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 творческому заданию**

Дисциплина: «Информатика»

Тема: Создание калькулятора и задача Коммивояжера средствами Qt

Вариант 17

Выполнил:

Студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Постановка задачи**

1. **Задача Коммивояжера**
2. В качестве варианта для демонстрации работы программы взять свой вариант задания из лабораторной работы по графам. Модифицировать граф таким образом, чтобы для этого графа можно было решить задачу Коммивояжера. Можно придумать собственную альтернативную задачу, которую можно решить методом ветвей и границ. Разработать программу, которая будет универсальной на любом наборе исходных данных.
3. Проработать визуализирующую часть, а именно - построение графа, средствами Qt.
4. Исходные данные должны приниматься с консоли и через графический интерфейс.
5. Задокументировать программу диаграммой классов UML.
6. **Калькулятор «Хранение капиталов в банке»**
7. Разработать алгоритм калькулятора в соответствии с заданием.
8. Реализовать алгоритм в виде программы на алгоритмическом языке C++.
9. Разработать интерфейс средствами Qt.

**Задание варианта:**

Банк предлагает своим клиентам условия:

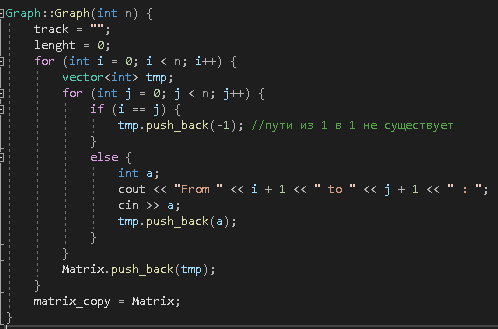
1. Положить деньги на год под p-процентов годовых с начислением процентов по окончанию года
2. Положить деньги на год под p1-процентов с начислением процентов ежемесячно
3. Положить деньги на год с ежедневным начислением p2-рублей на каждую полную тысячу вклада
4. Положить деньги на год с ежедневным начислением p3-процентов на сумму вклада

**Примечание:** Банк может изменять уговоренные проценты. Калькулятор получает на вход величину вклада в рублях и копейках или в долларах и центах, а также величину срока в месяцах или годах и вариант условий размещения капитала с вычислением величины вклада к концу указанного срока. Калькулятор должен иметь окна для исходных данных и результатов, для ввода данных, операций и служебных клавиш, клавиши изменения знака числа и вида валюты. Должен позволять корректировать последнее введённое число и выдавать сообщения об ошибках.

**I. Задача коммивояжера**

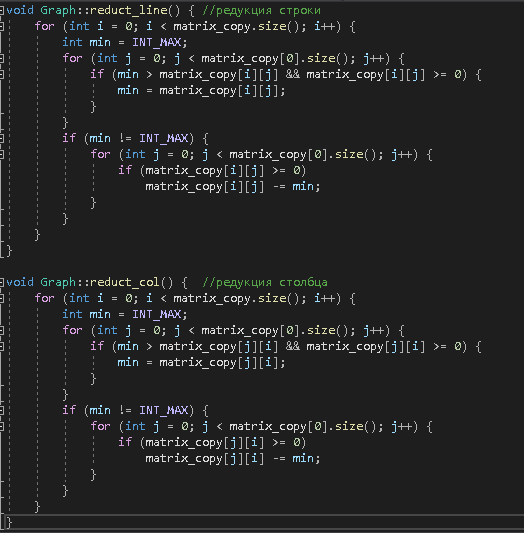
Задача коммивояжера заключается в нахождении оптимального маршрута, проходящего через все узлы не менее одного раза и возвращающегося в начало. Совершенно очевидно, что задача может быть решена перебором всех вариантов объезда и выбором оптимального. Беда в том, что количество возможных маршрутов очень быстро возрастает с ростом n (оно равно n! — количеству способов упорядочения пунктов). Поэтому для решения задачи были придуманы различные методы, которые хорошо справляются с ней. Однако может пострадать точность. К ним относятся метод градиентного спуска, имитации отжига и метод ветвей и границ. Я решил воспользоваться последним.

1. Алгоритм данного метода реализован с помощью класса Graph, содержащего матрицу смежности vector<vector<int>> Matrix и методы, позволяющие вычислить оптимальный путь и его длину.
2. Ввод матрицы смежности осуществляется в конструкторе класса.



1. Приведение матрицы.

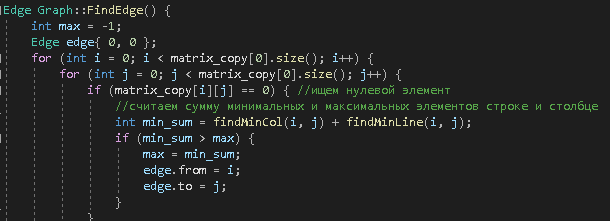
Приведенная матрица получается путем редукции строк и столбцов (поиск минимального значения в строке/столбце и вычитание его из каждого элемента строки/столбца). С помощью вложенного цикла проходим по строке/столбцу матрицы и ищем минимальный элемент, не включая -1, затем вычитаем его из каждого элемента.



В результате выполнения редукции строк и столбцов получаем приведенную матрицу.

1. Поиск границы.

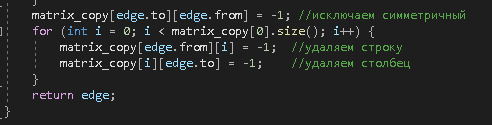
Поиск границы осуществляется оценкой нулей в функции Edge FindEdge (). Для оценки нулей ищутся минимальные элементы в строке и в столбце, в котором находится оцениваемый 0 и после этого складываются. Результатом сложения и будет оценка 0.



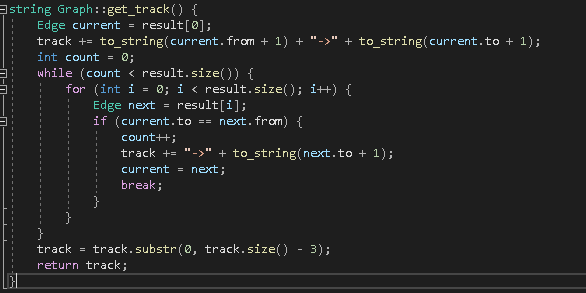
После оценивания нулей выбираем 0 с самой большой оценкой и запоминаем его координаты – это и будет ребро, по которому Коммивояжер должен будет пройти. Координаты хранятся в структуре Edge с полями from (откуда) и to (куда).



Удаляем строку и столбец, в котором находится этот 0, а также симметричный ему элемент, устанавливая им значение -1.



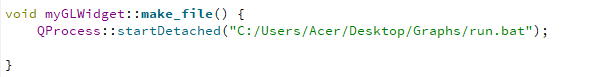
1. В функции void make\_track () объединяем шаги 3-4 и повторяем в цикле, до тех пор, пока не найдем полный путь, сохраняя найденные границы в vector <Edge> result.
2. Полученный вектор с границами мы будем использовать в функции string get\_track () для связывания городов и получения пути решения в виде строки. Для этого мы берём каждую границу и связываем её со следующей, пока не дойдем до границы, с которой начали связывание.



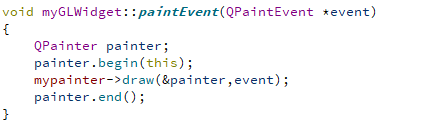
1. Длина пути рассчитывается по аналогии с нахождением пути в строке, только здесь мы используем матрицу смежности и считаем расстояние от узла к узлу.
2. Последняя функция void make\_files () создает файлы с матрицей и высчитанным наикратчайшим путем, и его длиной. Данные файлы будут считаны при нажатии кнопки “Input Matrix” в окне приложения и визуализированы.
3. Визуализация выполнена в Qt. Окно приложения содержит кнопку ввода матрицы с консоли “Input Matrix”, кнопку печати “Print Matrix” и поле отрисовки графа.



1. При нажатии “Input Matrix” открывается консоль, где необходимо ввести матрицу. Формируются файлы с матрицей и наикратчайшим путем. Это реализовано следующими строчками кода:

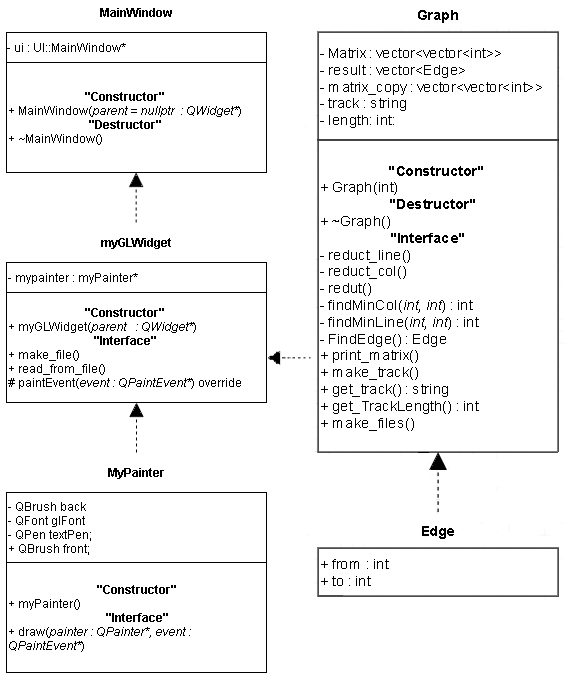


1. При нажатии “Print Matrix” вызывается функция void myGLWidget::read\_from\_file(), в которой данные файлов записываются в глобальные переменные, необходимые для построения графа.
2. Рисование на виджете myGLWidget реализовано в функции void myGLWidget:: paintEvent(QPaintEvent \*event) , в которой создается объект класса QPainter.

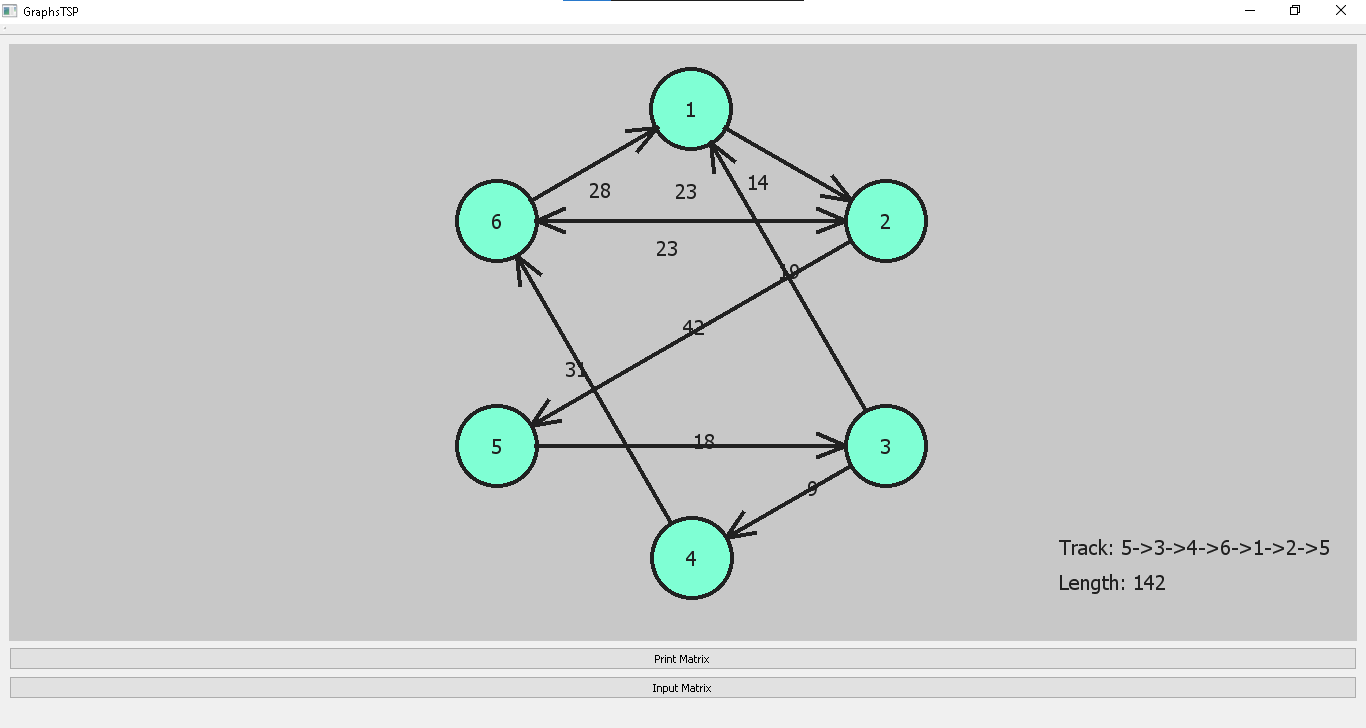


1. Отрисовка происходит в функции void myPainter::draw(QPainter \*painter, QPaintEvent \*event).

**UML диаграмма задачи Коммивояжера**

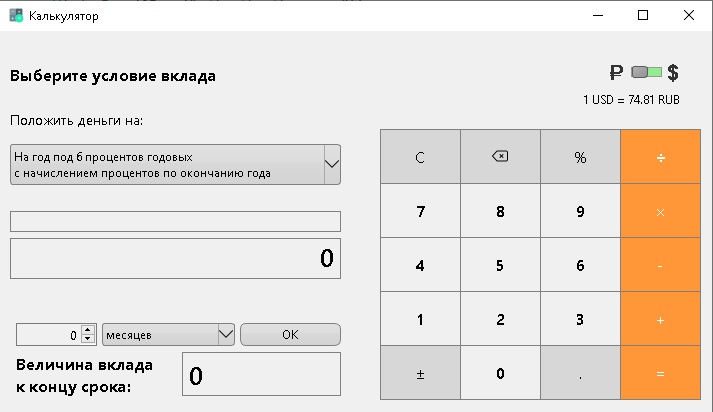


**Результаты работы**

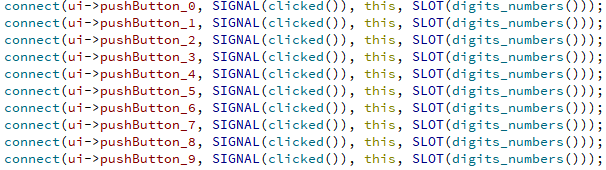


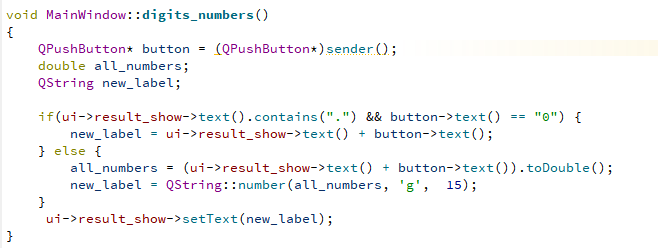
**II. Калькулятор «Хранение капиталов в банке»**

1. На поле калькулятора находится выпадающий список с условиями вклада поле ввода суммы, кнопки цифр и операций, поле ввода срока вклада в месяцах/годах, величина вклада к концу указанного срока (вычисляется при нажатии кнопки «Ок») и ползунок смены валюты рубли-доллары.



1. Окно приложения описано в классе MainWindow, которое унаследовано от QMainWIndow.
2. Кнопки цифр при нажатии вызывают функцию void MainWindow:: digits\_numbers(), которая отображает в поле ввода данную цифру.

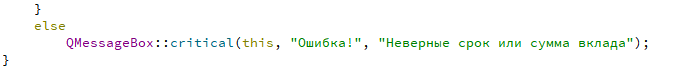




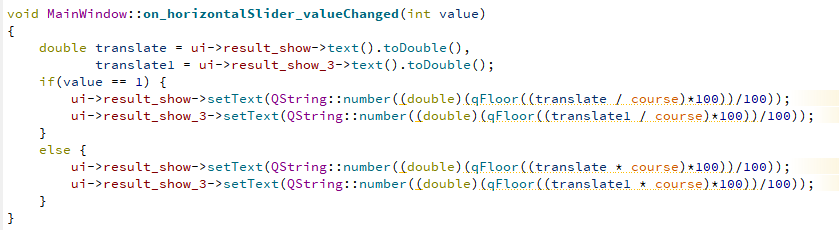
1. При нажатии кнопки «Ок» вызывается функция void MainWindow::on\_ok\_clicked(), которая считывает введенную сумму вклада, срок и условие. Далее в условном операторе switch в зависимости от выбранного условия высчитывает сумму вклада к концу срока в цикле.



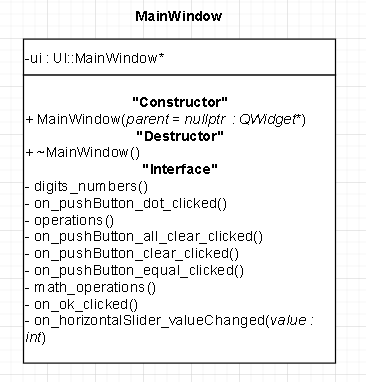
1. Здесь же выводится сообщение об ошибке, если сумма или срок вклада неверные.



1. В функции void MainWindow::on\_horizontalSlider\_valueChanged(int value) происходит перевод из рублей в доллары или наоборот.



**UML диаграмма калькулятора**



**Результаты выполнения**

