**Разработка приложения печати графиков.**

1. **Постановка задачи**

Разработать приложение для печать графиков в .pdf. 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).

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).

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

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. Предлагаемое решение.**

Предлагаемое решение излагается на уровне идеи. Описывается общее решение задачи, с указанием структуры решения, в частности описание предлагаемых сущностей(классы), функций. Если в качестве решения предлагается архитектура ПО, она также описывается и представляется с помощью UML диаграммы. Если предложенная архитектура ПО базируется на одном их известных паттернов, то в отчете обязательно описывается назначение этого паттерна его архитектура, а также какие шаги были выполнены для применения этого паттерна для решения задачи. (Таким образом сдаете теоретическую часть, рассматриваемой темы + ваше решение)

* S: Single Responsibility Principle (Принцип единственной ответственности).
* O: Open-Closed Principle (Принцип открытости-закрытости).
* L: Liskov Substitution Principle (Принцип подстановки Барбары Лисков).
* I: Interface Segregation Principle (Принцип разделения интерфейса).
* D: Dependency Inversion Principle (Принцип инверсии зависимостей).

При разработке учитывался принцип инверсии зависимости. Согласно ему, модули верхних уровней не должны зависеть от модулей нижних уровней, оба уровня должны зависеть от абстракций. А абстракции в свою очередь не должны зависеть от деталей, а детали должны быть зависимы от абстракций. Нам нужно уменьшить связанность программы. Реализация скрыта и может быть заменена любой другой, реализуем с помощью IOC-контейнера.

IOC-контейнер -- это специальный объект-сборщик, который на основании схемы зависимостей между классами и абстракциями может создать граф объектов. Любой IOC-контейнер реализует принцип инверсии зависимостей. Отвечает за управление зависимостями. Типы регистрируются в контейнере, а затем используются для создания объектов. Контейнер автоматически определяет отношения зависимостей. Многие контейнеры IOC также позволяют управлять временем существования объекта и областью действия.

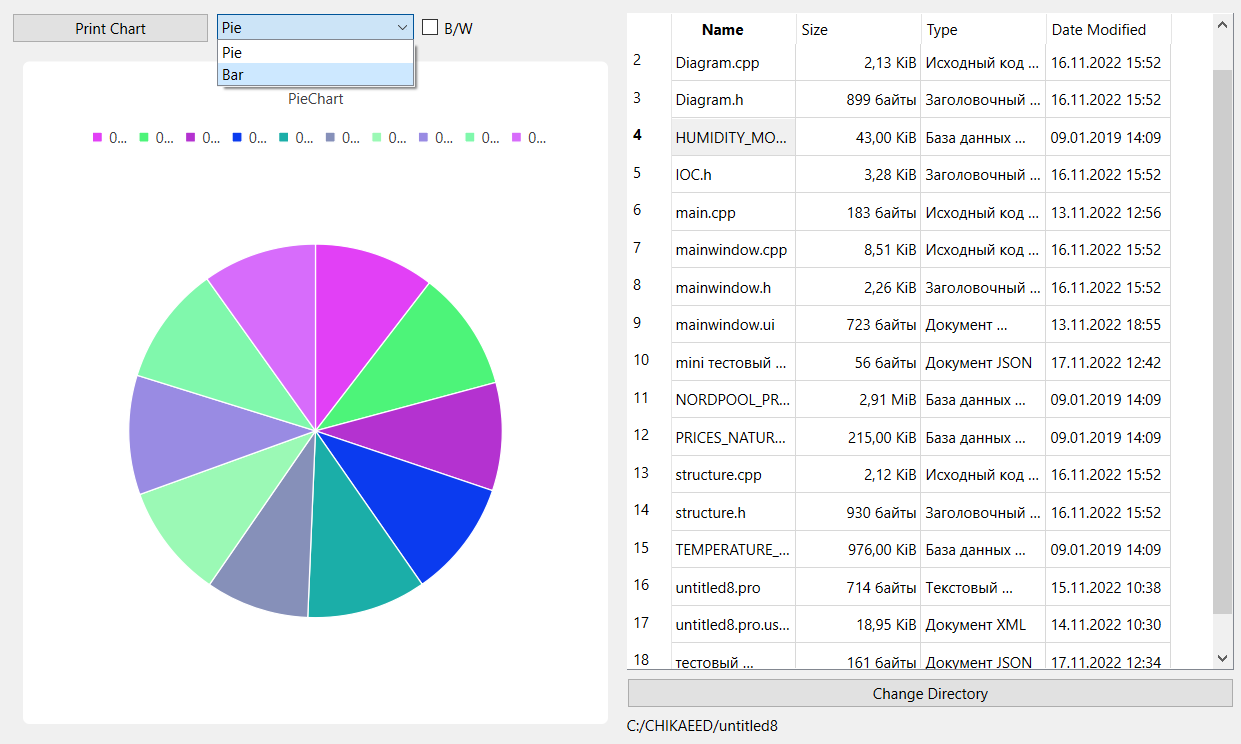
Таким образом, мы с помощью него можем создать объект, выбрать конструктор и возвратить ссылочный тип, получить экземпляры любых зависимостей перед вызовом конструктора.

Для реализации программы использовалась схема MVC. Значит, в программе должны присутствовать:

* Модель (QFileSystemModel \*fileModel; - модель для выбора файла, класс QFileSystemModel предоставляет модель данных для локальной файловой системы)
* Представление(в программе табличное QTableView \*tableFileView;)
* Контроллер

Модификация каждого компонента может осуществляться независимо.

Внешний вид программы:



Для начала выбираем директорию, нажав на кнопку “Change Directory”, всё отображено в табличном представлении с помощью слота **changeDirectory**().

На основе данных из файла на который нажмешь, появляются данные в виде диаграммы. Это сделано с помощью слота, который одновременно учитывает данные, тип, цвет **fileSelection**(const QItemSelection &selected, const QItemSelection &deselected).

***getData*** (QString path\_) обрабатывает и получает данные в нужном виде. Именно эта функция проверяет .sqlite и .json (нужный тип).

Возвращаем данные вида QList<Data>

//структура, в которой хранятся данные

struct **Data** {

public:

QString key;

double value;

};

За выпадающий список с изменением типа графика отвечает слот **changeChartType**();. Тут мы выбираем подходящий нам тип графика, связываем интерфейс с реализацией с помощью функции, строящей график.

Если тип нужно изменить, то в выпадающем списке меняем тип на другой и перерисовываем график с помощью функции **drawChart**();

Нажмем на квадрат checkbox, можно выбрать каким мы график хотим выбрать: черно-белым или цветным. За это отвечает слот **colorSwap**();, наличие самой отметки проверяется в **checkState**().

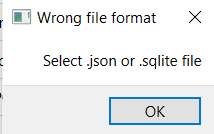
Далее вызывается функция **recreateChart** (QList<Data>, bool isBWEnabled) = 0;, тут дело в истинности аргумента isBWEnabled. Если его значение истинно, то график черно-белый, если ложно, то цветной.

**printChart**();//печать диаграммы – определяем пук к файлу и печатаем в формате .pdf. Отрисовываем график с помощью QPainter.

QPainter painter(*pdfWriter*);

chartView->render(*&painter*);

При выборе из директории в табличном виде не тот файл(то есть тип не .sqlite и не .json), то появляется messangebox с ошибкой



Окно вызывается с помощью

**exceptionCall**(QString title, QString message);//вызов месседжбокса

Для задачи предложена архитектура, поэтому представим ее в виде uml-диаграммы:

1. **Коды программ. Полностью прокомментированы, с включением объяснений решения.**

mainwindow.h

mainwindow.cpp

structure.h

structure.cpp

Diagram.h

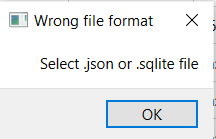
Diagram.cpp

IOC.h

main.cpp

**4. Инструкция пользователя. В этом разделе демонстрируются варианты взаимодействия с вашим ПО с точки зрения пользователя. Если требуется, то предоставляете готовые примеры (входные файлы с исходными данными и т.д.).**

После запуска приложения нам нужно выбрать директорию (кнопка “Choose directory”), открыть папку с файлами .sqlite и .json(у меня в папке файлы типа .sqlite). Если будет выбран файл с другим разрешением, то в окне появится ошибка:

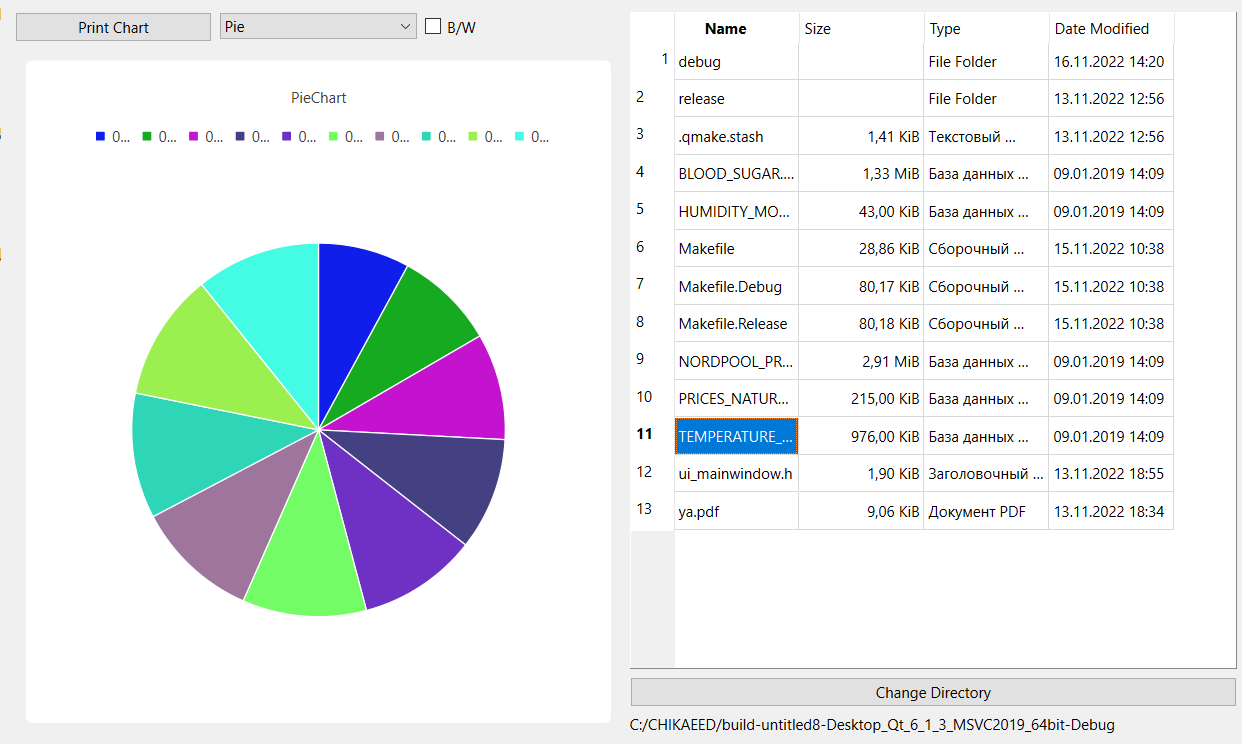


Справа мы выбираем нужный нам файл, а слева отображается график.

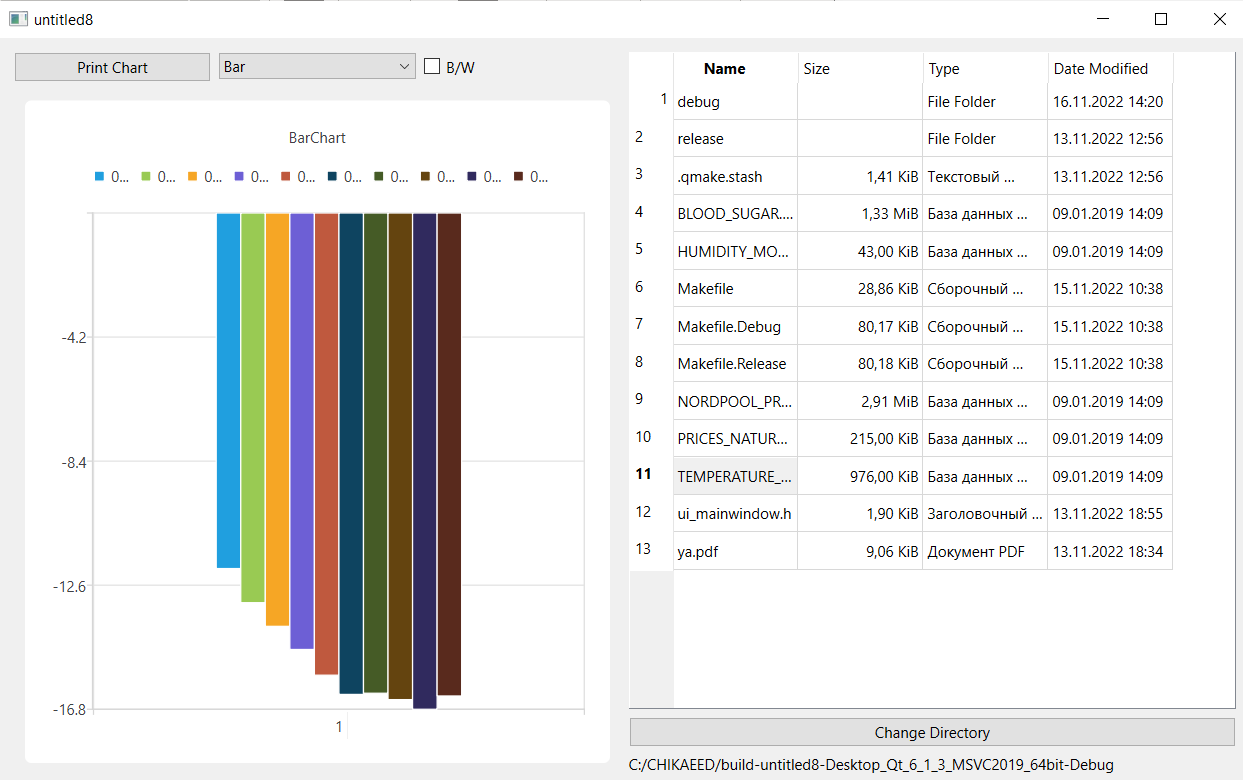
В верхнем левом углу есть кнопки



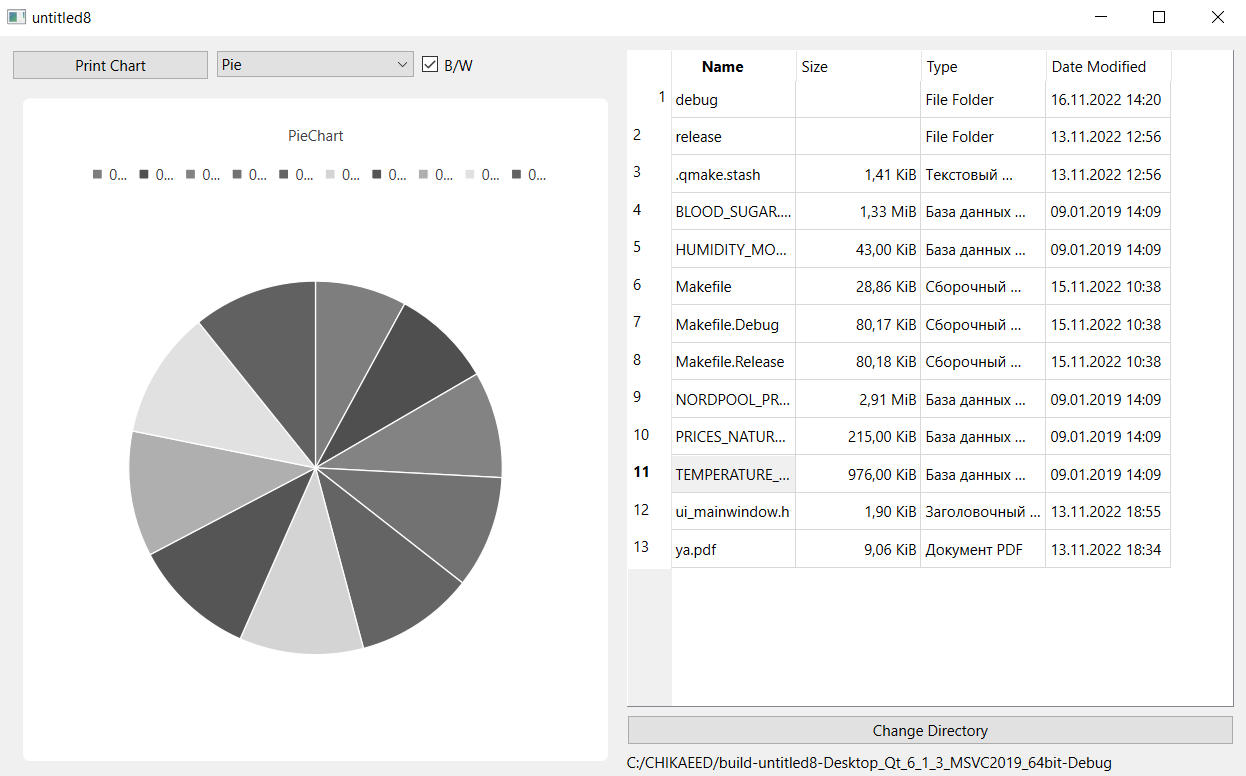
Мы можем выбрать какую диаграмму мы хотим получить. Если хотим круговую диаграмму, то выбираем “Pie”,



если хотим столбчатую диаграмму, то выбираем “Bar”.

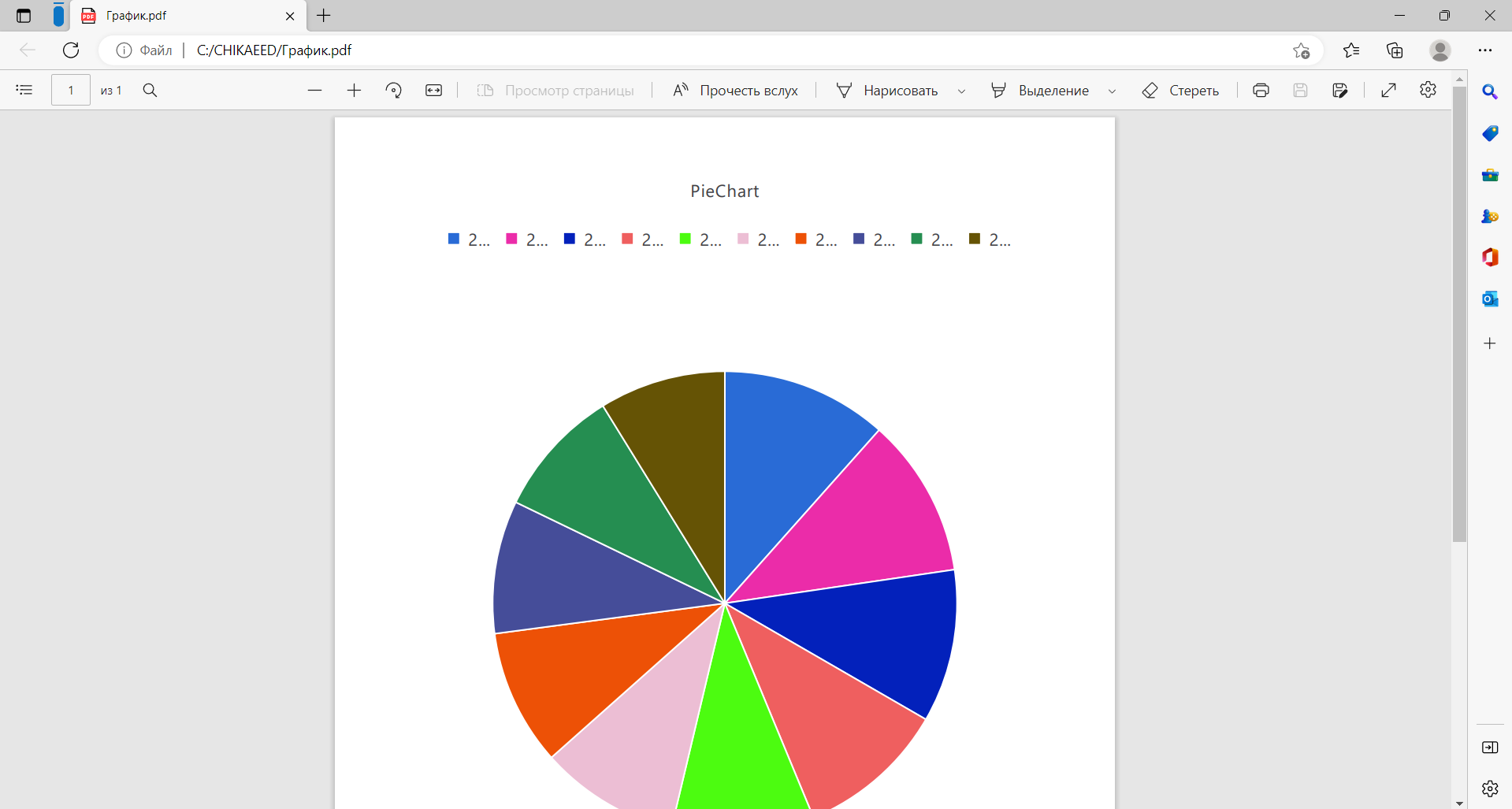


Если поставить галочку рядом с “B/W”, то диаграмма отобразится в разных оттенках серого,

если оставить без галочки, то диаграмма будет разноцветная.

Чтобы напечатать график нужно нажать на кнопку “Print Chart”, далее выбираем путь до файла, где хотим его сохранить.

Таким образом у нас сохраняется файл в .pdf.



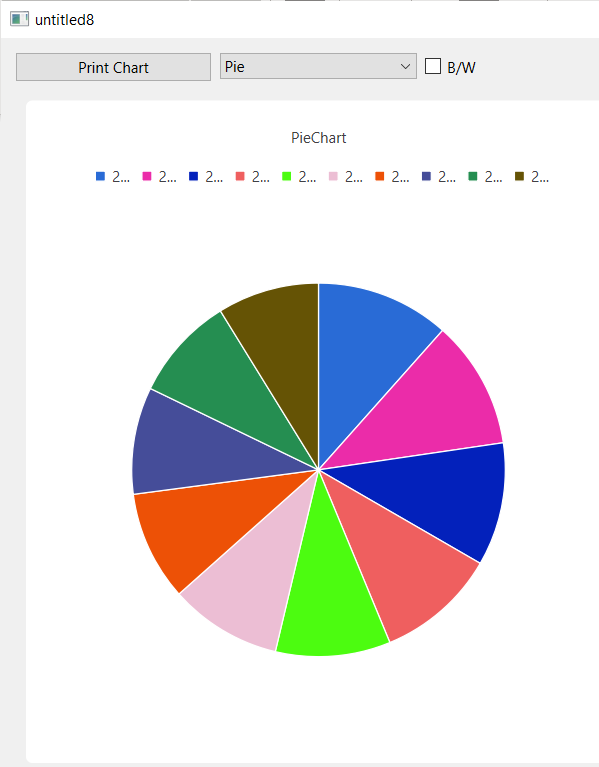
1. **Тестирование. В этом разделе описывается процесс тестирования, в частности все случаи (case1, case2) допустимые и недопустимые при которых ваше ПО выдает адекватную реакцию. Также, тестовые примеры, относительно которых проводилось тестирование.**

Запускаем программу, выбираем директорию. У меня в файле заглавными буквами написаны файлы с разрешением .sqlite.

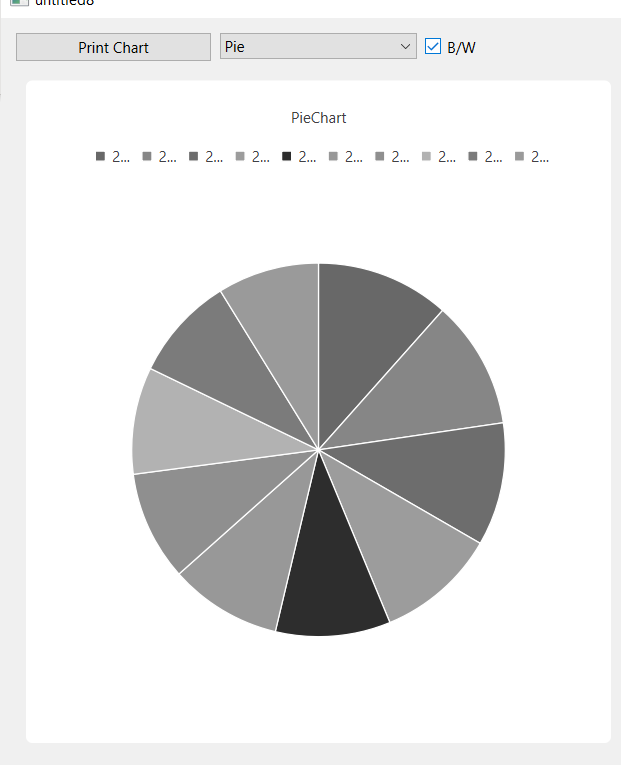
Изображение выглядит как стол

Автоматически созданное описание

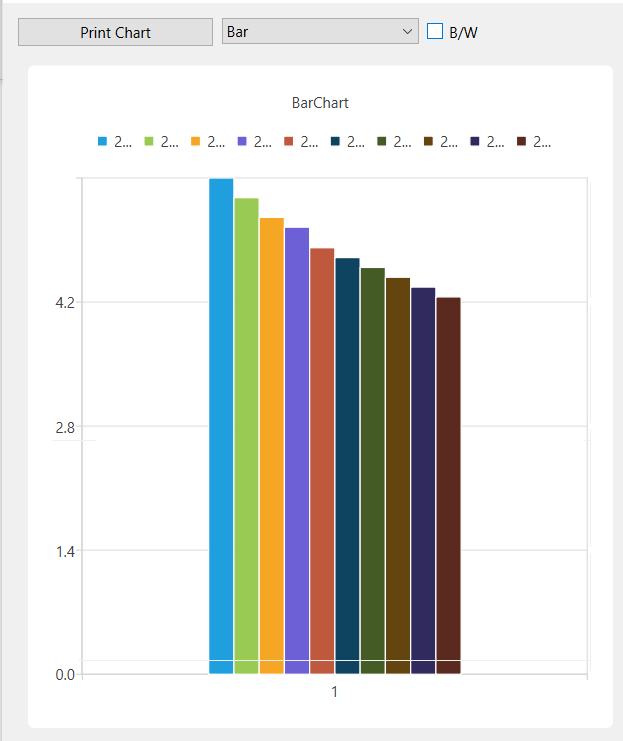
Изначала диаграмма будет цветной

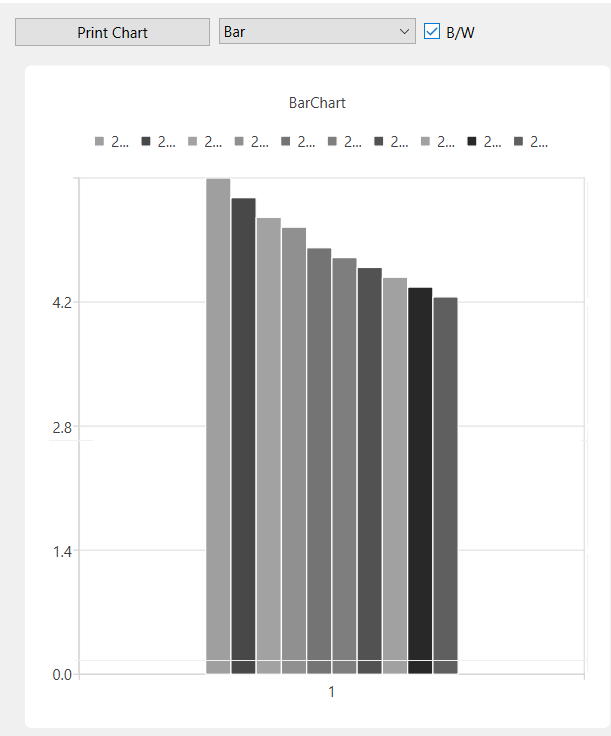


Пока не поставишь галочку около “B/W”



Можно изменить тип графика



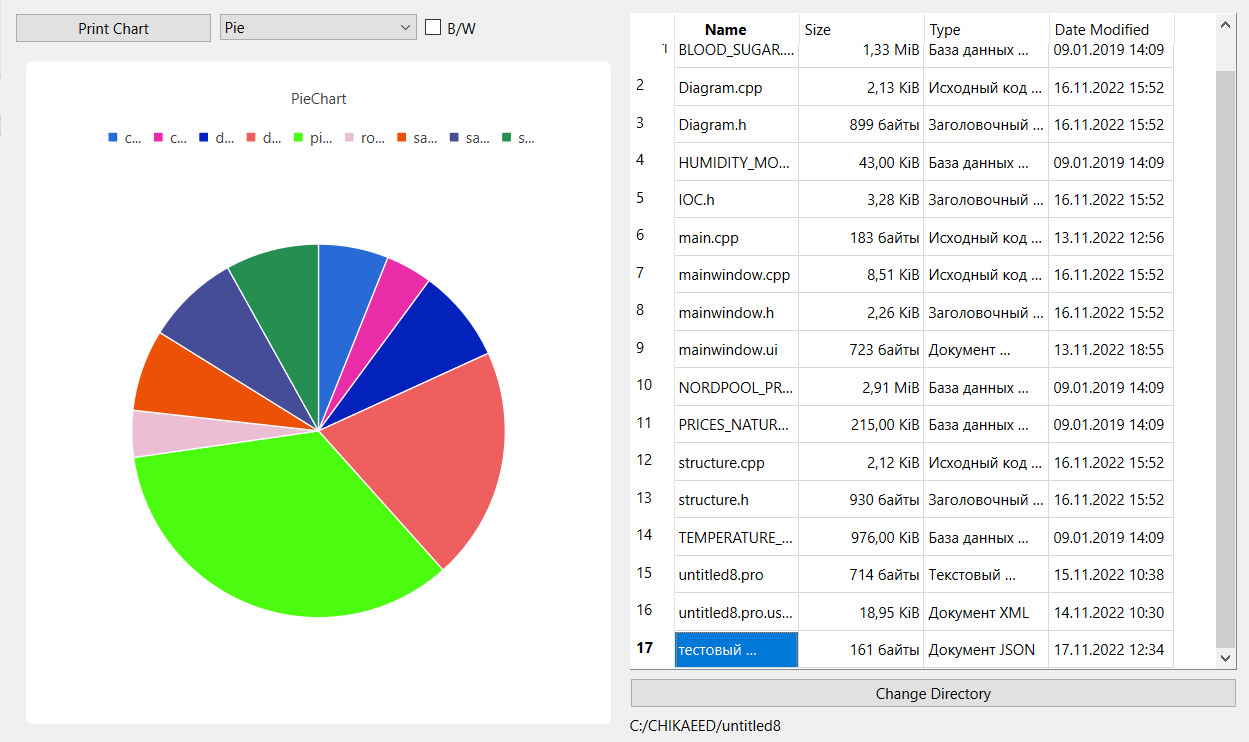


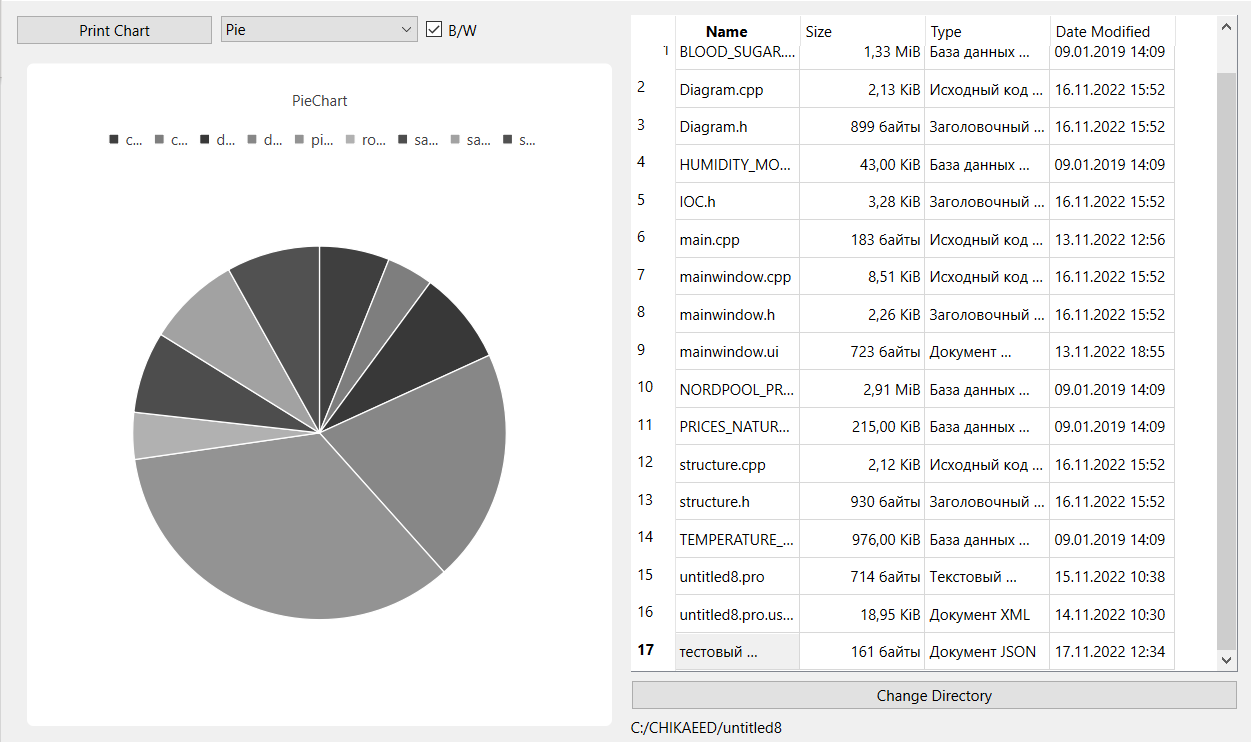
Теперь протестируем .json файл, для начала добавим его в папку.

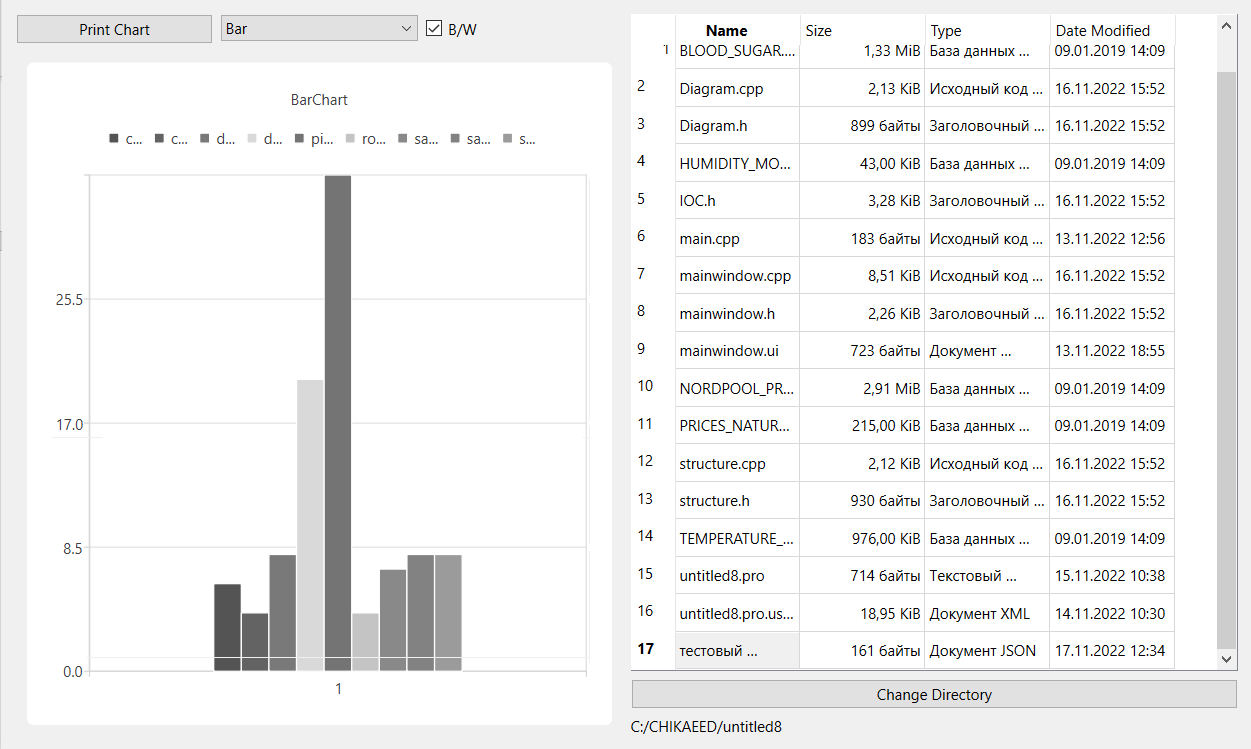
Содержание .json файла:

Изображение выглядит как текст

Автоматически созданное описание



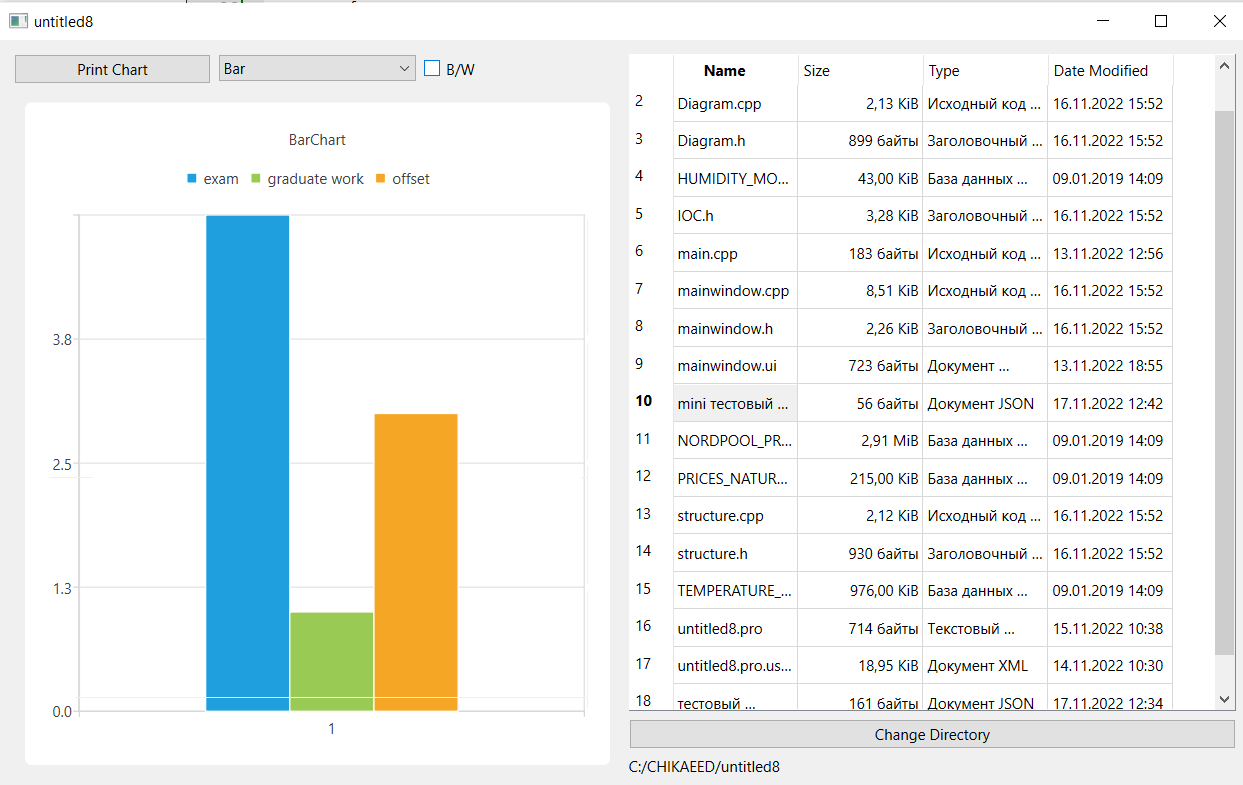


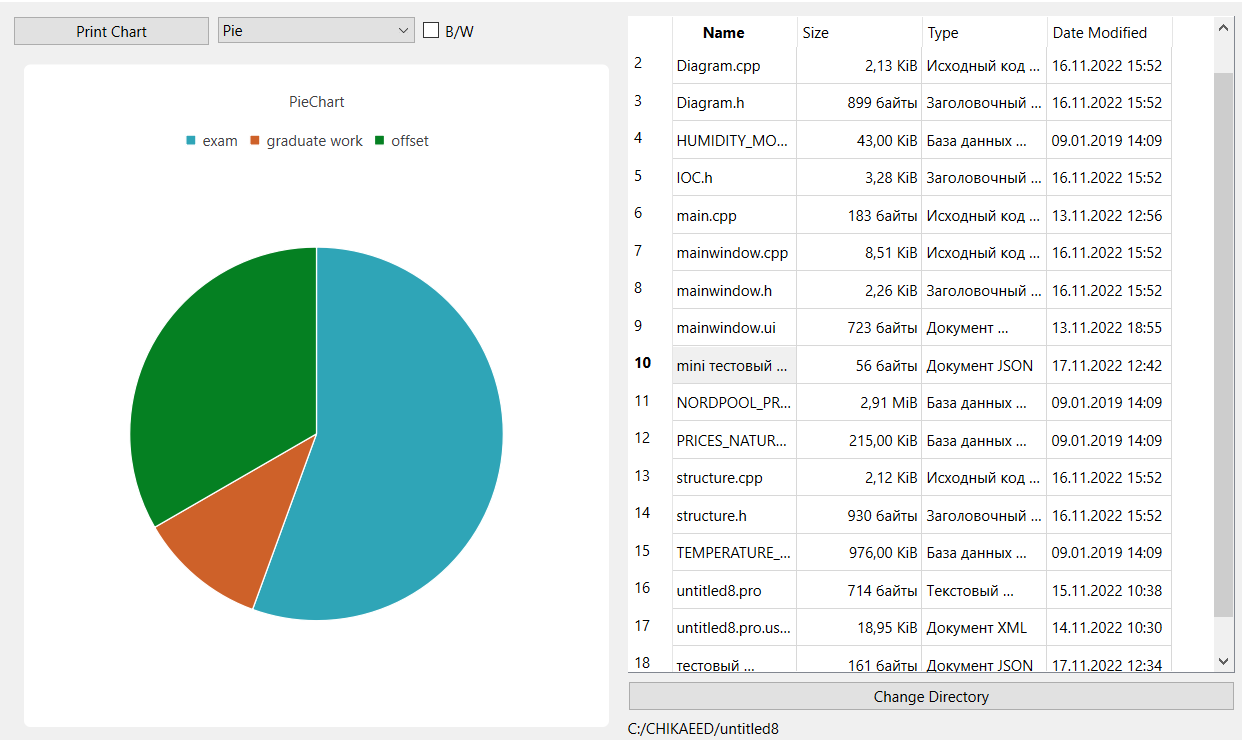


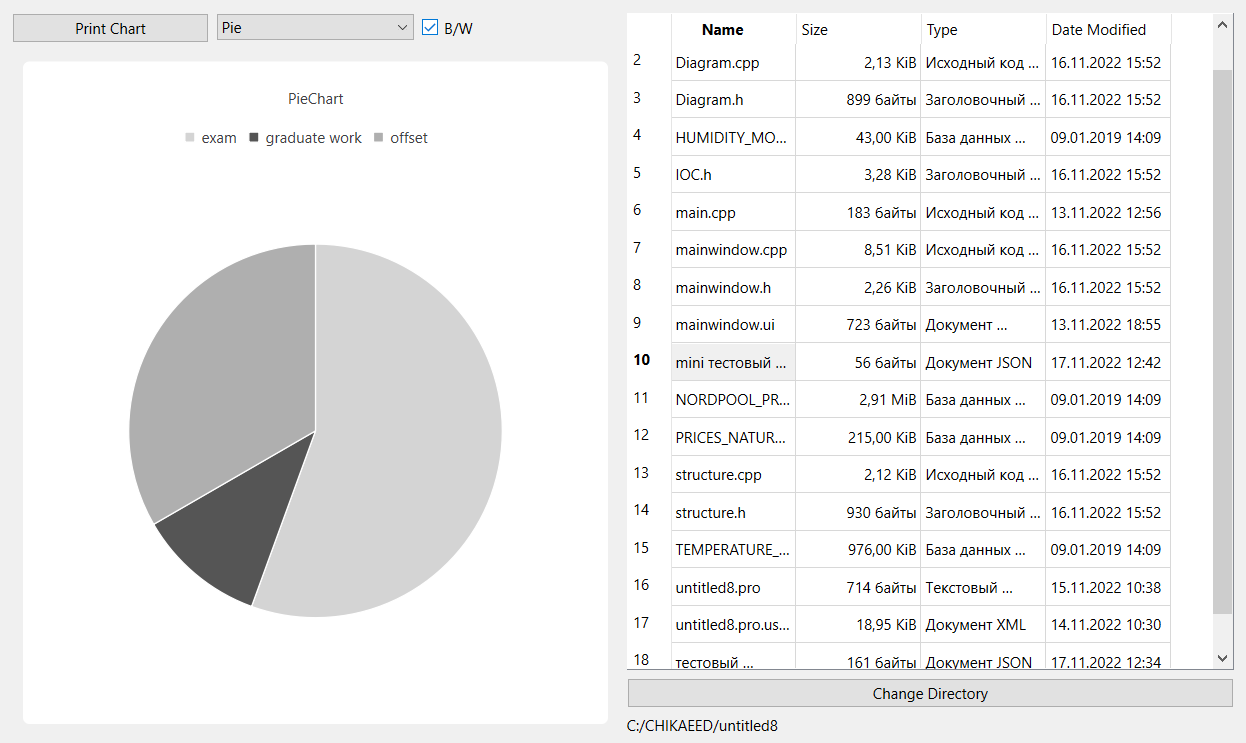
Добавим еще файл с маленьким содержанием данных .json:

Изображение выглядит как текст

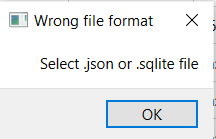
Автоматически созданное описание







Если будет выбран файл с другим разрешением(не .sqlite и .json), то в окне появится ошибка:



Если нажать на кнопку “Print Chart”, то можно выбрать папку, в которой мы хотим сохранить файл в .pdf и он там сохранится

