>);

while(queryToSqlite.next())

{

vvdate.last()->append(queryToSqlite.value(0).toDateTime());

vvparam.last()->append(queryToSqlite.value(1).toString());

}

if(vvdate.last()->isEmpty())

{

QMessageBox::warning(this, "Ошибка","Отсутствуют данные при заданных параметрах\nНомер эксперимента: "+exp\_id+"\nНомер объекта: "+QString::number(num));

return;

}

//Вызов метода setData()

graph.last()->setData(vvdate.last(),vvparam.last(),"X");

//Добавление графика в QGridLayout

grid\_layout->addWidget(graph.last(),graph\_row,graph\_column);

//инкремент строк и(или) столбцов

if(graph\_column>=1)

{

graph\_column=0;

graph\_row++;

}

else

{

graph\_column++;

}

}

}

//Если выбрана координата Y

if(check\_graph[1])

{

//Создание запроса

QSqlQuery queryToSqlite = QSqlQuery(db);

//Выполнение запроса выборки

if(!queryToSqlite.exec("select date,y from param where expid = "+exp\_id+" and number="+QString::number(num)+" order by date"))

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с базой данных");

qDebug() << queryToSqlite.lastError().databaseText();

qDebug() << queryToSqlite.lastError().driverText();

} else

{

//Добавление новых значений в массивы

graph.append(new MyGraphicView(this));

vvdate.append(new QVector<QDateTime>);

vvparam.append(new QVector<QString>);

while(queryToSqlite.next())

{

vvdate.last()->append(queryToSqlite.value(0).toDateTime());

vvparam.last()->append(queryToSqlite.value(1).toString());

}

if(vvdate.last()->isEmpty())

{

QMessageBox::warning(this, "Ошибка","Отсутствуют данные при заданных параметрах\nНомер эксперимента: "+exp\_id+"\nНомер объекта: "+QString::number(num));

return;

}

//Вызов метода setData()

graph.last()->setData(vvdate.last(),vvparam.last(),"Y");

//Добавление графика в QGridLayout

grid\_layout->addWidget(graph.last(),graph\_row,graph\_column);

//инкремент строк и(или) столбцов

if(graph\_column>=1)

{

graph\_column=0;

graph\_row++;

}

else

{

graph\_column++;

}

}

}

//Если выбрана координата Z

if(check\_graph[2])

{

//Создание запроса

QSqlQuery queryToSqlite = QSqlQuery(db);

//Выполнение запроса выборки

if(!queryToSqlite.exec("select date,z from param where expid = "+exp\_id+" and number="+QString::number(num)+" order by date"))

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с базой данных");

qDebug() << queryToSqlite.lastError().databaseText();

qDebug() << queryToSqlite.lastError().driverText();

} else

{

//Добавление новых значений в массивы

graph.append(new MyGraphicView(this));

vvdate.append(new QVector<QDateTime>);

vvparam.append(new QVector<QString>);

while(queryToSqlite.next())

{

vvdate.last()->append(queryToSqlite.value(0).toDateTime());

vvparam.last()->append(queryToSqlite.value(1).toString());

}

if(vvdate.last()->isEmpty())

{

QMessageBox::warning(this, "Ошибка","Отсутствуют данные при заданных параметрах\nНомер эксперимента: "+exp\_id+"\nНомер объекта: "+QString::number(num));

return;

}

//Вызов метода setData()

graph.last()->setData(vvdate.last(),vvparam.last(),"Высота");

//Добавление графика в QGridLayout

grid\_layout->addWidget(graph.last(),graph\_row,graph\_column);

//инкремент строк и(или) столбцов

if(graph\_column>=1)

{

graph\_column=0;

graph\_row++;

}

else

{

graph\_column++;

}

}

}

//Если выбрана ЭПР

if(check\_graph[3])

{

//Создание запроса

QSqlQuery queryToSqlite = QSqlQuery(db);

//Выполнение запроса выборки

if(!queryToSqlite.exec("select date,epr from param where expid = "+exp\_id+" and number="+QString::number(num)+" order by date"))

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с базой данных");

qDebug() << queryToSqlite.lastError().databaseText();

qDebug() << queryToSqlite.lastError().driverText();

} else

{

//Добавление новых значений в массивы

graph.append(new MyGraphicView(this));

vvdate.append(new QVector<QDateTime>);

vvparam.append(new QVector<QString>);

while(queryToSqlite.next())

{

vvdate.last()->append(queryToSqlite.value(0).toDateTime());

vvparam.last()->append(queryToSqlite.value(1).toString());

}

if(vvdate.last()->isEmpty())

{

QMessageBox::warning(this, "Ошибка","Отсутствуют данные при заданных параметрах\nНомер эксперимента: "+exp\_id+"\nНомер объекта: "+QString::number(num));

return;

}

//Вызов метода setData()

graph.last()->setData(vvdate.last(),vvparam.last(),"ЭПР");

//Добавление графика в QGridLayout

grid\_layout->addWidget(graph.last(),graph\_row,graph\_column);

//инкремент строк и(или) столбцов

if(graph\_column>=1)

{

graph\_column=0;

graph\_row++;

}

else

{

graph\_column++;

}

}

}

} else

{

//Если номер объекта некорректный

QMessageBox::warning(this, "Ошибка","Введите номер объекта");

return;

}

}

**/\*Обработка нажатия на QTableView\_2 - 2-ю таблицу**

**\*Входные данные: строка ввода номера объекта**

**\*Результат: установление номера объекта в строку**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_2\_clicked(const QModelIndex &index)

{

int row = index.row();

int num = index.sibling(row, 6).data().toInt();

ui->lineEdit\_2->setText(QString::number(num));

}

**/\*Обработка изменения состояния QCheckBox - координата X**

**\*Входные данные: аргумент состояния arg1;**

**\*Массив выбранных параметров для построения графиков check\_graph[].**

**\*Результат: изменение массива check\_graph[].**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_checkBox\_12\_stateChanged(int arg1)

{

if (arg1==2)

{

check\_graph[0]=true;

}

else

{

check\_graph[0]=false;

}

}

**/\*Обработка изменения состояния QCheckBox - координата Y**

**\*Входные данные: аргумент состояния arg1;**

**\*Массив выбранных параметров для построения графиков check\_graph[].**

**\*Результат: изменение массива check\_graph[].**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_checkBox\_16\_stateChanged(int arg1)

{

if (arg1==2)

{

check\_graph[1]=true;

}

else

{

check\_graph[1]=false;

}

}

**/\*Обработка изменения состояния QCheckBox - высота**

**\*Входные данные: аргумент состояния arg1;**

**\*Массив выбранных параметров для построения графиков check\_graph[].**

**\*Результат: изменение массива check\_graph[].**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_checkBox\_14\_stateChanged(int arg1)

{

if (arg1==2)

{

check\_graph[2]=true;

}

else

{

check\_graph[2]=false;

}

}

**/\*Обработка изменения состояния QCheckBox - ЭПР**

**\*Входные данные: аргумент состояния arg1;**

**\*Массив выбранных параметров для построения графиков check\_graph[].**

**\*Результат: изменение массива check\_graph[].**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_checkBox\_15\_stateChanged(int arg1)

{

if (arg1==2)

{

check\_graph[3]=true;

}

else

{

check\_graph[3]=false;

}

}

**/\*Обработка нажатия на кнопку "Очистить" во вкладке "График"**

**\*Входные данные: аргумент состояния arg1;**

**\*Массив выбранных параметров для построения графиков check\_graph[].**

**\*Результат: очистка массивов и удаление графиков.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_4\_clicked()

{

for (int i=0;i<graph.count();i++)

delete graph[i];

for(int i=0;i<vvdate.count();i++)

delete vvdate[i];

for(int i=0;i<vvparam.count();i++)

delete vvparam[i];

for(int i=0;i<vvpar1.count();i++)

delete vvpar1[i];

for(int i=0;i<vvpar2.count();i++)

delete vvpar2[i];

graph\_row=0;

graph\_column=0;

graph.clear();

vvdate.clear();

vvparam.clear();

vvpar1.clear();

vvpar2.clear();

}

**/\*Обработка нажатия на кнопку "Очистить" во вкладке "обработка"**

**\*Входные данные: модель таблицы обработки результатов model3**

**\*Результат: очистка таблицы**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_5\_clicked()

{

for(int i=0;i<model3->rowCount();i++)

{

for(int j=0;j<model3->columnCount();j++)

{

model3->clear();

}

}

model3 = new QStandardItemModel(0,3,this);

ui->tableView\_3->setModel(model3);

}

**/\*Обработка нажатия на кнопку "Сохранить в файл"**

**\*Входные данные: QTableView\_3 - таблица результатов обработки**

**\*Результат: сохранение таблицы в файл формата CSV**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_6\_clicked()

{

QFile f(QFileDialog::getSaveFileName(this, "Сохранить результаты обработки", "", "\*.csv"));

//QString filename = QFileDialog::getSaveFileName(this, tr("Save"), "", tr("CSV (\*.csv)"));

if( f.open( QIODevice::WriteOnly ) )

{

QTextStream ts( &f );

QStringList strList;

strList << "\" \"";

for( int c = 0; c < ui->tableView\_3->horizontalHeader()->count(); ++c )

strList << "\""+ui->tableView\_3->model()->headerData(c, Qt::Horizontal).toString()+"\"";

ts << strList.join( ";" )+"\n";

for( int r = 0; r < ui->tableView\_3->verticalHeader()->count(); ++r )

{

strList.clear();

strList << "\""+ui->tableView\_3->model()->headerData(r, Qt::Vertical).toString()+"\"";

for( int c = 0; c < ui->tableView\_3->horizontalHeader()->count(); ++c )

{

//QString tmp\_str = QString::number(ui->tableView\_3->model()->data(ui->tableView\_3->model()->index(r, c), Qt::DisplayRole),'f',3).replace(".",",");

QString tmp\_str = QString::number(model3->data(model3->index(r,c),Qt::DisplayRole).toDouble(),'f',3).replace(".",",");

strList << "\""+tmp\_str+"\"";

}

ts << strList.join( ";" )+"\n";

}

f.close();

}

}

**/\*Обработка нажатия на кнопку "Добавить" во вкладке "доп. Обработка"**

**\*Входные данные: QTableView\_4 - таблица выбранных экспериментов;**

**\*ID эксперимента - exp\_id;**

**\*модель таблицы - model4;**

**\*Множество id параметров sind2.**

**\*Результат: добавление выбранного эксперимента в таблицу QTableView\_4**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_8\_clicked()

{

//Если не выбран эксперимент

if(exp\_id=="0")

{

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

return;

}

//Создание запроса выборки

QSqlQuery query("SELECT \* FROM experiment WHERE expid="+exp\_id);

//Выполнение запроса

if(!query.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

//Если ID эксперимента уже содержится во множестве

if(sind2.contains(exp\_id.toInt()))

{

QMessageBox::warning(this, "Ошибка","Эксперимент уже выбран");

return;

//Если ID эксперимента не содержится во множестве

} else sind2.insert(exp\_id.toInt());

query.next();

model4->setColumnCount(3);

model4->setHeaderData(1,Qt::Horizontal,"Название эксперимента");

model4->setHeaderData(2,Qt::Horizontal,"Дата");

model4->insertRow(model4->rowCount());

QModelIndex ind = model4->index(model4->rowCount()-1,0);

model4->setData(ind,query.value(0));

ind = model4->index(model4->rowCount()-1,1);

model4->setData(ind,query.value(1));

ind = model4->index(model4->rowCount()-1,2);

model4->setData(ind,query.value(2));

ui->tableView\_4->setModel(model4);

ui->tableView\_4->resizeColumnsToContents();

}

**/\*Обработка нажатия на кнопку "Удалить" во вкладке "доп. Обработка"**

**\*Входные данные: QTableView\_4 - таблица выбранных экспериментов;**

**\*ID эксперимента - exp\_id;**

**\*Множество id параметров sind2;**

**\*индекс выбранной строки row\_tmp2;**

**\*модель таблицы - model4.**

**\*Результат: Удаление выбранного эксперимента из таблицы QTableView\_4**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_7\_clicked()

{

//Если строка выбрана

if(row\_tmp2!=-1)

{

//Удалить строку из модели

model4->removeRow(row\_tmp2);

//Удалить индекс из множества

sind2.remove(row\_tmp2);

row\_tmp2=-1;

if(model4->rowCount()==0)

model4->clear();

}

else

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

}

**/\*Обработка нажатия на QTableView\_4 во вкладке "доп. Обработка"**

**\*Входные данные: выбранная строка row\_tmp2;**

**\*индекс выбранной строки index.**

**\*Результат: изменение row\_tmp2.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_4\_clicked(const QModelIndex &index)

{

row\_tmp2=index.row();

}

**/\*Обработка нажатия на кнопку "Удалить все" во вкладке "доп. Обработка"**

**\*Входные данные: множество id параметров sind2;**

**\*модель таблицы - model4.**

**\*Результат: очистка множества и модели**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_11\_clicked()

{

model4->clear();

sind2.clear();

}

**/\*Обработка нажатия на кнопку "Очистить" во вкладке "доп. Обработка"**

**\*Входные данные: строка вывода усреднения lineEdit\_3**

**\*Результат: очистка множества и модели**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_9\_clicked()

{

ui->lineEdit\_3->clear();

}

**/\*Обработка нажатия на кнопку "Обработка" во вкладке "доп. Обработка"**

**\*Входные данные: строка вывода усреднения lineEdit\_3;**

**\*модель таблицы model4;**

**\*кнопки выбора параметров: radioButton\_3, radioButton\_4, radioButton\_5, radioButton\_6;**

**\*Результат: очистка множества и модели**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_10\_clicked()

{

if(model4->rowCount()!=0)

{

QSqlQuery query;

QString str\_query;

//Создание запроса

str\_query="SELECT AVG(";

if(ui->radioButton\_3->isChecked())

str\_query+="x";

else

if(ui->radioButton\_4->isChecked())

str\_query+="y";

else

if(ui->radioButton\_5->isChecked())

str\_query+="z";

else

if(ui->radioButton\_6->isChecked())

str\_query+="epr";

else

{

QMessageBox::warning(this, "Ошибка","Выберите параметры");

return;

}

str\_query+=") FROM param WHERE expid=";

QModelIndex ind;

for(int i = 0; i<model4->rowCount();i++)

{

ind = model4->index(i,0);

str\_query+=model4->data(ind).toString()+" OR expid=";

}

str\_query.remove(str\_query.length()-10,10);

query.prepare(str\_query);

//Выполнение запроса

if(!query.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

query.next();

ui->lineEdit\_3->setText(query.value(0).toString());

} else

QMessageBox::warning(this, "Ошибка","Выберите эксперимент(ы)");

}

**/\*Обработка изменения состояния QCheckBox - "плавающая дельта"**

**\*Входные данные: аргумент состояния arg1;**

**\*строка ввода значения дельты doubleSpinBox.**

**\*Результат: изменение состояния doubleSpinBox.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_checkBox\_stateChanged(int arg1)

{

if (arg1==2)

{

ui->doubleSpinBox->setEnabled(false);

}

else

{

ui->doubleSpinBox->setEnabled(true);

}

}

**/\*Обработка нажатия на кнопку "Добавить" во вкладке "Обработка"**

**\*Входные данные: QTableView\_4 - таблица выбранных экспериментов;**

**\*ID эксперимента - exp\_id;**

**\*множество ID экспериментов - sind;**

**\*модель таблицы - model5.**

**\*Результат: добавление выбранного эксперимента в таблицу QTableView\_5**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_12\_clicked()

{

//Если не выбран эксперимент

if(exp\_id=="0")

{

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

return;

}

//Создание запроса выборки

QSqlQuery query("SELECT \* FROM experiment WHERE expid="+exp\_id);

//Выполнение запроса

if(!query.exec())

{

QMessageBox::warning(this, "Ошибка","Проверьте соединение с БД");

return;

}

//Если ID эксперимента уже содержится во множестве

if(sind.contains(exp\_id.toInt()))

{

QMessageBox::warning(this, "Ошибка","Эксперимент уже выбран");

return;

//Если ID эксперимента не содержится во множестве

} else sind.insert(exp\_id.toInt());

query.next();

model5->setColumnCount(3);

model5->setHeaderData(1,Qt::Horizontal,"Название эксперимента");

model5->setHeaderData(2,Qt::Horizontal,"Дата");

model5->insertRow(model5->rowCount());

QModelIndex ind = model5->index(model5->rowCount()-1,0);

model5->setData(ind,query.value(0));

ind = model5->index(model5->rowCount()-1,1);

model5->setData(ind,query.value(1));

ind = model5->index(model5->rowCount()-1,2);

model5->setData(ind,query.value(2));

ui->tableView\_5->setModel(model5);

ui->tableView\_5->resizeColumnsToContents();

}

**/\*Обработка нажатия на кнопку "Удалить" во вкладке "Обработка"**

**\*Входные данные: QTableView\_4 - таблица выбранных экспериментов;**

**\*ID эксперимента - exp\_id;**

**\*Множество ID параметров sind;**

**\*ID выбранного эксперимента id\_exp\_del;**

**\*индекс выбранной строки row\_tmp2;**

**\*модель таблицы - model5.**

**\*Результат: Удаление выбранного эксперимента из модели таблицы model5 и из множества sind.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_13\_clicked()

{

//Если выбрана строка

if(row\_tmp!=-1)

{

//Удалить из можели

model5->removeRow(row\_tmp);

//Удалить из множества

sind.remove(id\_exp\_del);

row\_tmp=-1;

//Если больше нет строк - очитсить модель

if(model5->rowCount()==0)

model5->clear();

}

else

QMessageBox::warning(this, "Ошибка","Выберите эксперимент");

}

**/\*Обработка нажатия на кнопку "Удалить все" во вкладке "Обработка"**

**\*Входные данные: множество id параметров sind;**

**\*модель таблицы - model5.**

**\*Результат: очистка множества и модели**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_14\_clicked()

{

model5->clear();

sind.clear();

}

**/\*Обработка нажатия на QTableView\_5 во вкладке "Обработка"**

**\*Входные данные: выбранная строка row\_tmp;**

**\*ID выбранного эксперимента id\_exp\_del;**

**\*индекс выбранной строки index.**

**\*Результат: изменение row\_tmp и id\_exp\_del.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_tableView\_5\_clicked(const QModelIndex &index)

{

row\_tmp=index.row();

int row = index.row();

id\_exp\_del = index.sibling(row, 0).data().toInt();

}

/\*Обработка нажатия кнопку "График по результатам обработки" во вкладке "График"

**\*Входные данные: модель таблицы обработанной информации model3;**

**\*динамические массивы параметров vvpar1, vvpar2;**

**\*динамический массив указателей на экземпляр класса mygraphicview - graph;**

**\*положение графика в QGridLayout - graph\_column, graph\_row.**

**\*Результат: изменение row\_tmp и id\_exp\_del.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_15\_clicked()

{ //Если выбрано меньше двух экспериментов

if(model3->rowCount()<2)

{

QMessageBox::warning(this, "Ошибка","Недостаточно данных");

return;

}

//Добавление новых значений в динамические массивы

graph.append(new MyGraphicView(this));

vvpar1.append(new QVector<double>);

vvpar2.append(new QVector<double>);

//Инициализация новых значений

for(int i = 0;i<model3->rowCount();i++)

{

vvpar1.last()->append(model3->data(model3->index(i,0)).toDouble());

vvpar2.last()->append(model3->data(model3->index(i,1)).toDouble());

}

//Если нет данных

if(vvpar1.last()->isEmpty()||vvpar2.last()->isEmpty())

{

QMessageBox::warning(this, "Ошибка","Отсутствуют данные");

return;

}

//Вызов метода setData

graph.last()->setData(vvpar1.last(),vvpar2.last(),model3->headerData(0,Qt::Horizontal).toString(),model3->headerData(1,Qt::Horizontal).toString());

//Добавление графика на QGridLayout

grid\_layout->addWidget(graph.last(),graph\_row,graph\_column);

//Изменение строки и(или) столбца положения графика

if(graph\_column>=1)

{

graph\_column=0;

graph\_row++;

}

else

{

graph\_column++;

}

}

**/\*Обработка нажатия кнопку "Очистить" во вкладке "Фильтр записей"**

**\*Входные данные: модель таблицы экспериментов model;**

**\*таблица QTableView.**

**\*Результат: очистка фильтра модели model.**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

void MainWindow::on\_pushButton\_16\_clicked()

{

model->setFilter("");

ui->tableView->resizeColumnsToContents();

}

# *Приложение Ж. Листинг заголовочного файла mygraphicview.h.*

**/\*Описание полей и методов класса построения графиков**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#ifndef MYGRAPHICVIEW\_H

#define MYGRAPHICVIEW\_H

//Подключение необходимых библиотек

#include <QWidget>

#include <QGraphicsView>

#include <QGraphicsScene>

#include <QGraphicsItemGroup>

#include <QTimer>

// Создание класса, унаследованного от QGraphicsView

class MyGraphicView : public QGraphicsView

{

Q\_OBJECT

public:

**/\*Конструктор. Инициализация полей класса построения графиков**

**\*Разработал студент А-08-17 Полушин А.**

**\*Дата написания 01.04.2021.**

**\*Версия 1\*/**

explicit MyGraphicView(QWidget \*parent = 0);

~MyGraphicView(); //Деструктор

private slots: //Определяем слот с модификатором доступа private

**/\*Обработчик переполнения таймера. Рисование графика**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*флаг состояния данных data\_flag;**

**\*флаг зависимости от времени no\_time;**

**\*группы элементов group\_1, group\_2;**

**\*сцена прорисовки scene;**

**\*массив времен и параметров data\_time, data\_param;**

**\*Разница между датами в секундах difDate;**

**\*начальное время zero, zero2;**

**\*максимальное значение параметров по модулю max;**

**\*название параметра param\_str.**

**\*Результат: изображение графика, оси графика, сетка, значения точек, названия осей.**

**\*Метод решения: преобразование значений в координаты**

**\*Дата написания 14.04.2021**

**\*Версия 1\*/**

void slotAlarmTimer();

private: //Объявление полей и методов класса с модификатором доступа private

QGraphicsScene \*scene; // Объявляем сцену для отрисовки

QGraphicsItemGroup \*group\_1; // Объявляем первую группу элементов

QGraphicsItemGroup \*group\_2; // Объявляем вторую группу элементов

QTimer \*timer; // Таймер для задержки отрисовки.

//Зависимость параметра от времени

QVector<QDateTime> \*data\_time; //Указатель на динамический массив QDateTime

QVector<QString> \*data\_param; //Указатель на динамический массив QString

bool data\_flag; //Установлены данные – true, иначе – false.

int count\_data; //Количество данных

qint64 difDate; //Разница времен между самой поздней датой и самой ранней в секундах

int zero, zero2; //Начальные значения времен

double max, max2; //Максимальные значения параметров по модулю

QString param\_str, param\_str2; //Название параметров (для подписи осей)

bool no\_time; //Зависимость параметра от другого параметра – true, иначе (зависимость параметра от времени) – false

//Зависимость параметра от другого параметра

QVector<double> \*vect\_par1,\*vect\_par2; //Указатели на динамические массивы типа double

**/\*Обработка события изменения размера окна**

**\*Разработал студент А-08-17 Полушин А.**

**\*Метод решения: запуск таймера для дальнейшей перерисовки.**

**\*Дата написания 01.04.2021.**

**\*Версия 1.\*/**

void resizeEvent(QResizeEvent \*event) override;

**/\*Рисование графика при зависимости одного параметра от другого**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*группы элементов group\_1, group\_2;**

**\*сцена прорисовки scene;**

**\*массив времен и параметров data\_time, data\_param;**

**\*Разница между датами в секундах difDate;**

**\*начальное время zero, zero2;**

**\*максимальное значение параметров по модулю max, max2;**

**\*название параметров param\_str, param\_str2.**

**\*Результат: изображение графика, оси графика, сетка, значения точек, названия осей.**

**\*Метод решения: преобразование значений в координаты.**

**\*Дата написания 01.05.2021.**

**\*Версия 1.\*/**

void no\_time\_graph(); //отрисовка при зависимости параметра от другого параметра

**/\*Метод удаления элементов из группы**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*группа элементов group;**

**\*сцена прорисовки scene;**

**\*Результат: очистка группы group и удаление соответствующих элементов на сцене scene.**

**\*Метод решения: перебор элементов.**

**\*Дата написания 01.04.2021.**

**\*Версия 1\*/**

void deleteItemsFromGroup(QGraphicsItemGroup \*group\_1);

protected:

**/\*Метод перехвата события колесика мыши для масштабирования сцены**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*сцена прорисовки scene.**

**\*Результат: изменение масштаба сцены.**

**\*Метод решения: перебор элементов.**

**\*Дата написания 01.04.2021.**

**\*Версия 1.\*/**

bool eventFilter ( QObject \* obj , QEvent \* event ) override;

public:

**/\*Метод установки данных для зависимости параметра от времени**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*динамический массив времен;**

**\*динамический массив параметров;**

**\*название параметра.**

**\*Результат: инициализация полей класса.**

**\*Дата написания 01.04.2021.**

**\*Версия 1.\*/**

void setData(QVector<QDateTime> \*datetime,QVector<QString> \*param,QString par\_str);

**/\*Метод установки данных для зависимости параметра от другого параметра**

**\*Разработал студент А-08-17 Полушин А.**

**\* Входные данные:**

**\*динамические массивы параметров;**

**\*названия параметров.**

**\*Результат: инициализация полей класса.**

**\*Дата написания 01.04.2021.**

**\*Версия 1.\*/**

void setData(QVector<double> \*vect\_param1, QVector<double> \*vect\_param2,QString par\_str, QString par\_str2);

};

#endif // MYGRAPHICVIEW\_H

# *Приложение И. Листинг файла mygraphicview.cpp.*

**/\*Реализация методов класса построения графиков**

**\*Разработал студент А-08-17 Полушин А.**

**\*Версия 1\*/**

#include "mygraphicview.h" //Подключение заголовочного файла

//Подключение необходимых библиотек

#include <QGraphicsSceneWheelEvent>

#include <QGraphicsTextItem>

#include <math.h>

#include <QDateTime>

//Реализация конструктора класса

MyGraphicView::MyGraphicView(QWidget \*parent)

: QGraphicsView(parent)

{

no\_time = false;

data\_flag=false;

zero=0;

// Настройка отображения виджета и его содержимого

// Включим полосу прокрутки по горизонтали

this->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOn);

// Включим полосу прокрутки по вертикали

this->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOn);

this->setAlignment(Qt::AlignCenter); // Делаем привязку содержимого к центру

// Растягиваем содержимое по виджету

this->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

// Задание минимальных размеров виджета

this->setMinimumHeight(400);

this->setMinimumWidth(100);

scene = new QGraphicsScene(); // Инициализируем сцену для отрисовки

this->setScene(scene); // Устанавливаем сцену в виджет

scene->installEventFilter(this); //перехватываем событие колесика мыши

scene->setObjectName("scene"); //Устанавливаем имя

group\_1 = new QGraphicsItemGroup(); // Инициализируем первую группу элементов

group\_2 = new QGraphicsItemGroup(); // Инициализируем вторую группу элементов

scene->addItem(group\_1); // Добавляем первую группу в сцену

scene->addItem(group\_2); // Добавляем вторую группу в сцену

// Инициализируем Таймер

timer = new QTimer();

timer->setSingleShot(true);

// Подключаем слот для отрисовки к таймеру

connect(timer, SIGNAL(timeout()), this, SLOT(slotAlarmTimer()));

timer->start(50); // Запускаем таймер на 50 миллисекунд

}

//Реализация метода отрисовки графика.

void MyGraphicView::slotAlarmTimer()

{

if(!data\_flag) //Если данные не установлены – возврат.

{

return;

}

/\*Если необходимо построить график с зависимостью параметра от другого параметра, то

вызываем метод no\_time\_graph()\*/

if(no\_time)

{

this->no\_time\_graph();

return;

}

// Удаляем все элементы со сцены, если они есть перед новой отрисовкой

this->deleteItemsFromGroup(group\_1);

this->deleteItemsFromGroup(group\_2);

int width = this->width(); // определяем ширину виджета

int height = this->height(); // определяем высоту виджета

int n = 10; // Устанавливаем делитель сцены (чтобы график не был впритык к краям)

//Устанавливаем размер сцены по размеру виджета с некоторым запасом

scene->setSceneRect(0,-50,width+100,height+100);

// Задаём чёрную кисть (для отрисовки осей)

QPen penBlack(Qt::black);

// Задаём красную кисть (для отрисовки самого графика)

QPen penRed(Qt::red,5,Qt::SolidLine,Qt::RoundCap);

// Задаём серую кисть (для отрисовки сетки)

QPen penGray(Qt::gray,0.5,Qt::SolidLine,Qt::RoundCap);

//Задаём шрифт (для отрисовки названий осей)

QFont faxis("Times", 10, QFont::Bold);

//Добавление на сцену и в группу оси абсцисс

group\_1->addToGroup(scene->addLine(width/n, height/2,width\*1.05, height/2, penBlack));

//Добавление на сцену и в группу оси ординат

group\_1->addToGroup(scene->addLine(width/n, height\*1.05, width/n, 0, penBlack));

QString zero\_str,zero\_str2;

//Переменная для определения масштаба времени

int time\_interval=-1;

/\*

0-секунды

1-минуты

2-часы

\*/

//Получаем первую дату в массиве

zero\_str = data\_time->value(0).toString();

//Инициализация регулярных выражений для поиска времени в строке даты

QRegExp rg("\\d+"),re("\\d+:\\d+:\\d+");

//Поиск времени в строке даты

re.indexIn(zero\_str,0);

zero\_str=re.cap(0);

zero\_str2=zero\_str;

//Инициализация промежуточного элемента графической библиотеки

QGraphicsItem \*tmp;

int count\_x=15; //Количество точек на оси абсцисс

int difDate\_res; //Результат преобразования разности времён

if(difDate>60) //Если разность дат в секундах больше 60, то измерения в минутах или часах

{

if(difDate/60>60) //Если разность дат в минутах больше 60, то измерения в часах

{

time\_interval=2;

difDate\_res=static\_cast<int>(difDate/60/60)+1;

//Поиск часов и минут в строке времени

rg.indexIn(zero\_str,0);

zero\_str = rg.cap(0);

rg.indexIn(zero\_str2,2);

zero\_str2=rg.cap(0);

zero=zero\_str.toInt();

zero2=zero\_str2.toInt();

int tmp\_int=0; //Количество секунд от первой точки до текущей

int inc=0; //счетчик

//Проход по всем точкам на оси абсцисс

while(tmp\_int<difDate\_res || width/n+(width-(width/n))\*inc/count\_x <= width)

{

//Добавление линии сетки на сцену и в группу и расчет её координат

group\_2->addToGroup(scene->addLine(width/n+(width-(width/n))\*i nc/count\_x,0 ,width/n+

(width-(width/n))\* inc/count\_x,height,penGray));

tmp\_int=1+difDate\_res\*inc/count\_x;

QString tmp\_str;//Значение времени в текущей точке

//Определение часа в текущей точке

tmp\_str=QString::number(zero+difDate\_res\*inc/count\_x>=24 ?

zero+difDate\_res\*inc/count\_x-24\*floor( (zero+difDate\_res\*inc/count\_x)/24 ):

zero+difDate\_res\*inc/count\_x);

//Определение минуты в текущей точке

if(zero2 + difDate\_res \* 60 \* inc / count\_x >= 60)

{

if(( zero2+difDate\_res\*60\*inc/(count\_x)) - ( 60 \* floor(( zero2 + difDate\_res \* 60 \* inc /count\_x )/60)) == zero2)

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number ((59\*floor( (zero2 +difDate\_res \*60\*inc/count\_x ) /60)));

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number((60\*floor(( zero2 +difDate\_res \* 60 \* inc/count\_x)/60))));

}

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number( zero2 + difDate\_res\* 60 \* inc/count\_x);

}

//Установление позиции времени на сцене в текущей точке

tmp->setPos(width/n+(width-(width/n))\*inc/count\_x,height/2);

group\_2->addToGroup(tmp); //Добавление элемента в группу

inc++;

}

} else //Если разность дат в минутах меньше или равна 60, то измерения в минутах

{

time\_interval=1;

difDate\_res=static\_cast<int>(difDate/60)+1;

//Поиск минут и секунд в строке времени

rg.indexIn(zero\_str,2);

zero\_str = rg.cap(0);

rg.indexIn(zero\_str2,5);

zero\_str2=rg.cap(0);

zero=zero\_str.toInt();

zero2=zero\_str2.toInt();

int tmp\_int=0; //Количество минут от первой точки до текущей

int inc=0; //счетчик

//Проход по всем точкам на оси абсцисс

while(tmp\_int<difDate\_res || width/n+(width-(width/n))/count\_x\*inc <= width)

{

//Добавление линии сетки на сцену и в группу и расчет её координат

group\_2->addToGroup(scene->addLine(width/n+(width-(width/n))\*inc/ count\_x,0, width/n+

(width-(width/n))\*inc/count\_x,height,penGray));

tmp\_int=1+ceil(difDate\_res\*inc/count\_x);

QString tmp\_str; //Значение времени в текущей точке

//Определение минуты в текущей точке

tmp\_str=QString::number(zero+difDate\_res\*inc/count\_x>=60 ?

zero+difDate\_res\*inc/count\_x-60\*floor((zero +difDate\_res\* inc/count\_x)/ 60):zero+

difDate\_res\* inc/count\_x);

//Определение секунды в текущей точке

if(zero2+difDate\_res\*60\*inc/count\_x>=60)

{

If((zero2+difDate\_res\*60\*inc/(count\_x))-((60 \*floor((zero2 +difDate\_res\* 60\*inc/

count\_x)/60))==zero2)

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number (59\*floor((zero2

+difDate\_res \*60\*inc/count\_x)/60));

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number(60\*

floor((zero2+difDate\_res\* 60\*inc/count\_x)/60)));

}

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number( zero2+ difDate\_res \*

60\*inc/count\_x);

}

//Установление позиции времени на сцене в текущей точке

tmp->setPos(width/n+(width-(width/n))/count\_x\*inc,height/2);

group\_2->addToGroup(tmp); //Добавление элемента в группу

inc++;

}

}

} else //Если разность дат в секундах меньше или равна 60, то измерения в секундах

{

time\_interval=0;

difDate\_res=static\_cast<int>(difDate);

//Поиск секунд в строке времени

rg.indexIn(zero\_str,5);

zero\_str = rg.cap(0);

zero=zero\_str.toInt();

zero2=0;

int tmp\_int=0; //Количество секунд от первой точки до текущей

int inc=0; //счетчик

//Проход по всем точкам на оси абсцисс

while(tmp\_int<difDate\_res || width/n+(width-(width/n))\*inc/count\_x <= width)

{

//Добавление линии сетки на сцену и в группу и расчет её координат

group\_2->addToGroup(scene->addLine(width/n+(width-

(width/n))\*inc/count\_x,0,width/n+(width-(width/n))\*inc/count\_x,height,penGray));

QString tmp\_str; //Значение времени в текущей точке

//Определение секунды в текущей точке

tmp\_str=QString::number(zero+difDate\_res\*inc/count\_x>=60 ?

zero+difDate\_res\*inc/count\_x-60\*floor((zero+difDate\_res \*inc/count\_x)/60) : zero

+difDate\_res \*inc/count\_x);

tmp\_int=1+ceil(difDate\_res\*inc/count\_x);

//Определение миллисекунды в текущей точке

if(zero2+difDate\_res\*60\*inc/count\_x>=60)

{

If((zero2+difDate\_res\*60\*inc/(count\_x))-((60 \*floor((zero2+ difDate\_res\* 60\* inc/

count\_x) /60))==zero2)

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number (59\*floor((zero2 +

difDate\_res \* 60 \*inc / count\_x)/60));

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number(60 \*floor((zero2+

difDate\_res \* 60\*inc/count\_x)/60));

}

} else

{

tmp = scene->addText( tmp\_str+ ":"+ QString::number( zero2 +difDate\_res\* 60 \*inc

/count\_x);

}

//Установление позиции времени на сцене в текущей точке

tmp->setPos(width/n+(width-(width/n))\*inc/count\_x,height/2);

group\_2->addToGroup(tmp); //Добавление элемента в группу

inc++;

}

}

//Отрисовка оси ординат

int count\_y=20; //Количество точек на оси ординат

//Проход по каждой точке

for (int i=0;i<=count\_y;i++)

{

//Добавление линии на сцену и в группу

group\_2->addToGroup(scene->addLine(width/n,height-height\*(i)/count\_y,width,height-height\*(i)/count\_y,penGray));

//Расчет значения в текущей точке и добавление на сцену

tmp = scene->addText(QString::number(2\*i\*max/count\_y-max));

//Установление позиции значения на сцене в текущей точке

tmp->setPos(width/n-35,height-height\*(i)/count\_y-10);

group\_2->addToGroup(tmp); //Добавление элемента в группу

}

//Отрисовка графика

QString time\_x="Время, "; //Подпись оси ординат

double x1,x2,y1,y2=0; //коэффициенты для определения координат на сцене

for (int i=0;i<data\_time->count()-1;i++)

{

x1=data\_time->value(0).secsTo(data\_time->value(i));

y1=data\_param->value(i).toDouble()/max;

x2=data\_time->value(0).secsTo(data\_time->value(i+1));

y2=data\_param->value(i+1).toDouble()/max;

//Корректировка по интервалу времени

switch (time\_interval)

{

case 0:

break;

case 1:

x1=x1/60;

x2=x2/60;

break;

case 2:

x1=x1/60/60;

x2=x2/60/60;

break;

}

//Расчет координат линии и добавление на сцену и в группу

group\_1->addToGroup(scene->addLine(width/n+((width-width/n)\*x1)/difDate\_res,height/2-

height/2\*y1,width/n+((width-width/n)\*x2)/difDate\_res,height/2-height/2\*y2, penRed));

}

//В зависимости от интервала устанавливаем надпись оси

switch (time\_interval)

{

case 0:

time\_x+="сек:миллисек";

break;

case 1:

time\_x+="мин:сек";

break;

case 2:

time\_x+="час:мин";

break;

}

//Добавление надписей на сцену и в группу

tmp = scene->addText(time\_x,faxis);

tmp->setPos(scene->width()-150, height/2+15);

group\_2->addToGroup(tmp);

tmp = scene->addText(param\_str,faxis);

tmp->setPos(width/n, -0.01\*height/n);

group\_2->addToGroup(tmp);

}

//Реализация метода изменения размера виджета

void MyGraphicView::resizeEvent(QResizeEvent \*event)

{

timer->start(10); // Как только событие произошло запускаем для перерисовки

QGraphicsView::resizeEvent(event); // Запускаем событие родительского класса

}

//Реализация метода построения графика при зависимости одного параметра от другого

void MyGraphicView::no\_time\_graph()

{

// Удаляем все элементы со сцены, если они есть перед новой отрисовкой

this->deleteItemsFromGroup(group\_1);

this->deleteItemsFromGroup(group\_2);

int width = this->width(); // определяем ширину виджета

int height = this->height(); // определяем высоту виджета

int n = 10; // Устанавливаем делитель сцены

//Устанавливаем размер сцены по размеру виджета с некоторым запасом

scene->setSceneRect(0,-50,width+100,height+100);

// Задаём чёрную кисть (для отрисовки осей)

QPen penBlack(Qt::black);

// Задаём красную кисть (для отрисовки самого графика)

QPen penRed(Qt::red,5,Qt::SolidLine,Qt::RoundCap);

// Задаём серую кисть (для отрисовки сетки)

QPen penGray(Qt::gray,0.5,Qt::SolidLine,Qt::RoundCap);

//Задаём шрифт (для отрисовки названий осей)

QFont faxis("Times", 10, QFont::Bold);

//Добавление на сцену и в группу оси абсцисс

group\_1->addToGroup(scene->addLine(width/n, height/2,width\*1.05, height/2, penBlack));

//Добавление на сцену и в группу оси ординат

group\_1->addToGroup(scene->addLine((width+width/n)/2, height\*1.05, (width+width/n)/2, 0,

penBlack));

//Инициализация промежуточного элемента графической библиотеки

QGraphicsItem \*tmp;

int count\_x=15; //Количество точек на оси абсцисс

int count\_y=10; //Количество точек на оси ординат

//Отрисовка оси абсцисс

for (int i=0;i<=count\_x;i++)

{

//Рассчет координат линии сетки и добавление ее на сцену и в группу

group\_2->addToGroup(scene->addLine(width/n+(width-(width/n))\*i/count\_x,0,width/n+(width-

(width/n))\*i/count\_x,height,penGray));

//Расчет значения в текущей точке и добавление его на сцену и в группу

tmp = scene->addText(QString::number(2\*i\*max/count\_x-max,'f',2));

//Установление позиции добавленного элемента

tmp->setPos(width/n+(width-(width/n))\*i/count\_x-10,height/2);

group\_2->addToGroup(tmp); //Добавление элемента в группу

}

//Отрисовка оси ординат

for (int i=0;i<=count\_y;i++)

{

//Рассчет координат линии сетки и добавление ее на сцену и в группу

group\_2->addToGroup(scene->addLine(width/n,height-height\*(i)/count\_y,width,height-

height\*(i)/count\_y,penGray));

//Расчет значения в текущей точке и добавление его на сцену и в группу

tmp = scene->addText(QString::number(2\*i\*max2/count\_y-max2,'f',2));

//Установление позиции добавленного элемента

tmp->setPos((width+width/n)/2-35,height-height\*(i)/count\_y-10);

group\_2->addToGroup(tmp); //Добавление элемента в группу

}

//Отрисовка графика

double x1,x2,y1,y2=0; //коэффициенты для определения координат на сцене

for (int i=0;i<vect\_par1->count()-1;i++)

{

x1=vect\_par1->value(i);

x2=vect\_par1->value(i+1);

y1=vect\_par2->value(i)/max2;

y2=vect\_par2->value(i+1)/max2;

//Расчет координат линии и добавление на сцену и в группу

group\_1->addToGroup(scene->addLine((width+width/n)/2+(((width-

width/n)/(max\*2))\*x1),height/2-height/2\*y1,(width+width/n)/2+(((width-

width/n)/(max\*2))\*x2),height/2-height/2\*y2, penRed));

}

//Добавление надписей на сцену и в группу

tmp = scene->addText(param\_str,faxis);

tmp->setPos(scene->width()-150, height/2+15);

group\_2->addToGroup(tmp);

tmp = scene->addText(param\_str2,faxis);

tmp->setPos((width+width/n)/2, -0.01\*height/n);

group\_2->addToGroup(tmp);

}

//Реализация метода удаления элементов группа

void MyGraphicView::deleteItemsFromGroup(QGraphicsItemGroup \*group)

{

/\* Перебираем все элементы сцены, и если они принадлежат группе,

\* переданной в метод, то удаляем их\*/

foreach( QGraphicsItem \*item, scene->items(group->boundingRect())) {

if(item->group() == group ) {

delete item;

}

}

}

//Реализация метода перехвата события колёсика мыши

bool MyGraphicView::eventFilter(QObject \*watched, QEvent \*event)

{

if(watched->objectName() == "scene")

{

if(event->type() == QEvent::GraphicsSceneWheel)

{

//Преобразование типов

QGraphicsSceneWheelEvent \*wevent = static\_cast<QGraphicsSceneWheelEvent\*>(event);

//изменяем масштаб

this->scale(wevent->delta() < 0 ? 0.9 : 1.1, wevent->delta() < 0 ? 0.9 : 1.1);

//центрируем в точке позиции мыши на сцене

if(wevent->delta() < 0) this->centerOn(wevent->scenePos());

scene->update(); //Обновление сцены

event->accept(); //Применение изменений

return true;

}

}

return false;

}

//Реализация метода установки данных для зависимости параметра от другого параметра

void MyGraphicView::setData(QVector<double> \*vect\_param1, QVector<double> \*vect\_param2,QString par\_str, QString par\_str2)

{

//Инициализация полей класса

data\_flag=true; //Данные установлены

param\_str = par\_str;

param\_str2 = par\_str2;

vect\_par1=vect\_param1;

vect\_par2=vect\_param2;

no\_time = true; // Зависимость параметра от другого параметра

max=0;

max2=0;

double min=0; //Минимальное значение параметров

/\*Проход по всем параметрам vect\_par1 и определение минимального и максимального

значения\*/

for(int i=0;i<vect\_par1->count();i++)

{

if(vect\_par1->value(i)>max)

{

max = vect\_par1->value(i);

}

if(vect\_par1->value(i)<min)

{

min = vect\_par1->value(i);

}

}

//Нахождение максимального по модулю значения

if (fabs(min)>max)

{

max=fabs(min);

}

if(max==0) //Если у параметров нулевое значение, то установление max=1

max=1;

min=0;

/\*Проход по всем параметрам vect\_par2 и определение минимального и максимального

значения\*/

for(int i=0;i<vect\_par2->count();i++)

{

if(vect\_par2->value(i)>max2)

{

max2 = vect\_par2->value(i);

}

if(vect\_par2->value(i)<min)

{

min = vect\_par2->value(i);

}

}

//Нахождение максимального по модулю значения

if (fabs(min)>max2)

{

max2=fabs(min);

}

if(max2==0) //Если у параметров нулевое значение, то установление max2=1

max2=1;

}

//Реализация метода установки данных для зависимости параметра от времени

void MyGraphicView::setData(QVector<QDateTime> \*datetime, QVector<QString> \*param,QString par\_str)

{

//Инициализация полей класса

no\_time=false; // Зависимость параметра от времени

param\_str=par\_str;

data\_time=datetime;

data\_param=param;

data\_flag=true; //Данные установлены

//Определение разницы времени между самой ранней датой и самой поздней в секундах

difDate=data\_time->value(0).secsTo(data\_time->value(data\_time->count()-1));

max=0;

double min=0; //Минимальное значение параметров

QString tmp;//Промежуточная переменная для хранения значений параметров

/\*Проход по всем параметрам data\_param и определение минимального и максимального

значения\*/

for(int i=0;i<data\_param->count();i++)

{

tmp = data\_param->value(i);

tmp.replace(",",".");//Замена точки на запятую для преобразования в тип double

if(tmp.toDouble()>max)

{

max = data\_param->value(i).toDouble();

}

if(tmp.toDouble()<min)

{

min = data\_param->value(i).toDouble();

}

}

//Нахождение максимального по модулю значения

if (fabs(min)>max)

{

max=fabs(min);

}

if(max==0) //Если у параметров нулевое значение, то установление max=1

max=1;

}

1. Под термином «воздушный объект» понимается любое тело, находящееся в воздухе. [↑](#footnote-ref-1)
2. Автоинкремент – функция, которая автоматически генерирует последовательные натуральные числа. [↑](#footnote-ref-2)
3. Виджет - это исходный элемент для создания пользовательского интерфейса в Qt. [↑](#footnote-ref-3)