Романов Вячеслав 931921

1.Постановка задачи

Разработка приложения печати графиков.

Исходные входные  данные

***Исходные данные для печати*** соответствуют некоторому типу, который определятся пользователем. Данные определенного типа могут отображаться конкретным графиком, который ориентирован на этот тип данных.

Примеры данных.

1. Данные характеризуются парой **[значение, дата]**, хранятся в БД SQLite(архив с файлами прилагается).

        Информация по организации работы с БД SQLite.

<https://habr.com/ru/post/128836/>

2. Данные представлены JSON файлом. Формат данных [**значение , дата].**

   Информация по организации взаимодействия с JSON файлами.

[https://doc.qt.io/qt-5/json.html](  %09%20https://doc.qt.io/qt-5/json.html)

**Дано:** предложен начальный вариант архитектуры ПО, в которую требуется внести изменения с целью снижения связности архитектуры. Используется принцип внедрения зависимости. Реализация внедрения зависимости с помощью IOC контейнера.

При разработке архитектуры учесть

1. Возможность добавления новых графиков (графики отличаются видом и данными
2. Изменение визуального стиля графиков (цветной, черно белый)

**Общие требования к GUI**

1. Загружаем данные, путем выбора нужного файла. Данные в ПО не отображаем, отображаем только график, построенный относительно считанных данных.
2. При печати в pdf выбираем место сохранения графика.

**Использование предложенной реализации IOC контейнера  на с++**

Необходимо разобраться в предложенной реализации IOC контейнера.

Код сопроводить соответствующими объяснениями.

Рассмотреть необходимые темы, используемые при реализации IOC контейнера

2.Предполагаемое решение

Для реализации поставленной задачи будут использоваться IOC контейнер и Фабрика.  
Инверсия управления(IOC) – это важный принцип объектно-ориентированного программирования, используемый для уменьшения связей в компьютерных программах. Также он входит в пятёрку важнейших принципов SOLID. IoC уменьшает зависимость между компонентами системы.

Фабричный метод – это паттерн, который определяет интерфейс для создания объектов некоторого класса, при этом непосредственное решение о том, объект какого класса будет создан происходит в подклассе.

Для начала я реализовал класс-фабрику I\_Print, и два его наследника Print\_Bar и Print\_Pie, которые нужны для вывода графиков на диаграмму, также в них предусмотрена чёрно-белая палитра.  
Также был написан класс-фабрика I\_Reader, от которого наследуется также два класса: Reader\_SQL\_lite, Reader\_JSON. Соответственно они нужны для считывания информации из баз данных, и если у нас файл .sqlite, то будем пользоваться Reader\_SQL\_lite, а если .json Reader\_JSON.  
Для вывода самих диаграмм и считывания из файлов разного формата был написан класс Chart, который работает на основе IOC контейнера. За счёт IOC контейнера мы сможем дописать ещё один класс реализующий другой тип данных никак не взаимодействуя с основным интерфейсом.

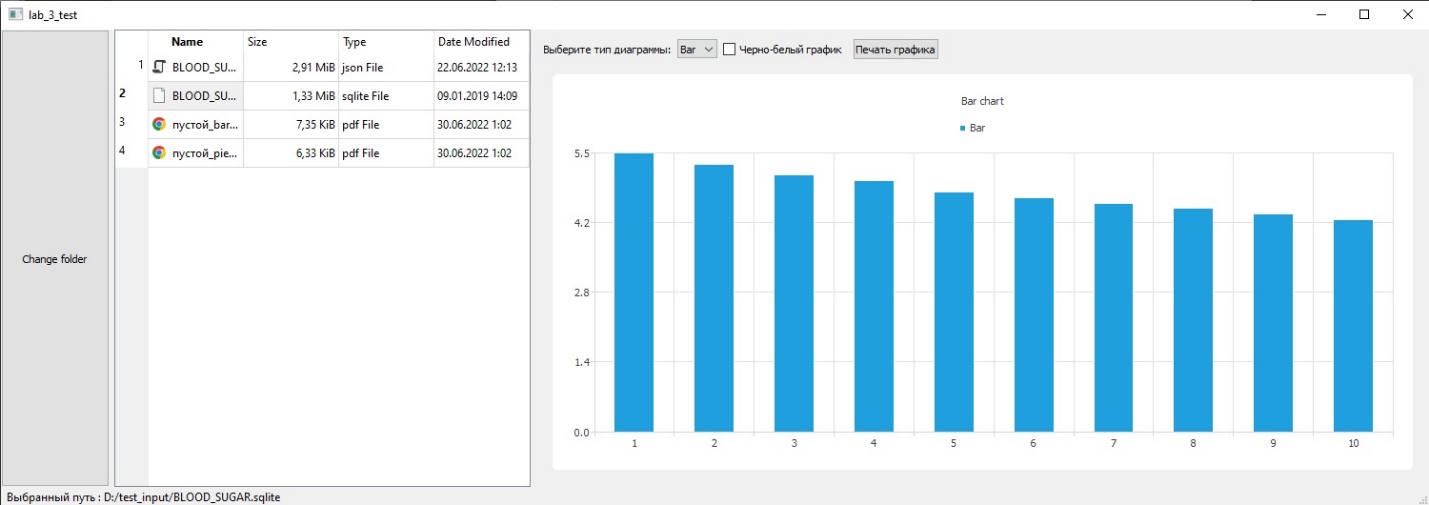
Был написан класс ChartWidget, который отвечает за работоспособность окна, то есть в нём находятся объекты кнопок(принт, черно-белый график, выбор диаграммы). Также в нём находится метод, который возвращает путь для сохранения pdf, метод реализующий печати pdf файла в указанное место, метод обновления данных при смене базы данных.

Был изменён класс MainWindow, само окно я разделил, и добавил таблицу с файлами, а также сами диаграммы. Написан метод диалогового окна, через которое мы выбираем путь до файла.

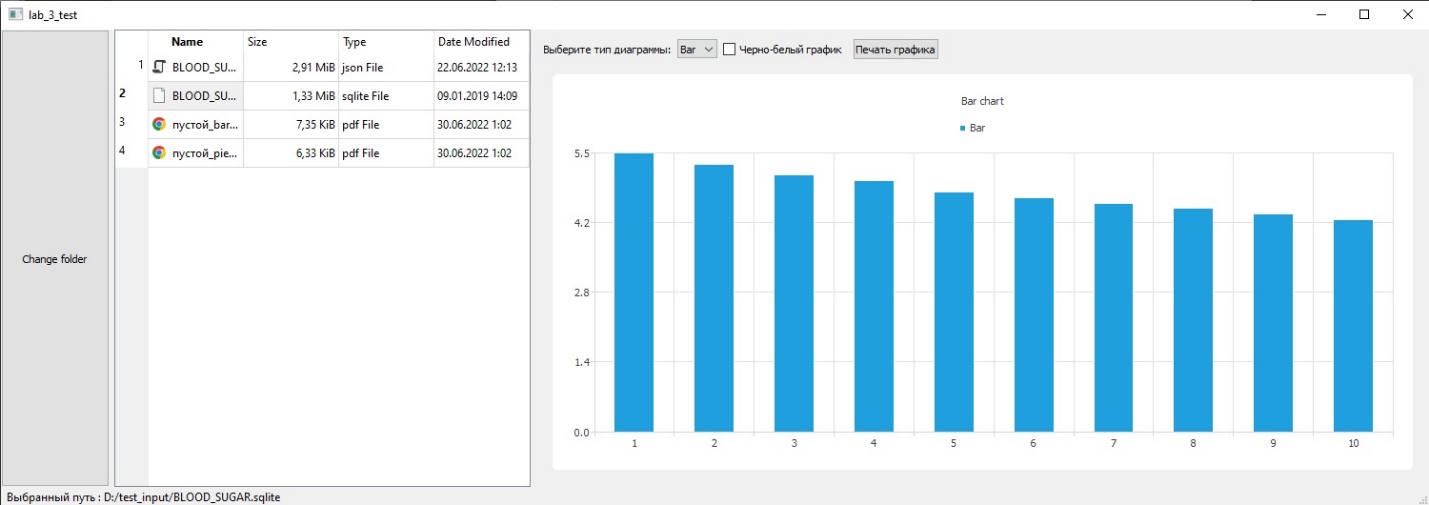
3. Код программы:

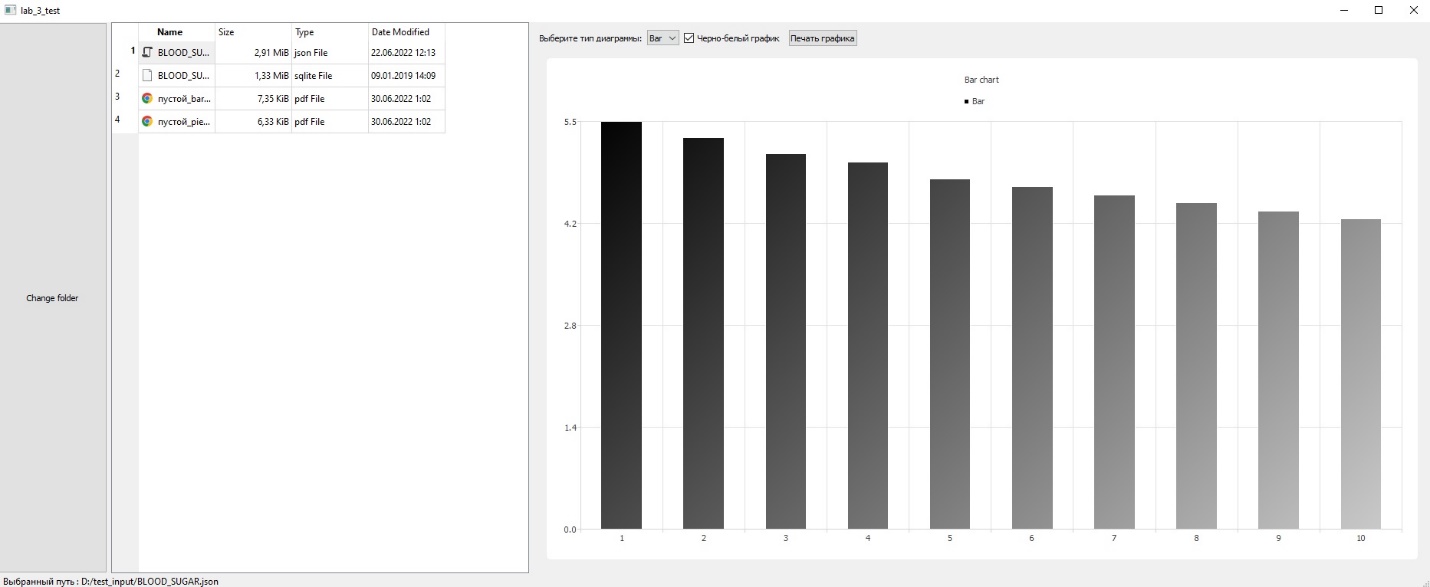
<https://github.com/Dontemon/lab3>

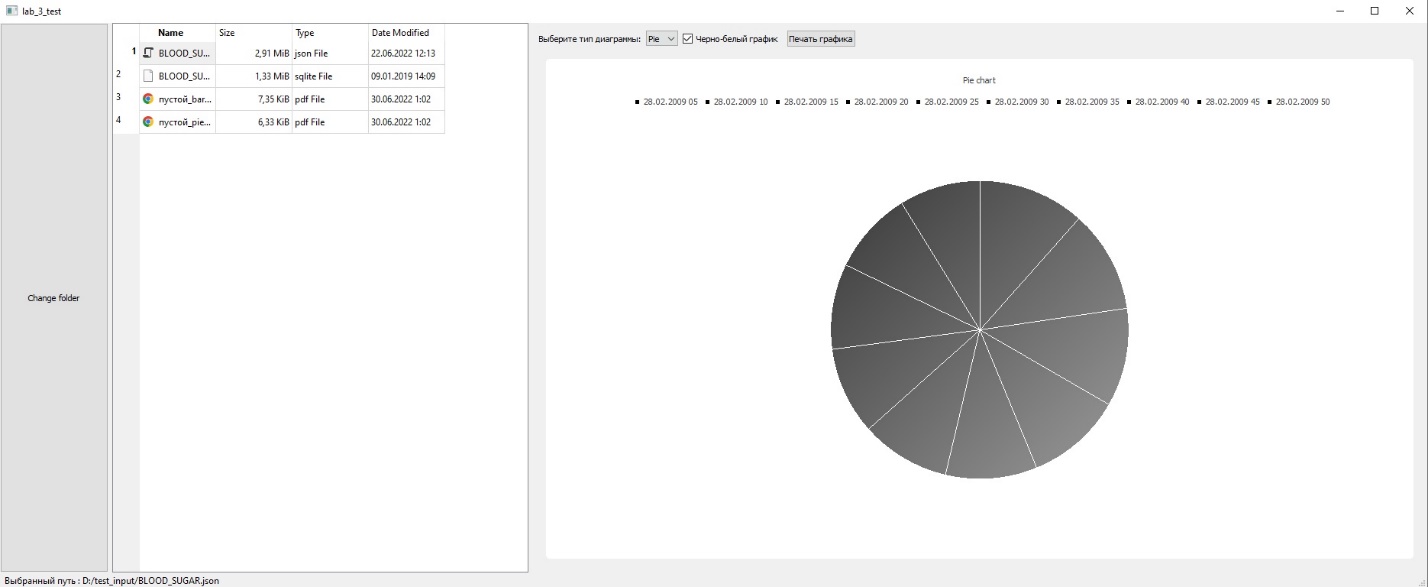
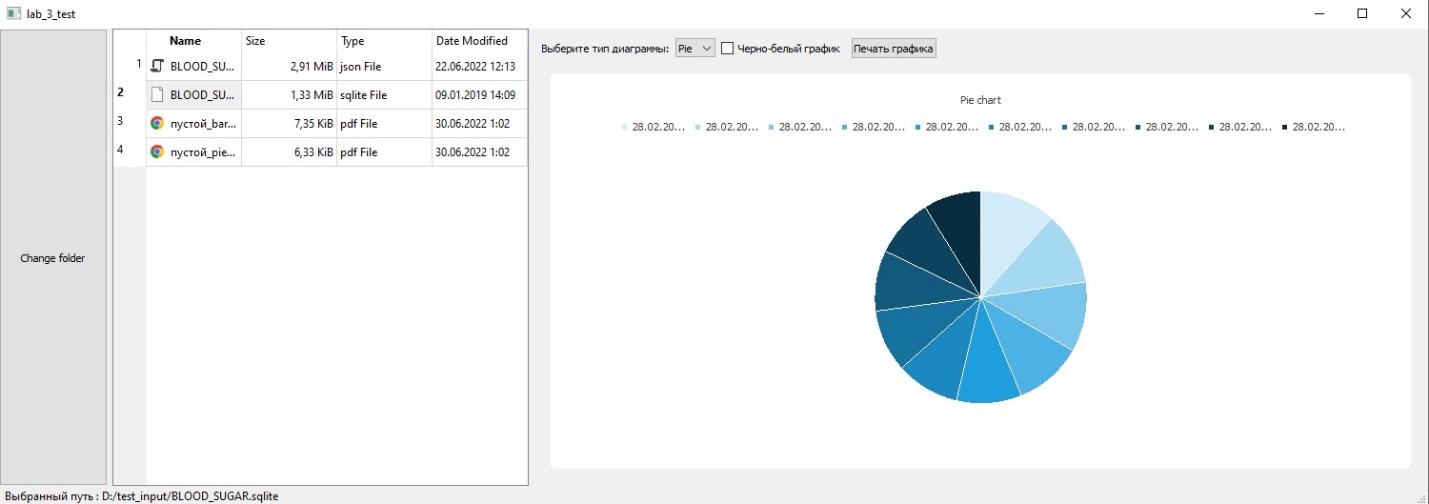
4.Руководство пользования

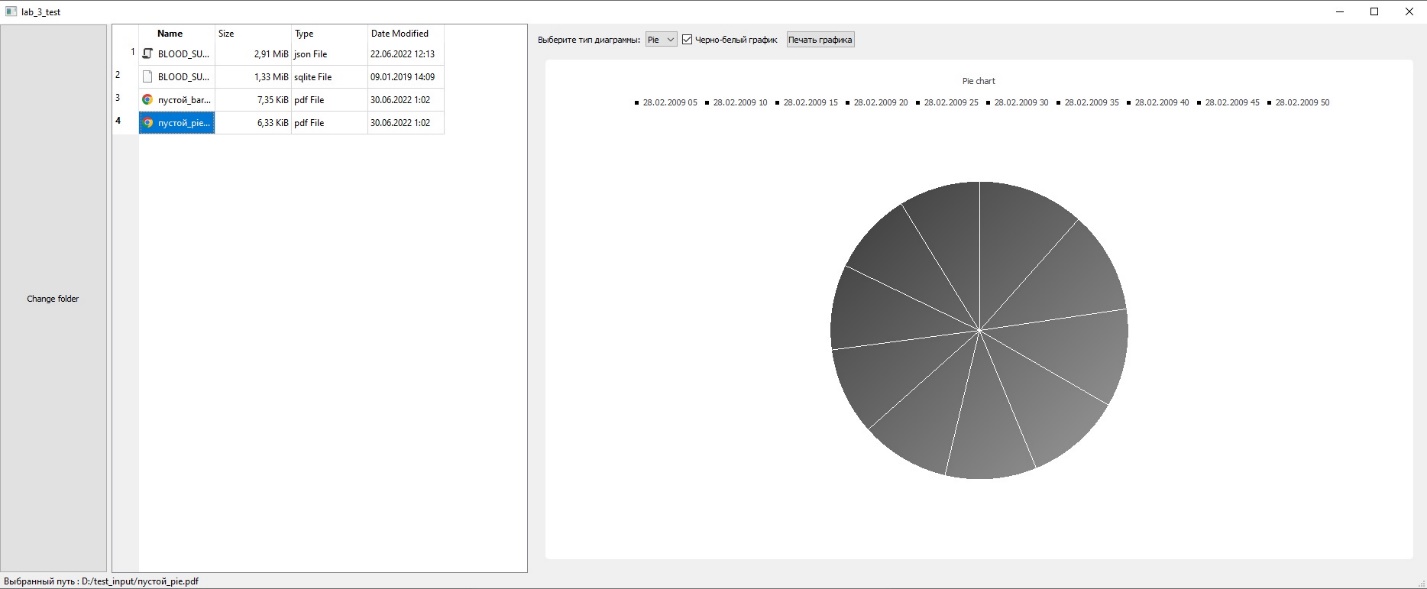
При запуске программы открывается окно, нажимая на большую кнопку Change folder можно выбрать папку. В самой папке хранятся файла типа .json и .pdf.(если их в папке нет, то папка показывается как пустая), соответственно, если открыта нужная папка, надо кликнуть на нужный файл, чтобы вывелась диаграмма по выбранное базе данных, формата .json или .pdf  
Выбрав нужный файл, в правой части окна появится диаграмма, которую можно будет поменять при необходимости. Сверху над диаграммой расположены 3 кнопки, одна из них собственно и переключает тип диаграммы. Также можно сделать диаграмму чёрно-белой, поставив галочку в соответствующем пункте. И есть кнопка ПЕЧАТЬ ГРАФИКА, которая позволяет в выбранном месте сохранить диаграмму, которая выведена в правой части окна.   


5.Тестирование

При выборе корректного файла .json или .pdf формата будет выведено такое окно

Соответственно, если поставить галочку, чтобы график стал чёрно-белый(также я поменял тут тип файла, чтобы непоказывать одно и тоже дважды), то программа выведет такую диаграмму.(левая верхняя часть диаграммы всегда будет наиболее тёмной, а правая нижняя наиболее светлой, так как при написании кода был использован QLinearGradient)

Соответственно, если поменять тип диаграммы на Pie, то правая часть окна изменится и теперь там будет круговая диаграмма  
и также другого формата и цветная диаграмма  
При этом, если выбрать неверный файл, например, файл с названием пустой\_bar.pdf, который является картинкой, а не базой данных, то ничего не произойдёт, программа просто будет выводить последний верный файл



Также если в самом начала, после запуска программы, когда ни одна база данных не была выбрана, нажать на кнопку Печать графика и выбрать имя и место сохранения PDF, то программа просто сделает пдф с разметкой, которая есть с самого начала, но без диаграммы(так как собственно она не была выбрана).  
