МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ И СИСТЕМ (ИКСС)
КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ (ПИиВТ)

Объектно-ориентированное программирование
Курсовая работа

ТЕМА: «Класс, характеризующий планеты солнечной системы»

(Вариант 20)

“Класс, характеризующий звездные объекты”
(Пользовательское расширение)

Выполнил: Саганенко Артемий Вадимович
Студент группы ИКПИ-04
Приняла: Петрова Ольга Борисовна

Саганенко А.В.
Подпись _____
Петрова О.Б.
Подпись _____
«___»_____2023

Содержание

Цель работы

Курсовая работа подразумевает в себе создание пользовательского интерфейса для CSV базы данных, что означает разработку оконного приложения windows “с-нуля”, в котором у пользователя есть возможность просматривать (читать), редактировать (изменять и создавать) базы данных.

В данном варианте необходимо создать оболочку для базы данных, характеризующей планеты солнечной системе. По собственной инициативе, этот вариант был расширен до звездных объектов в целом.

Постановка задачи

Задачей данной работы является создание базы данных, работающей на основе контейнера, который следует разработать в процессе выполнения курсовой работы. В качестве прототипа разрабатываемого контейнера можно использовать контейнеры “vector” или “list” из библиотеки STL. Для работы с контейнером следует предусмотреть итератор. Остальные требования к контейнеру определяются студентом самостоятельно.

Анализ задачи

В оболочке должны быть описаны методы для манипуляции базами данных. Было выбрано создать три разных режима работы.

View window (*окно просмотра*): здесь описаны функции открытия и чтения файла, вывод значений, вывод изображения, навигация (через лист и кнопки), открытие контекстного меню (menubar), через которое производиться доступ к другим окнам (режимам работы программы). В данном режиме максимально важна простота и “не перегруженность” интерфейса, он интуитивно понятен и прост. Главная задача режима просмотра, показать пользователю элементы базы данных, окно не нужно нагружать излишней информацией.

Create window (*окно создания*): функции ввода значений, очистки полей, сохранения файла, выбора файла, добавления объекта. Здесь интерфейс должен быть золотой серединой между загруженностью и простотой. Благодаря тому, что режимы разделены между собой, есть возможность убрать лишнее из интерфейса определенного режима. В create window, например не нужно отдельно редактировать каждый объект, с другой стороны, нет просмотра каждого элемента. В то же время сохранение здесь задано с помощью Save As в отличие от Edit window.

Edit window (*окно редактирования*): функции открытия и чтения файла, вывод значений, ввод значений, удаление объекта и его добавление, сохранения файла, очистка полей. Сохранение задано через обычный Save, что означает, что место сохранения файла нельзя изменить, какой файл был открыт тот и сохраняется.

Как ни странно, но в виде используемого контейнера были реализованы несколько разных, но похожих прототипов. Для режима просмотра, был использован собственный класс контейнер QStringList (класс среды Qt), что

является очевидным выбором, так как его основной положительной характеристикой служит то, что он напрямую задается массивом QString. Данное решение намного более надежно чем авторские контейнеры или контейнеры STL. Для создания новой базы данных, из-за однонаправленного статического порядка был выбран написанный в ходе Лабораторной работы номер 5, класс контейнер Queue (однонаправленная очередь). В то же время, для режима редактирования пришлось использовать вектор (STL vector). Так как редактирование, в отличие от чтения или создания требует иногда доступа к “не-боковым” элементам. Данный функционал невозможно реализовать в Queue, можно в QStringList, однако, это было бы слишком простой задачей.

В течение работы возникла дилемма на тему, стоит ли создавать новые (вторичные) формы типа окно (window) или диалоговое окно (dialog). Решение было принято в сторону диалоговых окон, так как при такой форме реализации продукта значительно упростились переходы от главного окна к вспомогательным. С другой стороны, в силу принятия такого решения, пришлось исключить контекстное меню (menubar). Главная сложность заключалась в “выключении” (enabled) главного окна через вторичные. В процессе работы с более ранними проектами на платформе Visual Studio 2017 и ЯП C#, связанной с созданием аудио плеера, главное окно можно было деактивировать, с помощью итерации через все доступные окна. Вероятно такая возможность есть и в Qt. Однако подобная функция не была найдена.

Проект следует выполнять поэтапно. Каждый из этапов целиком и полностью зависит как от факта выполнения предыдущего этапа, но, что не менее важно, от качества выполнения предыдущего.. Таким образом невозможно и неэффективно писать скрипты и построить взаимодействия объектов без разработанного алгоритма. Этапы выполнения работы перечислены ниже:

1. Определить задумку проекта, в частности, какие объекты будут находиться в базе данных;
    a. Определить тему проекта, обычно обозначена вариантом;
    b. Определить поля, необходимые для максимально точной передачи особенностей именно этого объекта.
2. Разработать концепт окон, их вероятный дизайн, сформулировать авторское представление о пользовательском интерфейсе. Также поверхностно определить суть и способ взаимодействия объектов;
    a. Создать визуальной представления, концепты окон
    b. Поверхностно обозначить суть каждого элемента, стоит избегать лишних элементов, которые никак не отразятся на работе программы.
3. На стадии проектирование и алгоритмизации следует продумать как будет работать программа, каждый ее элемент в отдельности и они вместе взятые:
    a. Разработка алгоритма для контейнера (queue, vector, QStringList);
    b. Разработка алгоритма для режима чтения (окно чтения);
    c. Разработка алгоритма для режима создания (окно создания);

d. Разработка алгоритма для режима редактирования (окно редактирования);

e. Дополнительные окна (about, sources, CSV-view);

4. Организация окон или одного окна. Разработка визуального дизайна и построения сетки. Главной целью этого этапа служит создание конечного продукта, в данном случае оболочки без ядра, обеспечивающего: наилучшее восприятие, простоту, в целом всех тех факторов, способствующих лучшему пользовательскому опыту и быстрой навигации в программе;

5. Программирование ключевых функций, классов, которые в себе заключают саму суть программы. Данные ключевые функции могут вызывать исключения в целом в программе. Вызывать общие исключения, зависящие от связующих элементов программы, но в то же время сами по себе эти функции продуманы до мелочей и должны быть локально безупречными;

a. В частности разработка шаблонного класса контейнера;

b. Разработка функций сохранения и открытия баз данных;

6. Разработка вспомогательных функций, которые исправляют общие исключения в программе. Такими функциями могут, например, служить, очистка полей, функция для избежания исключения “вне диапазона.” Функции для визуального улучшения программы и ее оптимизации. В целом те функции которые связывают ядро программы и ее оболочку.

7. Реализация и проверка работы программы. Обработка багов, исключений. Общий тест программы;

8. Тестирование;

9. Заключение, документация, пользовательское руководство и вывод Написание отчета, представление в нем руководства действий для пользователя, и формулировка выводов;

Проект

Для выполнения курсовой работы, необходимо создать оболочку для базы данных, характеризующей планеты солнечной системы. По собственной инициативе, этот вариант был расширен до небесных тел в целом. Каждый объект имеет следующие поля:

- **Названия поля** - пояснение (*пример*)
- **Name** - название звездного тела (*Солнце)*
- **Type** - тип *(Звезда)*
- **Subtype** - под тип *(Желтый карлик)*
- **Radius** - радиус *(700000 км)*
- **Weight** - вес *(1,98 * 10 ^ 30 кг)*
- **Moons** - луны *(нет лун)*
- **Rotation** - время обращения вокруг звезды *(нет)*
- **Population** - население *(0)*
- **Temperature** - температура *(5761K)*
- **Image** - изображение объекта, точнее путь изображения *(.../Sun.png)*

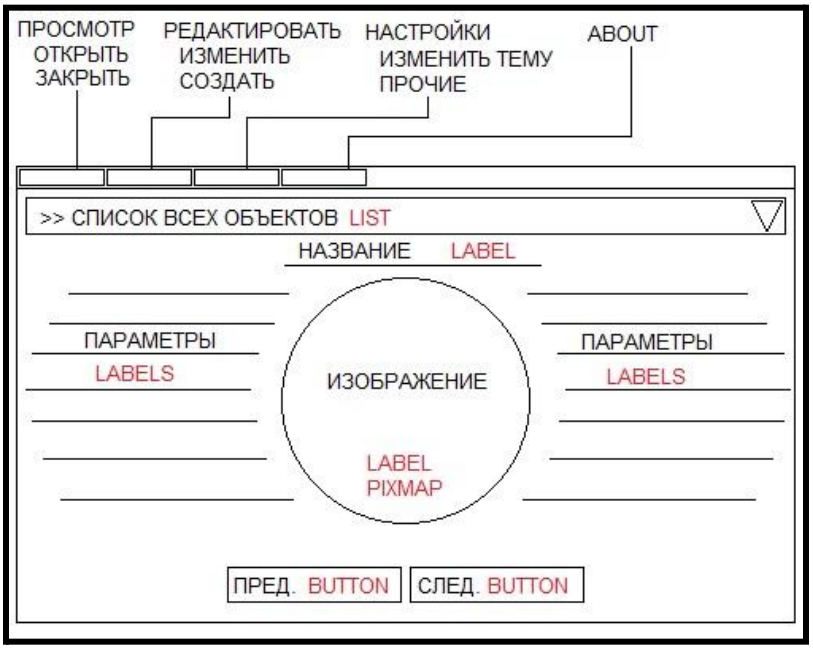
Все эти объекты задаются QString (частная строка среды Qt), что не является однозначно положительной чертой проекта, тем не менее, свою работу выполняет, в то же время суть принципа работы сохраняется. Каждый потенциальный тип int здесь должен иметь единицу измерения, единицу измерения надо реализовывать отдельно, что отразится и на скорости работы программы, и на пользовательское понимания. Т.е. это тоже не наилучшее решение. Пользователю гораздо понятнее и проще будет вводить целиком и значение и единицу измерения. В любом случае не в каких математических операциях и функциях поля не используются. По этой причине гораздо релевантнее использовать QString. Строки легче визуализировать, они понятнее пользователю, и поскольку поля не задействованы в математических операциях, строки остаются в таком же виде, что были заданы изначально.

Создание трех отдельных режимов упрощает работу с программой и общее понимание её сути. Пользователь понимает, что данное окно требует, и означает. Так например просмотр, не требует ничего лишнего кроме как, открытие файла.

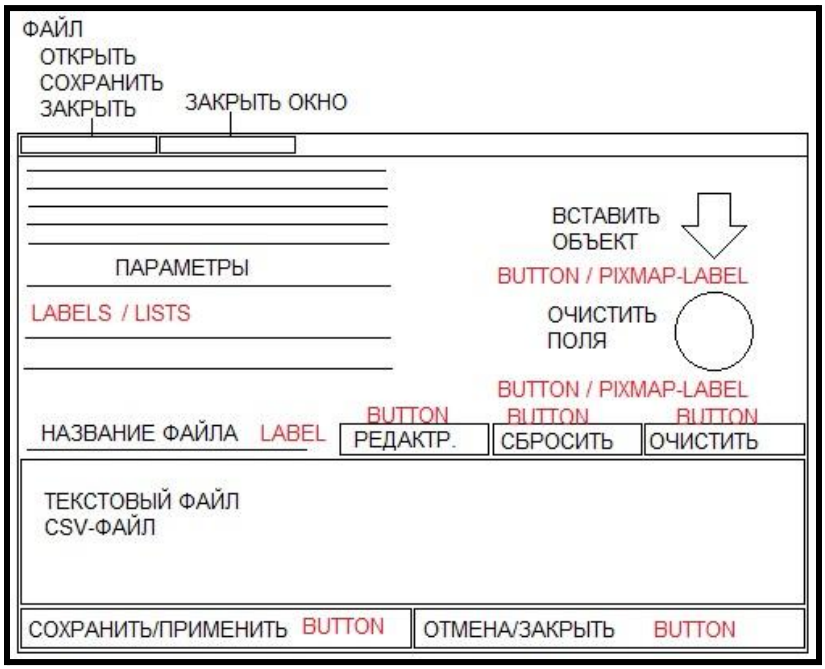
В то же время окна для просмотра и редактирования, немного перегружены, дабы обеспечить достаточный функционал. Необученный пользователь может потеряться в множестве полей и кнопок, однако для этого руководство пользователя и нужно, показать как сделать конкретно необходимое действие.

Разработка концепта

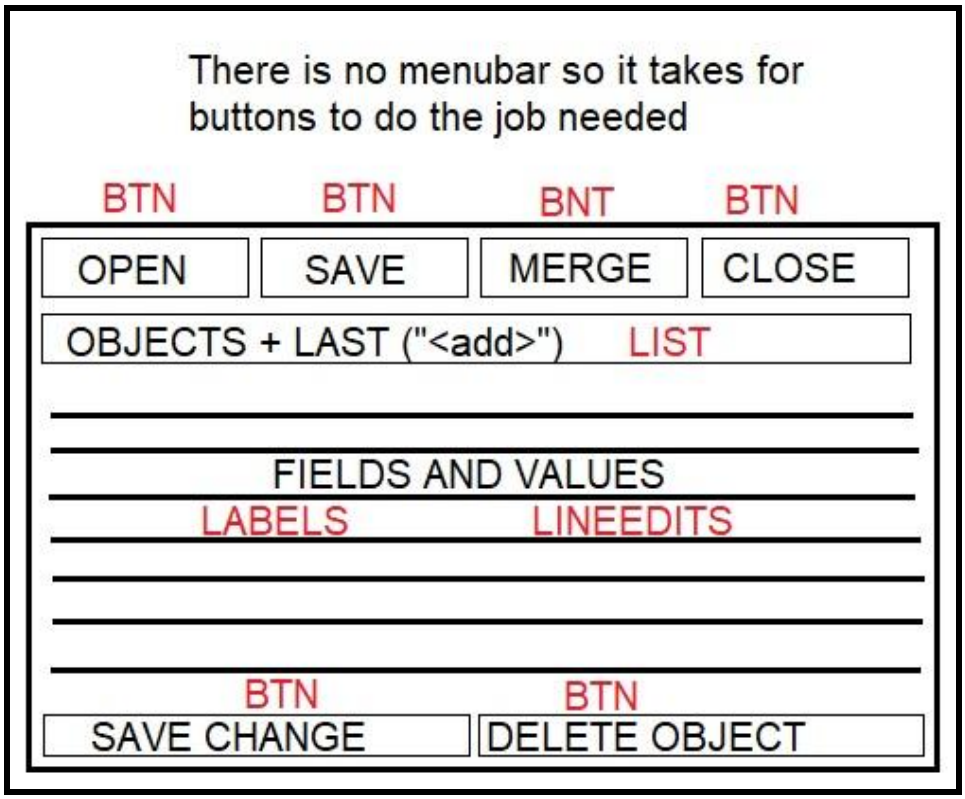
Перед построением сетки и конкретным дизайном, расположением всех элементов уже в Qt, стоит придумать (достаточно представить), как каждое окно и элемент должны выглядеть. В ходе разработки были сделаны следующие наброски.



(**рис. 1** - Концепт *mainwindow*, главного окна, окна для чтения (режима чтения))

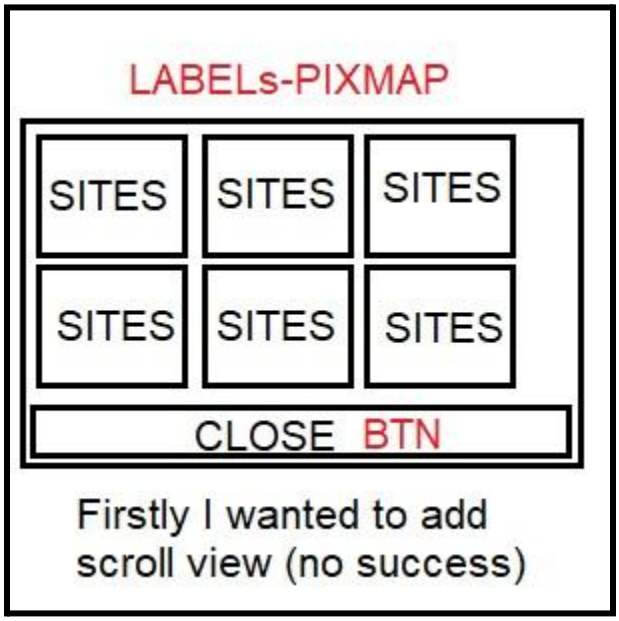


(**рис. 2** - Концепт *createdialog*, окна для создания (режима создания))



(**рис. 3** - Концепт *editdialog*, окна для редактирования (режима редактирования))

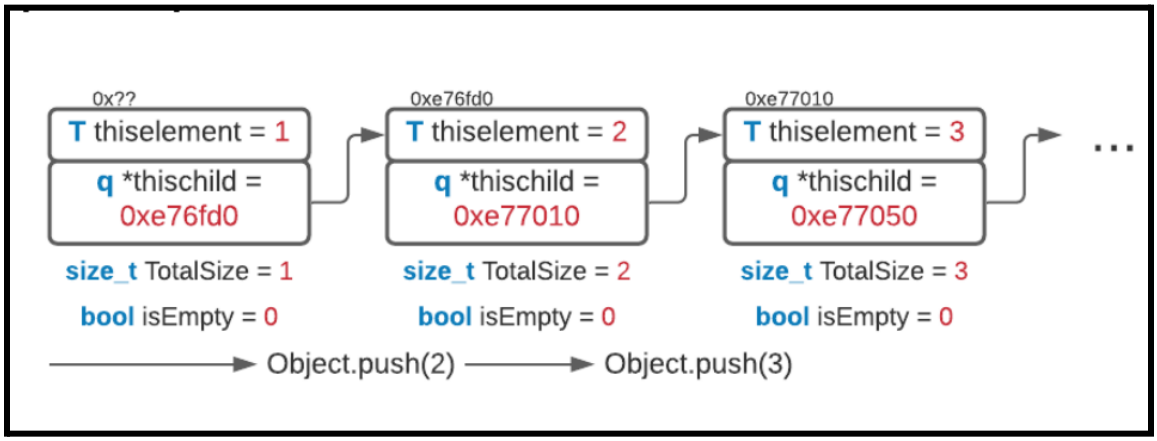
(**рис. 4** - Концепт *aboutdialog)*



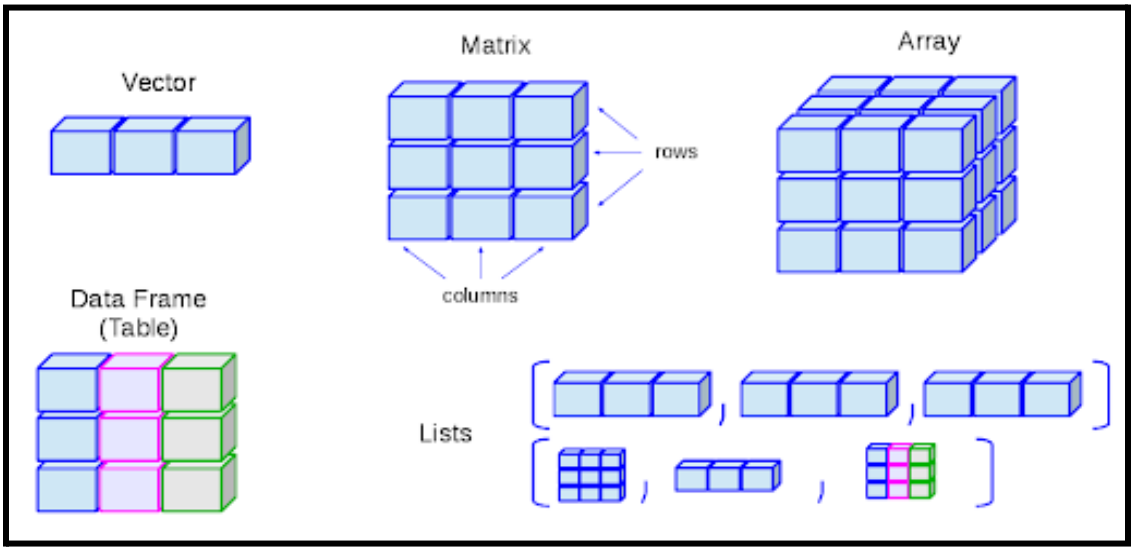
(**рис. 5** - Концепт sourcedialog)

Разработка алгоритма

Как было уже сказано, главными структурными элементами в данной работе являются контейнеры. Всего было использовано 3 разных контейнера. QStringList и Vector из них являются частью, соответственно, Qt libraries и STL (родная библиотека шаблонов C++), в это же время эти два класса контейнера являются двунаправленными, то есть, по сути, могут возвращать элемент любого индекса, что, впрочем, и делает и самостоятельно разработанный queue (однако только в режиме чтения). В то же время Vector и QStringList могут производить запись, удалять элементы и извлекать каждый элемент независимо от соседних. То есть по абсолютному индексу.



(**рис. 6** - Представление очереди на C++)



(**рис. 7** - Общее представление структур данных)

Вторыми по важности функциями являются функции открытия, чтения, записи данных. В данном случае работа не отличается сложностью, однако требует внимания к деталям. Следует описать как можно больше исключений, так как именно в OpenFileDialog, SaveFileDialog, все может выйти из строя. Стоит максимально понятно указать пользователю какой файл от него требуется, где он расположен, почему он не открывается и другую необходимую для пользователя информация. Предполагается, что на данном этапе можно ожидать до 30% потенциальных проблем. После исполнения предыдущего этапа происходит заключительная часть, в которой требуется прописать особенности взаимодействия контейнеров и файлов. На эту часть будут приходиться чуть ли не все исключения или ошибки. В первую очередь речь идет об исключении out_of_range, в последнюю просто о попытке записать недействительное значение. В данном месте работы также будет большое количество так называемых “костылей” - локальных, не самых эффективных, решений проблем. Предотвращение и обработка таких проблем, на самом деле, уже вопрос тестирования и дебаггинга, цель которых максимально оптимально прописать код, лишить его логических дырок.

На этом же этапе нужно объяснить с помощью комментариев внутри кода, как проблема, задача была решена. Описать вкратце, что конкретно подразумевает код.

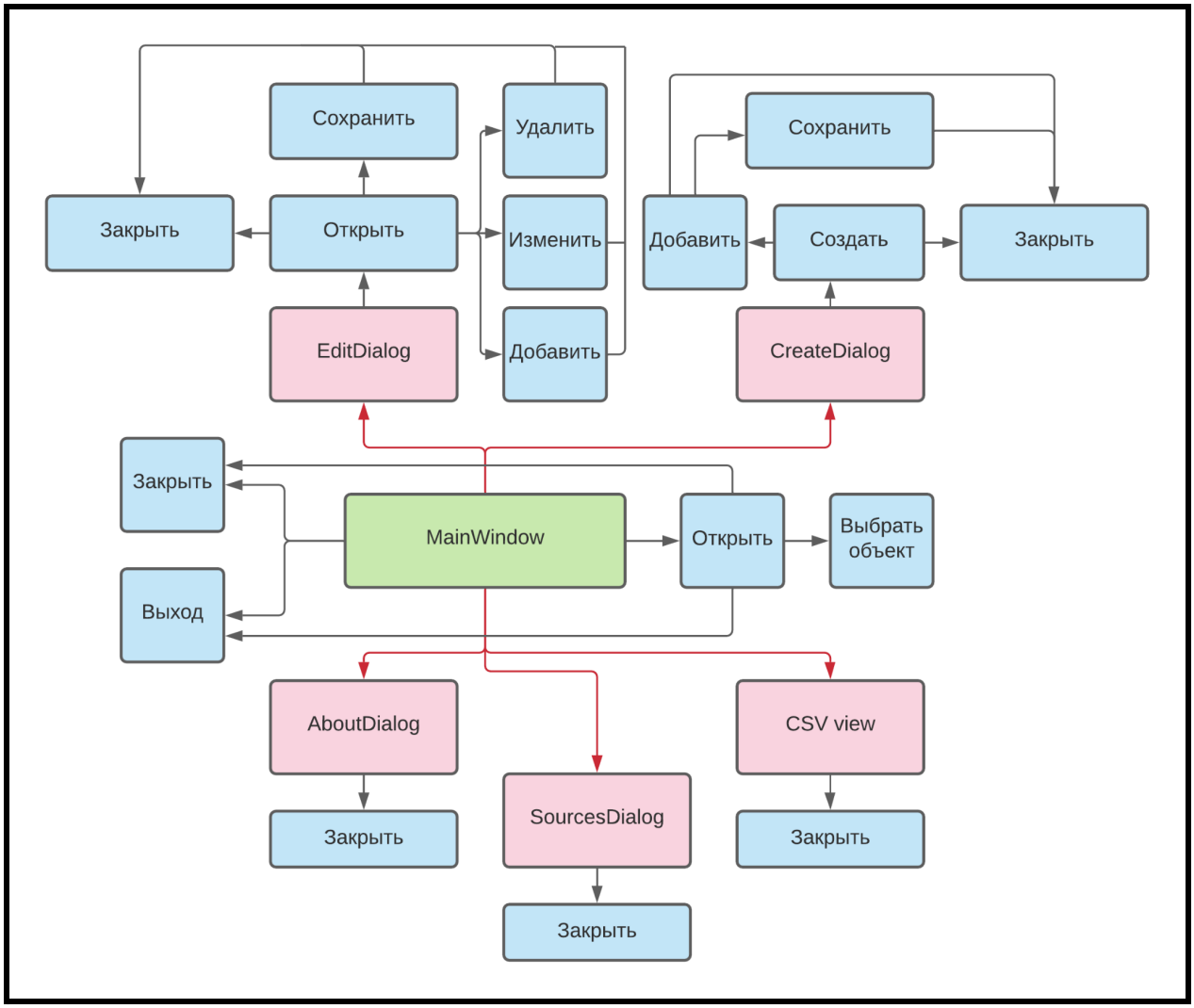
Таблица исключений

Код	Суть	Решение
ER:1 ошибка	*В OpenFileDialog файл не был выбран.*	Сброс функции
ER:2 ошибка	*Открываемый файл имеет неправильный тип.*	Сброс функции
ER:3 ошибка	*В открытом файле неправильный текст.*	Сброс функции
WR:4 предуп.	*Какое-либо из полей пусто, или неправильно задано.*	-
WR:5 предуп.	*Очистка полей, которые уже пусты.*	Сброс функции
WR:ADD предуп.	*При создании/редактировании файла, поле(я) пустое(ые).*	Оно заменяется - “Unknown” - т.е “неизвестно”
ER:6 ошибка	*При сохранении файла, заданная база данных была пуста.*	-
ER:IMAGE ошибка	*Выбранное изображение не существует.*	Сброс функции
ER:QUEUE1 ошибка	*Доступ к пустому контейнеру*	(незапланированный выход)
ER:QUEUE2 ошибка	*Доступ к индексу вне диапазона доступных*	(незапланированный выход)

(**таб. 1** - Прописанные исключения)

В данной таблице отражены все прописанные исключения, которые могут встречаться по мере прохождение через код программы, но, они по большому счету, не навредят работе программы. По причине того, что они прописаны, иначе говоря у программы есть алгоритм действия, который позволяет не закрывать программу в случае конкретной проблемы. В проекте, алгоритмическая сложность немного превышает техническую. Подобные ошибки в программе по большей части появляются скорее всего в свете логических дыр. Что и можно заметить в таблице выше (таб.1).

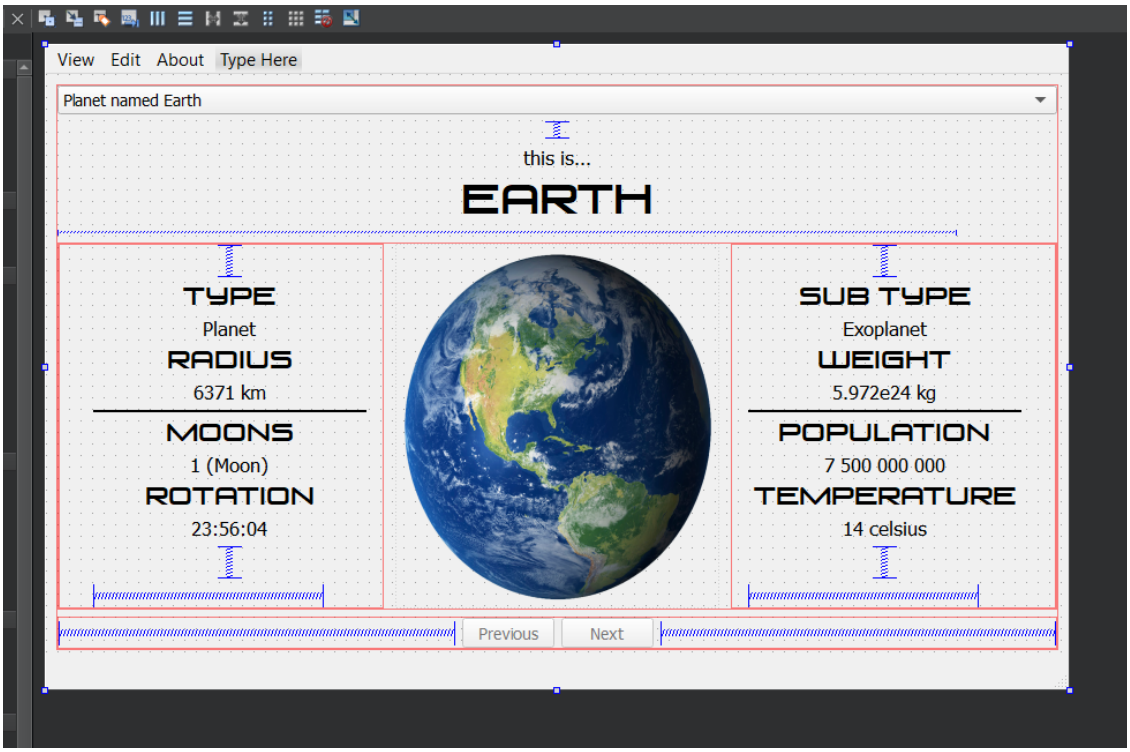
Диаграмма окон



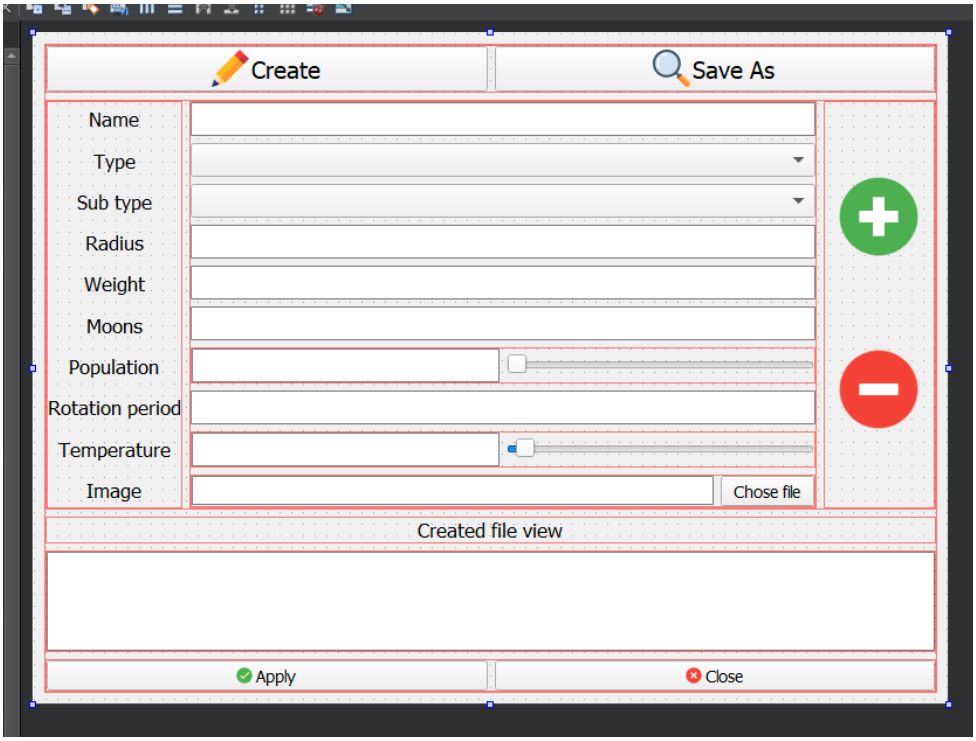
(**рис. 8** - Общий алгоритм)

Более подробно описывать алгоритм нет смысла, каждый процесс (квадрат в диаграмме сверху) можно охарактеризовать по написанному коду. На рисунке 8 отображены все возможности всех доступных окон. MainWindow, расположенное в центре диаграмме, отображает все возможности главного окна, окна просмотра, например, в MainWindow описываются функции: Закрыть базу данных, Открыть базу данных, Выход из программы, Выбор объекта, доступ к окнам AboutDialog, SourceDialog, CSV view, EditDialog, CreateDialog.

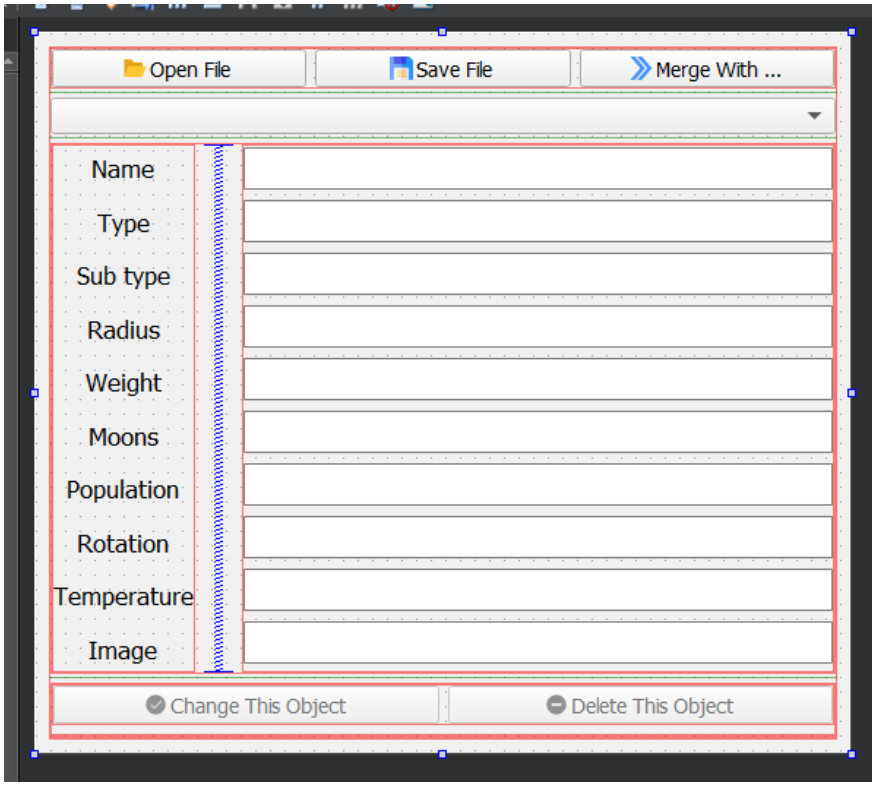
Построение окон



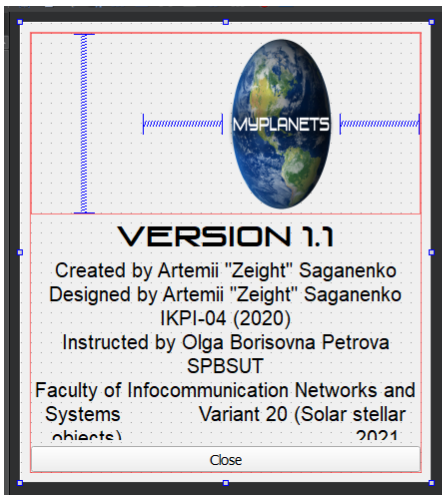
(**рис. 9** - mainwindow)



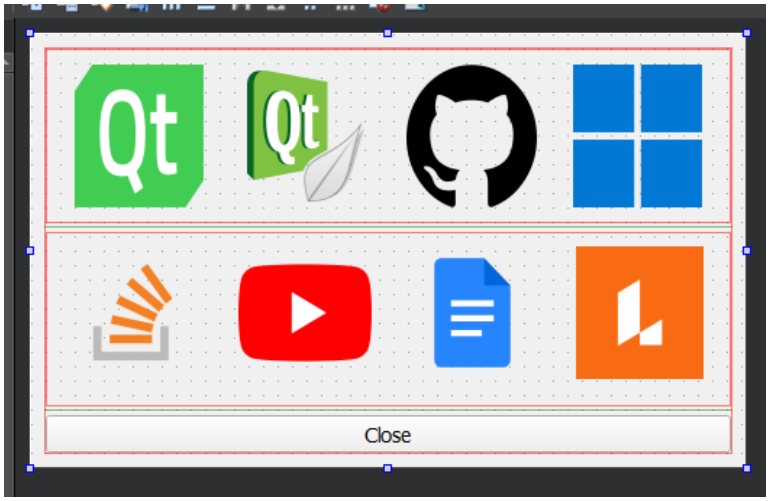
(**рис. 10** - createwindow)



(**рис. 11** - editwindow)



(**рис. 12** - aboutwindow)

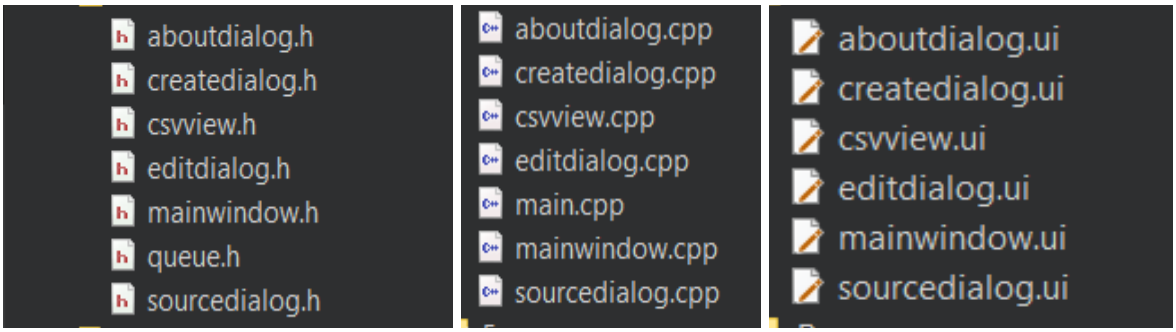


(**рис. 13** - sourcewindow)

Описание программы

Работа была проведена с использованием:

- Языка программирования: C++
- IDE: Qt (environment / GUI) 6.3.0
- Compiler (C++/C): MinGw 11.2.0 64-bit
- Debugger: GNU gdb 11.2 for MinGw 11.2
- CMake Tool: CMake 3.21.1 (Qt)
- ОС: Windows 10 (19044)

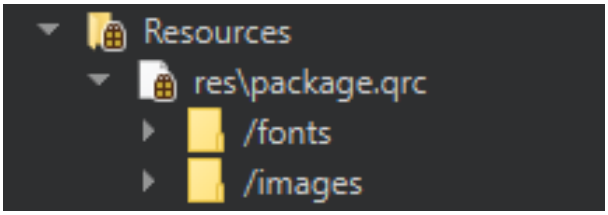


Headers Cpp UI(interface)

(**таб. 2** - файлы)

#include <QApplication>	- Целевое приложение Qt
#include <QMessageBox>	- Окно сообщения Qt
#include <QFontDatabase>	- Для загрузки шрифтов
#include <QFileDialog>	- Диалоговое окно файла
#include <QPixmap>	- Изображение Qt
#include <vector>	- STL структура данных

(**таб. 3** - библиотеки)

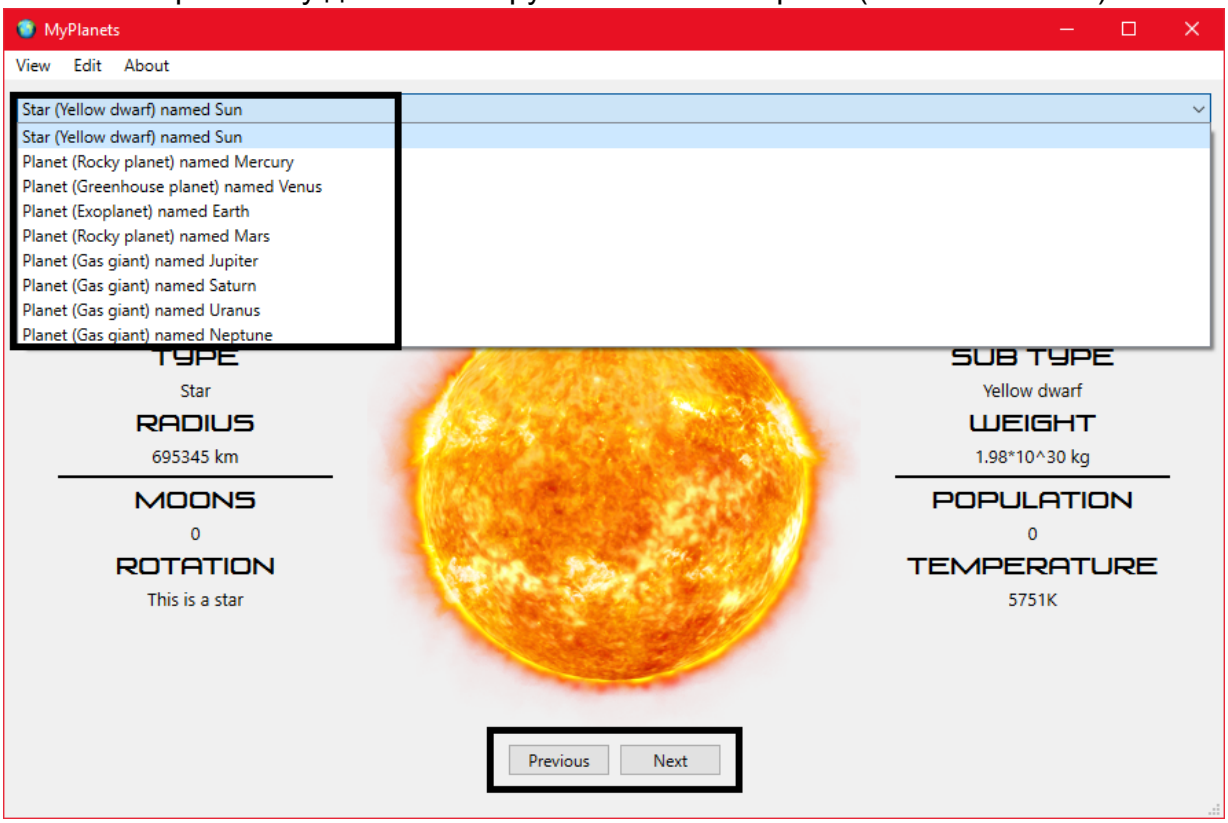


(**таб. 4** - ресурс файлы (изображения, иконки, и т.д)

Руководство пользователя

Просмотр

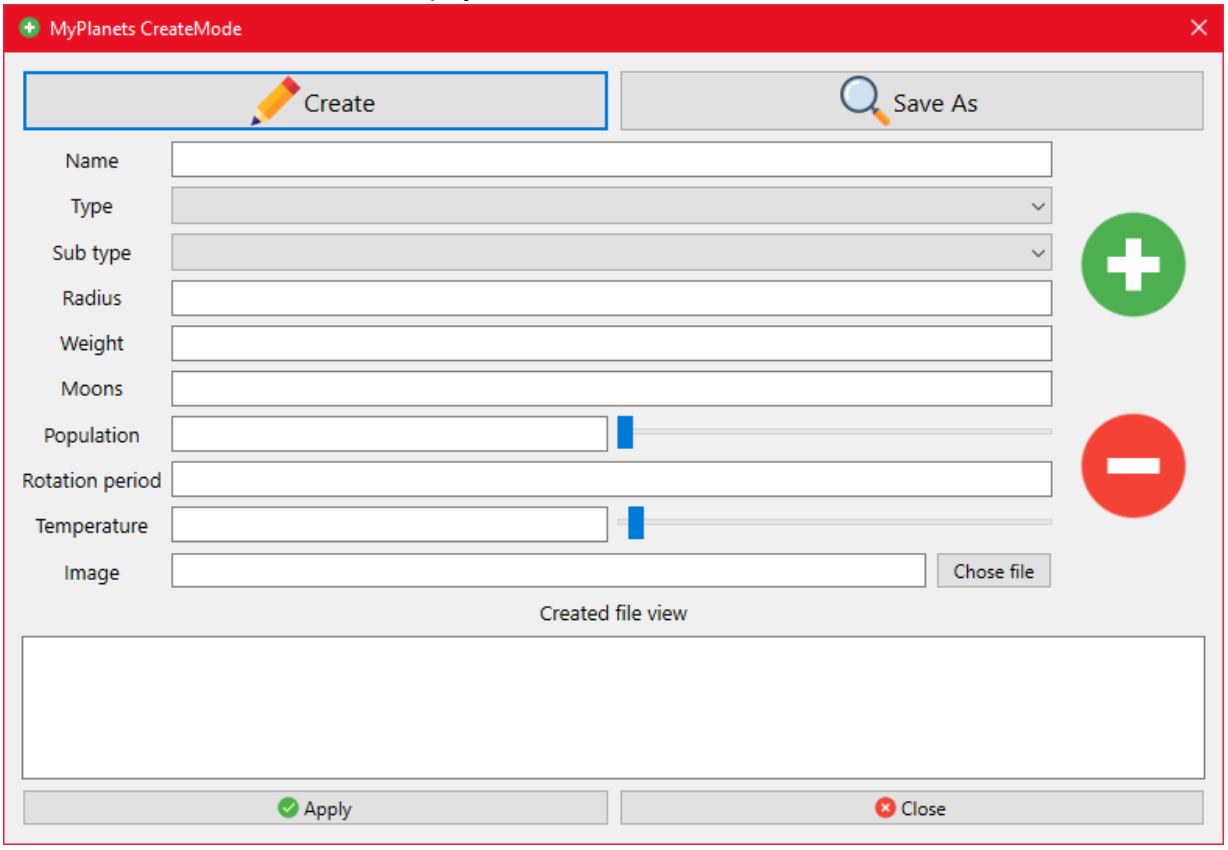
1. Для просмотра базы данных в контекстном меню выберите **View -> Open**
2. Выберите базу данных которую вы хотите открыть (обязательно txt)



3. Просматривайте базу данных с помощью кнопок **Previous, Next** и **ComboBox листа** в верхней части окна
4. Если нужно проверить CSV файл, нажмите **View -> Look CSV file**
5. Там же можно либо закрыть базу данных **View -> Close**, либо закрыть приложение **View -> Exit**

Создание

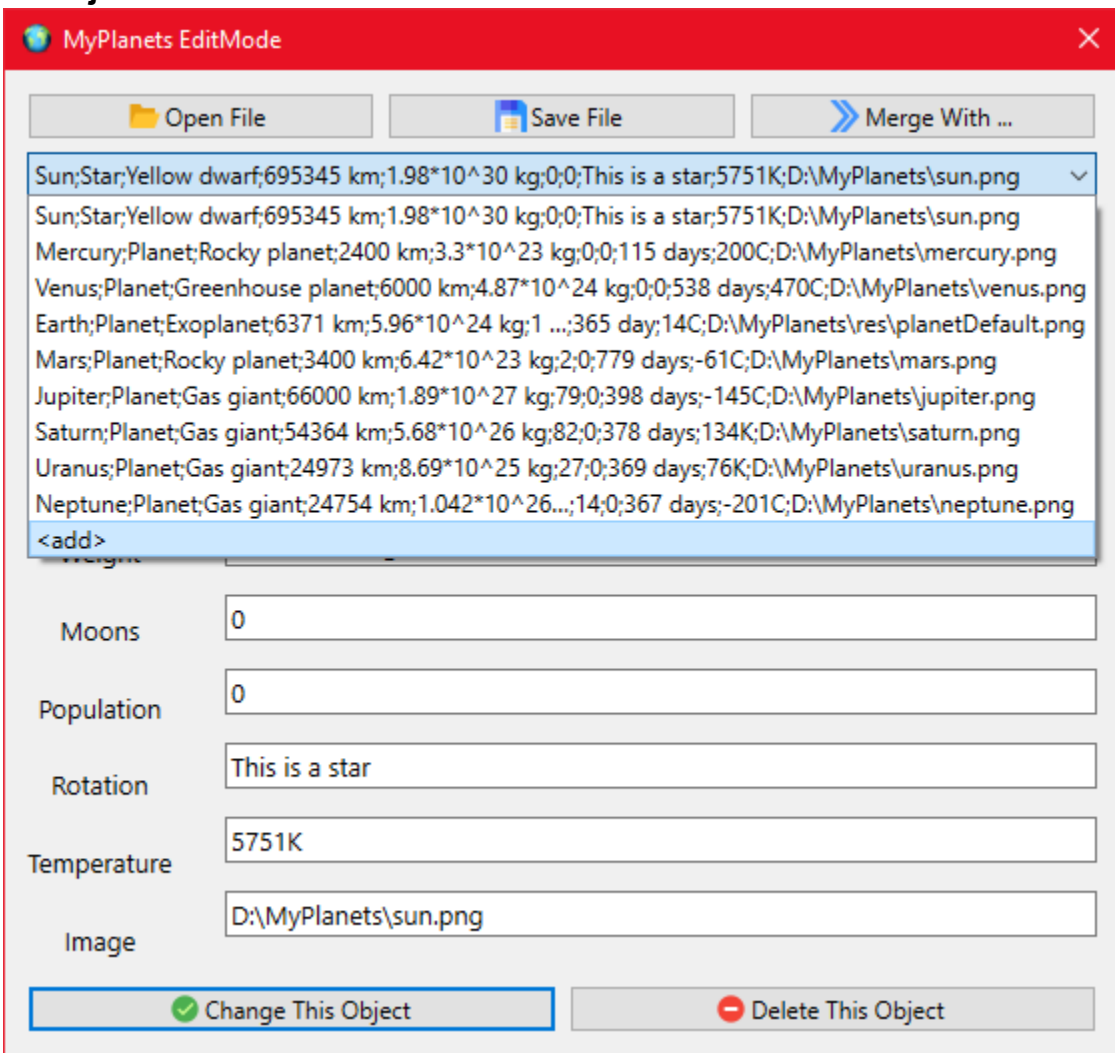
1. Для создания новой базы данных нажмите в главном окне **Edit -> Create** в контекстном меню сверху



2. Редактируйте поля, нажмите “**+**” - чтобы добавить объект, “**-**” чтобы очистить поля.
3. В нижней части окна будет отображен текст только что созданной базы-данных
4. После того как закончите, сохраните файл **SaveAs** или **Apply**. Если какое-то изменение вас не устраивает, вы желаете сбросить все изменения нажмите **Close** или крестик в углу окна

Редактирование

1. Для редактирования базы данных нажмите **View -> Edit**
2. Откройте базу данных которую хотите редактировать кнопка **Open File**
3. С помощью листа выбирайте элемент который хотите изменить. Поменяйте нужные вам поля. После изменения, нажмите **Change This Object**



4. Хотите добавить новый объект, выберете <add> в листе, заполните данные и нажмите **Change This Object.**
5. Если вам нужно удалить конкретно выбранный в листе сверху объект базы данных нажмите **Delete This Object**
6. Нужно слить несколько баз данных нажать **Merge With ...** и выберите файл
7. Не забудьте сохранить! **Save file**

Вывод

В процессе выполнения курсовой работы были получены навыки разработки полноценного ООП проекта, достойного по содержанию и смыслу. Проект - оболочка (пользовательский интерфейс) для работы с CSV базой данных. В Для решения поставленных задач были созданы способы манипулирования базы данных (просмотр элементов, их изменение, добавление новых, удаление старых). В ходе разработки были протестированы все функции на работоспособность. Дальнейшая более детальная и точная наладка работы программы является задачей оптимизации программного обеспечения. Программа вполне явно описывает небесные тела представляя законченный набор определенных параметров наиболее понятных пользователю. Каждый объект имеет 10 полей, включая пользовательское изображение. При редактировании каждое поле может быть задано пользовательски, а изображение выбрано из файлов компьютера.

Проведена достаточно подробная работа со средой разработки Qt. Изучены основы среды, её особенности.

Были закреплены знания ЯП C++, языка UI разметки, а также получено представление, о комплектации ПО разработанного в Qt.

Получен навык написания базовой документации и, что самое интересное, руководства пользователя.

Таким образом можно сделать вывод о том, что в связи с достижением цели работы и успешной поэтапной реализации всех ее задач, данную курсовую работу можно считать завершенной.

Источники и ресурсы

https://doc.qt.io/ (Qt)
Официальная документация IDE Qt; помощь со средой, интерфейсом, дизайном и кодом.

https://docs.microsoft.com/ru-ru/cpp/ (Microsoft)
Официальная документация ЯП C++ от главного его куратора Microsoft; помощь с кодом.

https://www.qtcentre.org/ (QtCentre)
Форум пользователей IDE Qt; помощь со средой, интерфейсом и кодом.

https://ru.stackoverflow.com/ (Stackoverflow)
Форум для программистов; помощь со средой, интерфейсом, дизайном и кодом.

https://docs.google.com/ (Google Docs)
Онлайн редактор текста и документов; помощь с отчетом.

https://www.lucidchart.com (LucidChart)
Онлайн ресурс для создания графиков и диаграмм; помощь с диаграммами.

https://www.youtube.com/ (YouTube)
Видеохостинг, главным образом были использованы каналы: (ProgrammingKnowledge) https://www.youtube.com/channel/UCs6nmQViDpUw0nuIx9c_WvA, (VoidRealms) https://www.youtube.com/user/VoidRealms. Помощь со средой, интерфейсом, дизайном и кодом.

https://www.flaticon.com (Flaticon)
Источник иконок и изображений.

https://icons8.com (Icons8)
Источник иконок и изображений.

https://www.nasa.gov (National Aero-space Agency)
Источник изображений для демонстрации работы программы.

https://replit.com/~ (Replit)
Облачная IDE и по совместительству форум для программистов; помощь с кодом.

https://www.sololearn.com (SoloLearn)
Онлайн платформа для обучения языкам программирования и по совместительству форум для программистов; помощь с кодом.