Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по творческой работе**

Дисциплина: «Информатика»

Тема: Калькулятор и задача коммивояжера

Вариант 15

Выполнила:

студентка группы ИВТ-20-1б

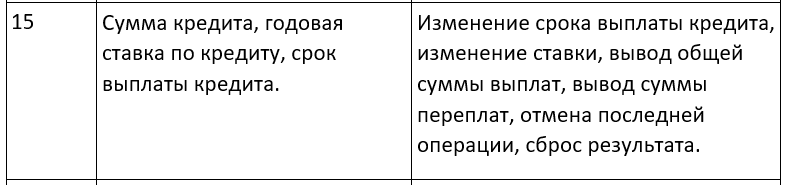
Лыкова Анна Викторовна

**Пермь, 2021Калькулятор**

**Постановка задачи**

1. Разработать алгоритм калькулятора в соответствии с заданием.
2. Реализовать алгоритм в виде программы на алгоритмическом языке C++.
3. Разработать интерфейс средствами Qt.

Задание**:**



**Анализ задачи**

Беря кредит в банке, необходимо учитывать, что помимо основного долга (тела кредита), придётся выплатить проценты. Величина суммы процентов, а это главная часть переплаты, зависит от того, какой способ погашения кредита выбран – аннуитетный или дифференцированный.

При аннуитетном платеже задолженность погашается равными платежами на протяжении всего срока кредита. В первую очередь уплачиваются проценты: каждый месяц они считаются от оставшегося долга по кредиту. Оставшаяся после уплаты процентов часть фиксированного платежа направляется на погашение основного долга. Соответственно, в следующем месяце остаток долга становится чуть-чуть меньше, на него начисляется меньше процентов, а на погашение основного платежа идет чуть большая часть фиксированного платежа.

Сегодня большее место на рынке отведено аннуитетному формату платежей, поэтому калькулятор создавался на его основе.

Для подсчета суммы ежемесячного платежа по кредиту нужны следующие данные:

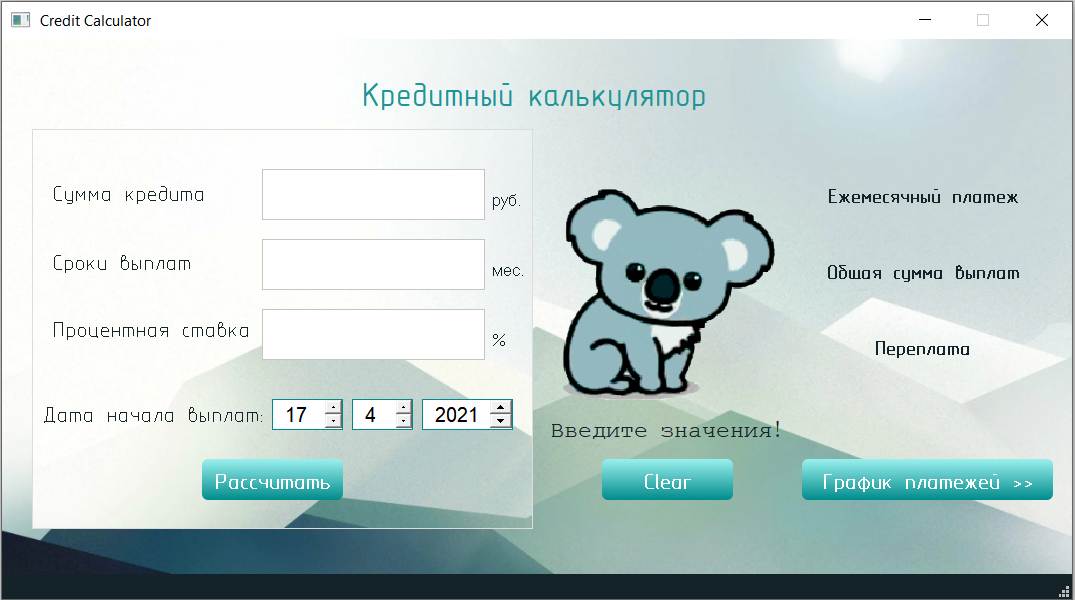
* общая сумма кредита;
* процентная ставка;
* период кредитования.

Эти данные были представлены как основные переменные:

double sum;

double year\_percent;

int terms;

Реализация калькулятора была начата с создания интерфейса, который состоит из полей ввода и вывода необходимых по заданию значений, а также 3 кнопок: для расчёта по кредиту, очистки всех полей и показа таблицы с графиком всех платежей. Значения вводятся с клавиатуры, допускается ввод только положительных чисел. Длина вводимых значений также ограничена.

**Функционал калькулятора**

Как производится расчёт по кредиту

*Основной платеж = Сумма кредита / Кол-во мес.*

Годовую ставку для удобства расчётов представим в виде десятичной дроби и найдём месячный процент:

*Месячная проц. ставка = Годовая проц. ставка/ 12*

Ежемесячный платеж зависит только от тела кредита и коэффициента аннуитета:

*Ежемесячный платеж* = Сумма кредита \* Коэф.ан

Формула расчёта коэффициента аннуитета такова:

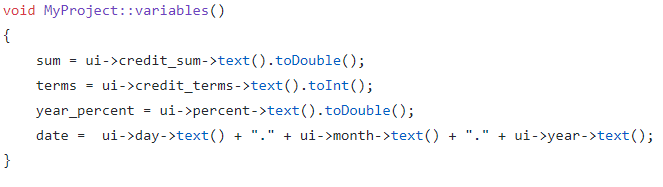


Поскольку месячные платежи одинаковы, а количество месяцев известно, то для вычисления общей суммы выплат остаётся только перемножить эти величины:

*Общ. сумма = Ежемесячный платеж \* Кол-во. мес.*

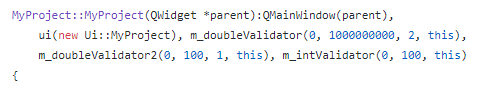
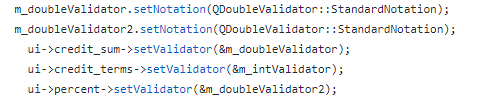
Переплата в чистом виде есть сумма процентов, без учёта комиссии, страховки и проч. Если они есть, то их придётся приплюсовать и к общей сумме, и к переплате.

*Переплата = Общ. сумма – Сумма кредита*

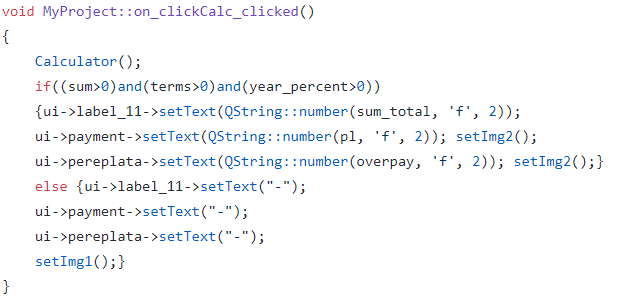
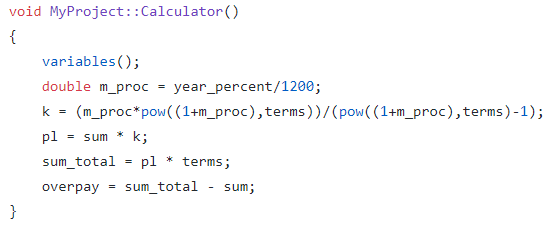
В коде для присвоения переменным данных из полей ввода предназначена функция variables().

Ограничение для вводимых пользователем данных было реализовано с помощью валидатора. Для этого в заголовочном файле программы подключается библиотека QValidator и в класс добавляются переменные типа QIntValidator и QDoubleValidator (для целых и дробных чисел соответственно).

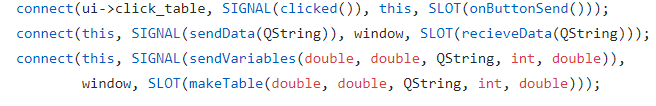
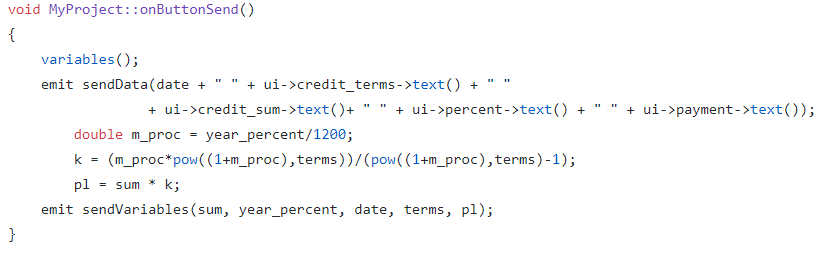
Для того, чтобы ограничитель работал в конструкторе класса необходимо добавить следующее:

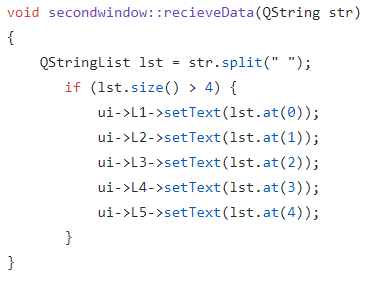


Для реализации расчета по кредиту была создана функция Calculator().

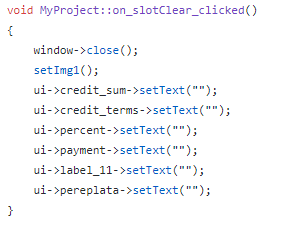
А для вывода результата вычислений – функция-слот on\_clickCalc\_clicked(), которая вызывается при нажатии на кнопку clickCalc.

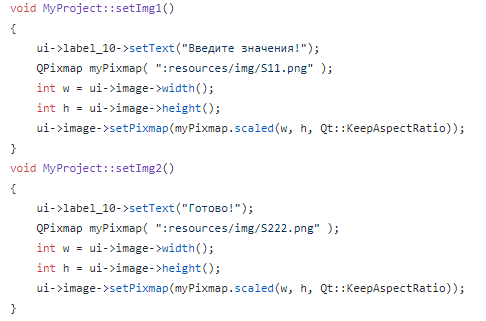
Однако, это не весь функционал кредитного калькулятора. Инструмент покажет наглядный график внесения средств, который поможет проанализировать сведения об имеющихся задолженностях. То есть об остатке за определённый промежуток времени, сумме платежа по долгу и процентной ставки банка. Для этого необходимо просто нажать на кнопку «График платежей», тем самым вызвав функцию-слот on\_click\_Table\_clicked(), которая открывает дополнительное окно с основной информацией по кредиту и таблицей при условии, что введенные пользователем значения не равны 0. Также кнопка click\_Table вызывает сигналы sendData и sendVariables в функции , которые передают необходимые значения из главного окна во второе в слоты receiveData() для вывода общей информации об кредите и makeTable() для создания таблицы с графиком платежей.







При ошибке ввода пользователь может сбросить результат, удалив все значения и очистив поля ввода и вывода. Для этого необходимо нажать на кнопку «clear».

Вставка и смена картинки коалы, показывающая текущее состояние калькулятора, выполняется функциями setImg1() setImg2().

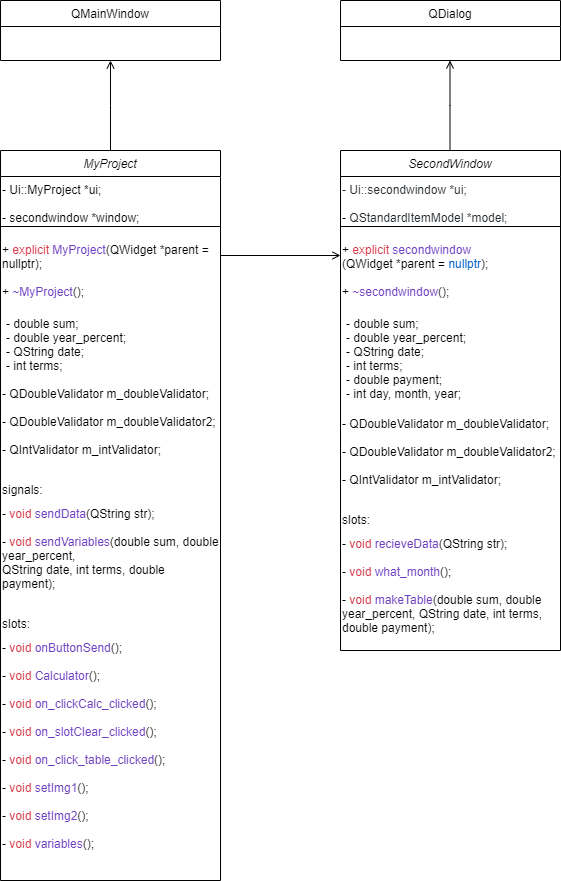
Дизайн калькулятора создавался с помощью таблицы стилей на форме Qt Designer. Внешний вид калькулятора разработан в спокойных тонах, не напрягающих глаза пользователя. Размер окна приложения фиксирован и изменяться не может. Поля ввода при наведении слабо выделяются, а при нажатии подсвечиваются более ярким цветом, для привлечения внимания пользователя.

## **Преимущества**

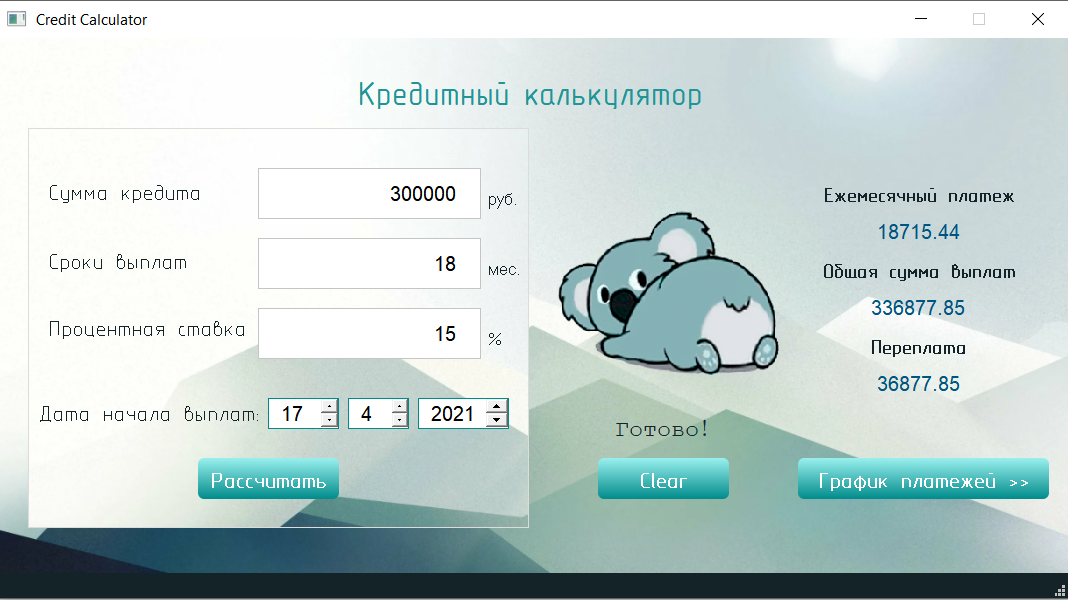
Основными преимуществами кредитного калькулятора являются его простота и удобство использования, все что нужно – это вбить сумму кредита, которую хотите взять и срок кредитования (срок указывается в месяцах). После этого система рассчитает все необходимые параметры.

Данный сервис отлично подойдет тем, кто хочет узнать свои финансовые возможности и предложения банков в целом.

Инструмент отлично подойдет для быстрого получения информации без помощи банковских специалистов. Его удобно применять для сравнения различных результатов с целью выбора более оптимальных вариантов для оформления.

**UML – диаграмма**

**Результат работы программы**





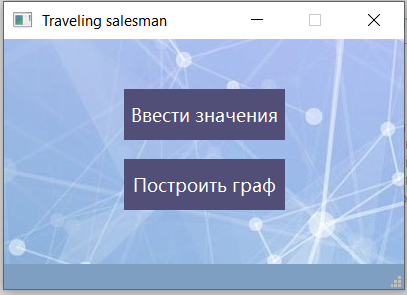
**Задача коммивояжера**

**Постановка задачи**

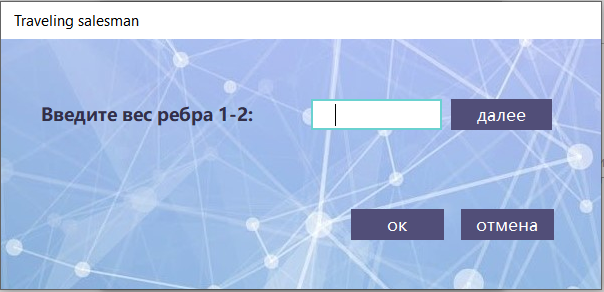
1. Разработать алгоритм решения задачи коммивояжера для графа.
2. Реализовать алгоритм в виде программы на алгоритмическом языке C++.
3. Разработать интерфейс средствами Qt.

**Анализ задачи**

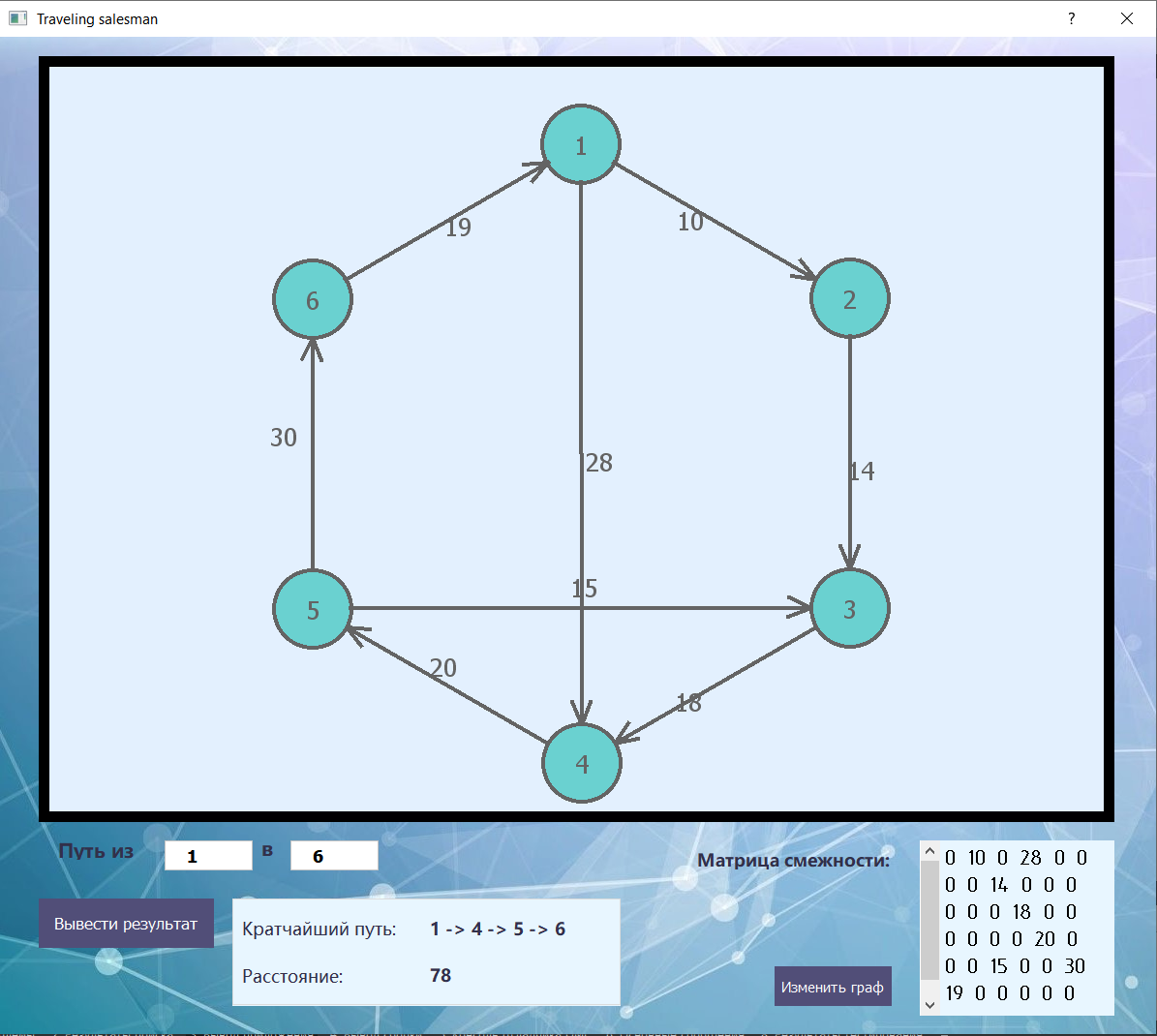
Рассмотрение выполненного задания начнем с интерфейса.

При запуске программы открывается небольшое основное окно, предоставляющее пользователю на выбор 2 кнопки: ввести значения или построить граф по уже имеющимся значениям. Значения берутся из дополнительного текстового файла. При наведении курсора на кнопку, она меняет цвет.

При нажатии на первую кнопку открывается новое окно. В поле пользователю предлагается ввести вес каждого ребра графа. Каждый раз после ввода значения необходимо нажимать кнопку «далее». Для удобства пользователя вместо кнопки на экране, можно нажимать Enter. По окончанию ввода появится сообщение «Все значения введены!» и поле с кнопкой станут неактивными. На окне также имеются кнопки «ok» и «отмена». Первая становится активна после появления сообщения. Поскольку значения для построения графа берутся из файла, они должны туда сохраняться. Поэтому кнопка ok перезаписывает файл, сохраняя в него новые значения. Кнопка отмены активна всегда. Нажав на нее, пользователь отменяет ввод и значения в файл не изменяются.



Для того, чтобы построить граф по введенным значениям, необходимо нажать на следующую кнопку главного окна «Построить граф». Также откроется дополнительное окно с построенным графом. Справа, ниже графа находится поле с его матрицей смежности, с помощью этого поля можно изменять значения и перестраивать граф. Кроме того, пользователь может найти кратчайший путь из одного пункта в другой. Для этого он должен ввести значения в соответствующие поля и нажать кнопку «вывести результат». В случае, когда найти путь невозможно, т.е. граф не является связным, появляется сообщение с предупреждением о том, что маршрута не существует.



В заголовочном файле главного окна mainwindow определены 2 функции-слоты.

on\_createGraph\_clicked() отвечает за открытие окна с графом. Функция on\_input\_btn\_clicked() открывает окно для ввода значений, а также создает новый файл, посредством копирования старого c помощью функции QFile::copy. Если старый файл не существует, появляется сообщение с предупреждением с помощью всплывающего окна QMessageBox.



Для ввода веса каждого ребра и записи в файл предназначена функция-слот onButtonSend(), которая вызывается при нажатии на кнопку onward\_btn().

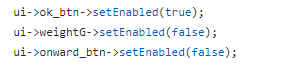




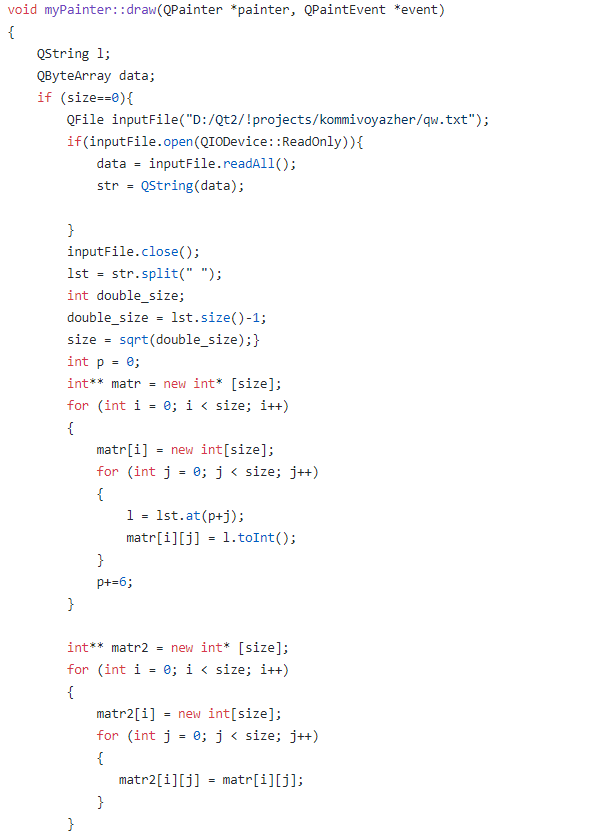
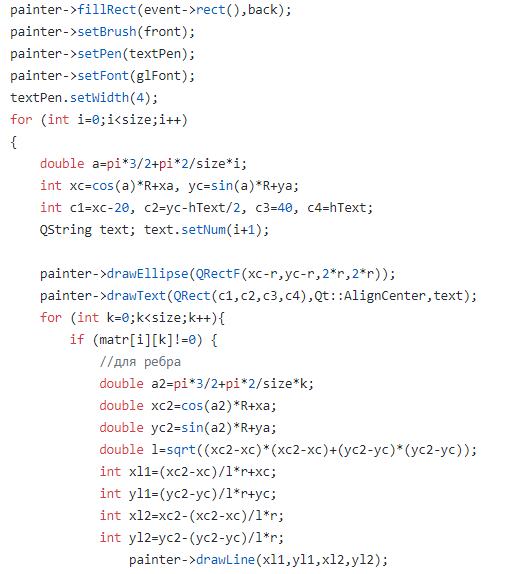
Функции-слоты on\_cancel\_clicked() и on\_ok\_btn\_clicked(), вызывающиеся нажатием кнопки «cancel» и «ok» соответственно, вызывают функцию btnPress() и закрывают текущее окно. Функция on\_ok\_btn\_clicked() помимо этого также возвращает из дополнительного текствого файла данные в основной. Функция btnPress() делает виджеты вновь активными и стирает все введенные ранее в поля ввода значения.

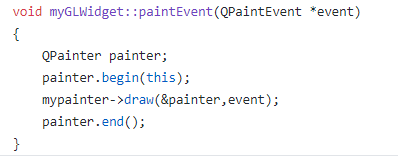


Чтобы сделать виджеты неактивными или активными использовалась встроенная функция setEnabled.

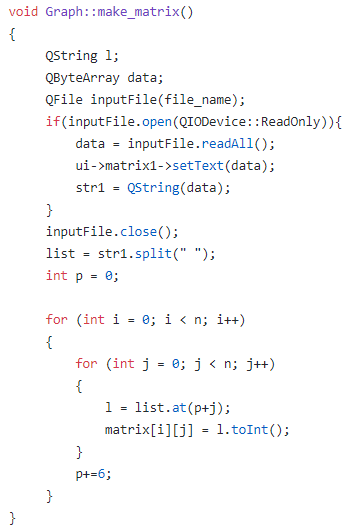


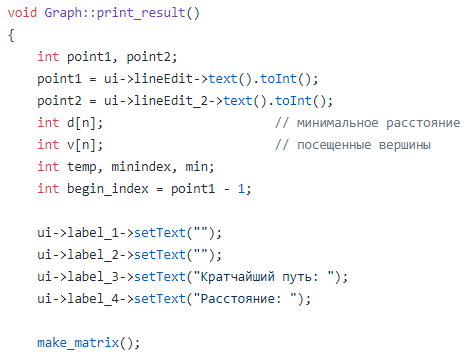
За отрисовку узлов, ребер и текста графа отвечает функция draw в файле mypaint. Граф рисуется с помощью средств openGL. Для этого были подключены библиотеки <QPainter> и <QOpenGLWidget>.

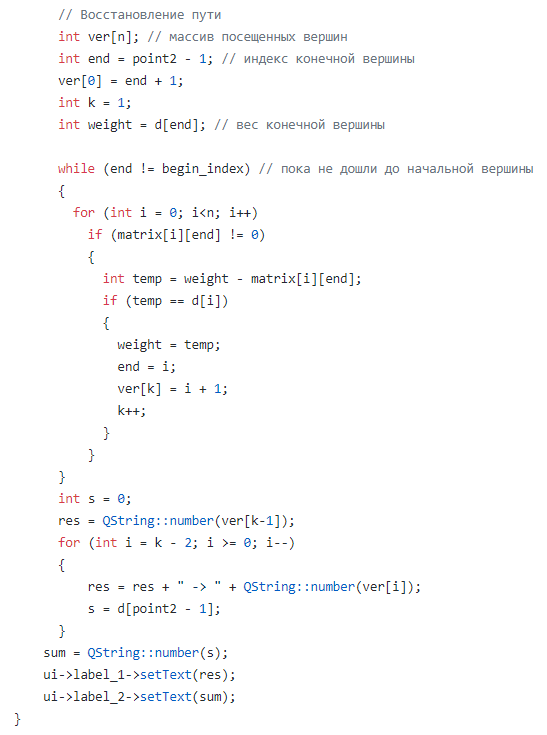




Для решения задачи Коммивояжера был использован алгоритм Дейкстры — [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) на [графах](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), изобретённый нидерландским учёным [Эдсгером Дейкстрой](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5). Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для связных графов без [рёбер](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B1%D1%80%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) отрицательного [веса](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#%D0%92). Для реализации этого алгоритма предусмотрено 2 функции: makeMatrix() для создания матрицы смежности посредством чтения данных из текстового файла и print\_result() для нахождения кратчайшего пути и расстояния и вывода результата.





**UML – диаграмма**

