Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по творческой работе**

Тема: «Разработка калькулятора и решение задачи коммивояжера»

Вариант 16

Выполнил:

студент группы РИС-20-2б

Исмагилов Айназ Рамзитович

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Введение**

С помощью средств фреймворка Qt и её среды разработки Qt creator были реализованы два приложения: калькулятор, работающий в градусной системе мер, и приложение, визуализирующее графы и решающее задачу коммивояжера.

**1. Задача по разработке калькулятора.**

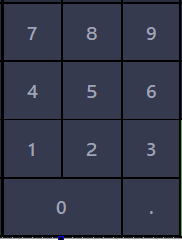
**1.1. Постановка задачи**

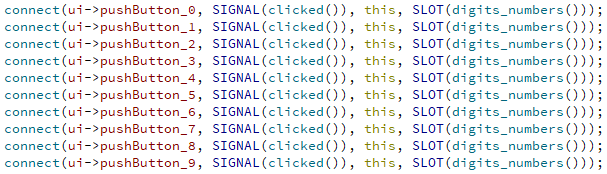
1. Разработать алгоритм калькулятора в соответствии с заданием.
2. Реализовать алгоритм в виде программы на алгоритмическом языке С++.
3. Разработать интерфейс средствами Qt.
4. Использование анимации, автоматизации, креативной визуализации повышает рейтинг автора проекта и влияет на экзаменационную оценку.

|  |  |  |
| --- | --- | --- |
| 16 | Градусная система мер. | Калькулятор работает с углами, измеренными в градусной системе мер. Например: 1 градус = 60 минут, 1 минута = 60 секунд.  Панель калькулятора должна иметь окна для отображения измеренных углов и результатов операций с углами: сложение, вычитание, умножение, деление, нахождения значений тригонометрических и обратных тригонометрических функций, изменения знака угла, а также корректировать последние введённые значения, выдавать сообщения об ошибках. |

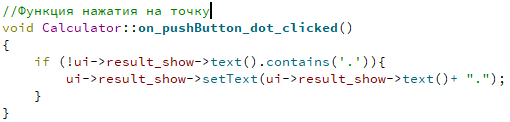
**1.2. Анализ задачи**

1. Для начала, необходимо реализовать ввод и вывод числовых значений.
   1. Для ввода числовых значений создаются кнопки для цифр от 0 до 9



* 1. Для вывода введённых значений создаётся объект Qlabel.
  2. Для кнопок с цифрами задаётся сигнал clicked(), который активирует функцию digits\_numbers().
  3. В функции digits\_numbers() в объект Qlabel вставляется текст нажатой кнопки. Также предусмотрены проверки некоторых условий: содержания в выводящей строке точки, содержание знаков угла, секунд и минут, и проверка на наличие ответа. Все данные условия пригодятся при дальнейшей работе.

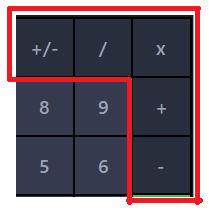


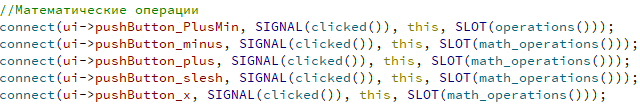
* 1. Теперь необходимо создать функция on\_pushButton\_dot\_clicked() для кнопки dot, для того чтобы пользователь мог вводить не целочисленные значения. Кнопка dot также соединяется с сигналом со своей функцией для её вызова по нажатию. 

В функции предусмотрено условие, чтобы пользователь не мог вводить сразу несколько точек.

* 1. Также необходимо реализовать кнопку для изменения знака введённого значения. Для этого создаётся кнопка plus/minus и функция operations()



* 1. После того были добавлена возможность вводить числовые значения, теперь необходимо реализовать математические операции для работы с числами. Создаются кнопки для математических операций. 

Созданные кнопки привязываются с функцией math\_operations()

Сама функция реализована таким образом:

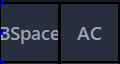
Каждая кнопка математический операция имеет функцию setChecked для проверки, какая кнопка нажата. При нажатии кнопки, ранее введенное числовое значений присваивается глобальной переменной first\_num, и весть текст с первой выводящей строки переходит на вторую строку, а первая строка очищается для дальнейшего ввода уже другого числового значения.

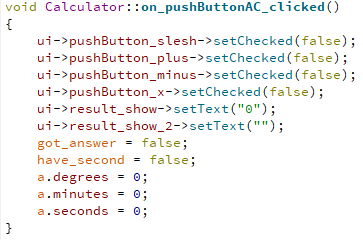


* 1. После добавления возможности использовать различные математические операции, необходимо создать кнопку equal и функцию для вычисления ответа on\_pushButton\_equal\_clicked(). 

В функции идёт проверка, на какую ранее операцию было нажато. И в зависимости от этого, выполняются соответствующие действия: сложение, вычитание, умножение, деление. Функция возвращает глобальную переменную answer, содержащую ответ.

* 1. Для удаления введенных значений создаются 2 кнопки: AC и backspace.



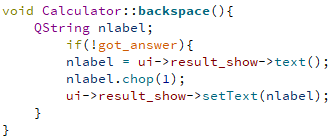
Для кнопки AC создаётся функция on\_pushButtonAC\_clicked().

Данная функция удаляет все введенные значения, и обнуляет некоторые глобальные переменные.

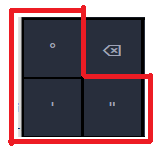
Кнопке backspace добавляется значок с помощью Unicode: QChar m\_backspace = QChar(0x0000232B);

ui->pushButton\_backspace->setText(m\_backspace);

Функция backspace



Данная функция удаляет из первой выводящей строки по 1 символу, что позволяет корректировать введенное значение.

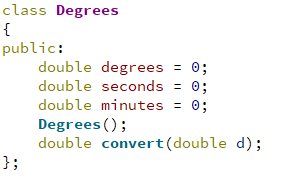
* 1. Теперь калькулятор необходимо научить работать с градусами, минутами и секундами. Создаются кнопки градуса, минуты и секунды

Эти кнопки также соединяются со своей функцией 

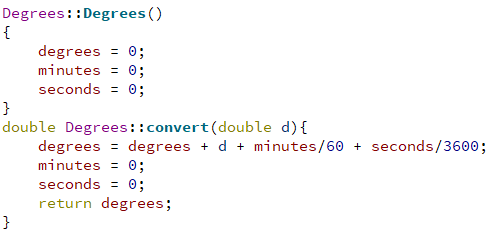
В функции convert() ведётся вся работа с градусами. 

В строку добавляются знаки градуса, минут и секунд. В функции идёт установка условий на то, что бы нельзя было вводить сразу нескольких знаков. Также идёт разделение строки на части: при вводе знака градуса, все цифры стоящие перед этим знаком вносятся в поле minutes класса Degrees. Тоже самое происходит при вводе остальных символов.

Класс Degrees использует для более удобной работы с градусными мерами. В нем есть три поля: degrees, minutes, seconds. А также метод convert.



Поля используются лишь для хранения значений, а в методе convert идёт перевод всех введённых значений в простые градусы, для более удобной работы с ними.

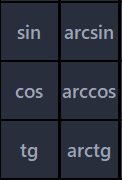


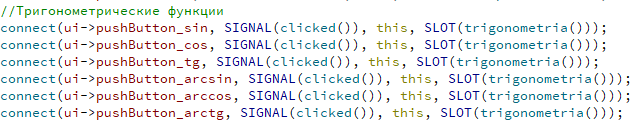
В метод конверт минуты делятся на 60, секунды на 3600 для их перевода в градусы.

После ввода необходимого значения, оно конвертируется, и в конвертированном виде показывается на второй выводящей строке, и дальнейшие действия будут проводится над конвертированным значением.



* 1. Для работы с тригонометрическими и обратными тригонометрическими функциями создаются кнопки



Кнопки соединяются с функцией trigonometria() 

В функции trigonometria() проверяется на какую кнопку было нажато, и выполняется соответствующие действия.

Например при нажатии кнопки sin ранее введённое значение переводится в радианы, поскольку тригонометрические функции из библиотеки #include <QtMath> работают с радиальными значениями.

Работа с обратными тригонометрическими функциями немного отличается

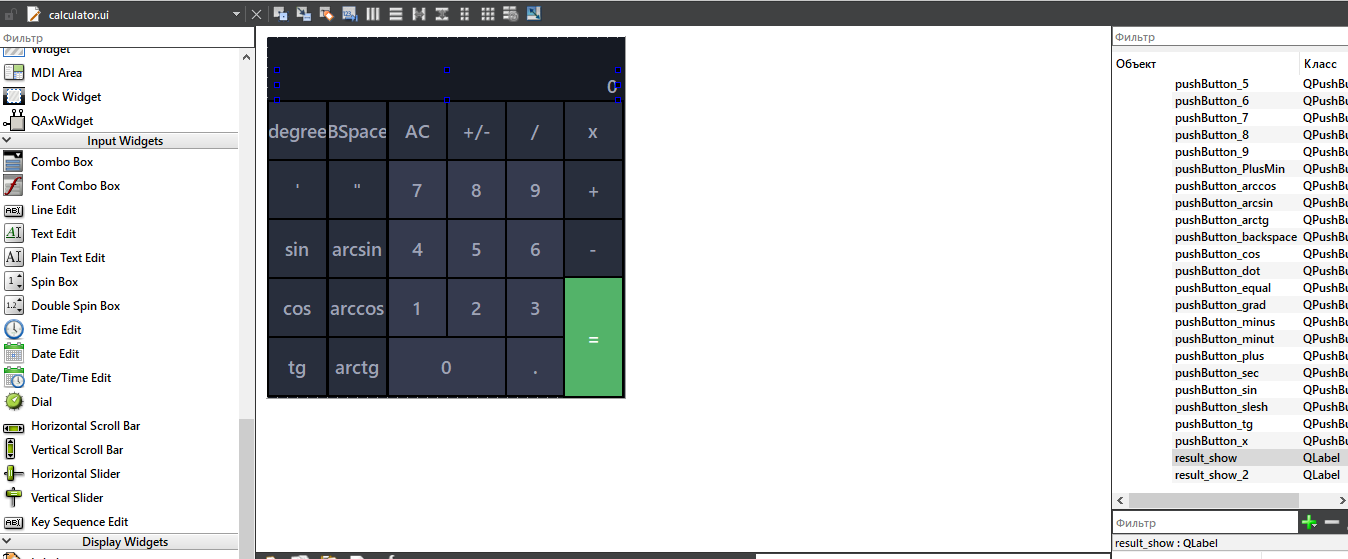


поскольку обратные тригонометрические работают с значениями от [-1; 1] и не принимают в качестве числового значения градусы.



Пример вычислений.

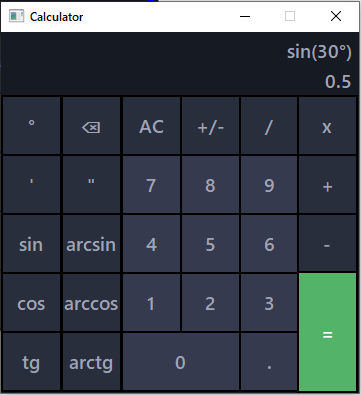
**1.3. Интерфейс приложения**

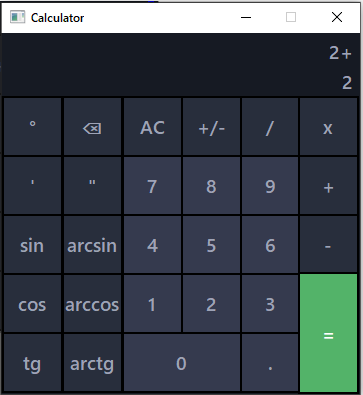
****

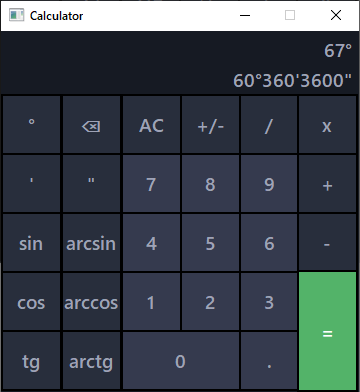
Интерфейс приложения был создан средствами редактора Qt creator.

Каждой кнопке стиль был задан с помощью styleSheet. Также все элементы интерфейса были выравнены по координатам.

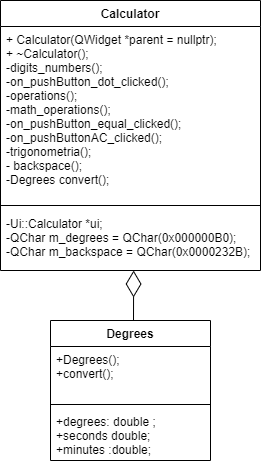
**1.4. Скриншоты работы приложения**

****

****

****

**1.5. UML-диаграмма.**



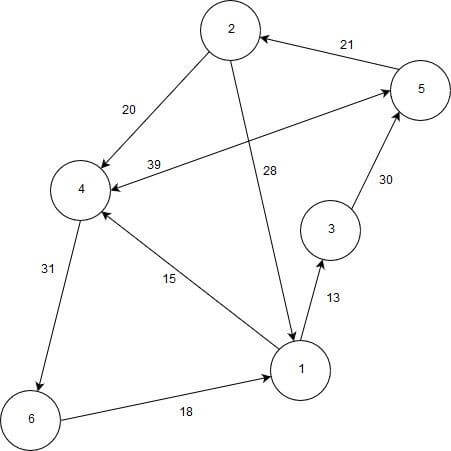
1. **Разработка приложения по решению и визуализации задачи о коммивояжере**
   1. **Постановка задачи.**

-Реализовать нахождение кратчайшего маршрута на графе

-Визуализировать граф

-Решить задачу коммивояжера.

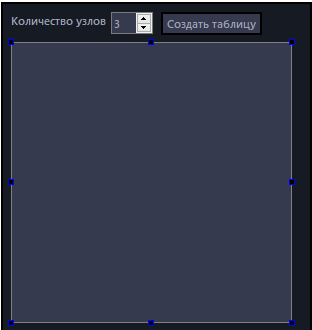
**2.2. Анализ задачи**



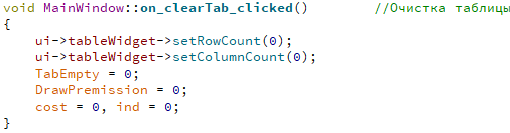
* 1. Для начала создаётся объект QSpinBox и выставляется значения, которые данный объект может принимать от 3 до 99.  с помощью данного объекта пользователь будет вводить количество узлов графа.
  2. Создаётся кнопка makeTab, при нажатии которой вызывается функция on\_makeTab\_clicked(). Данная функция создает таблицу размером n на n(где n – количество узлов).

Также глобальная переменная TabEmpty типа bool принимает значение true.

Таблица строиться внутри объекта tableWidget.

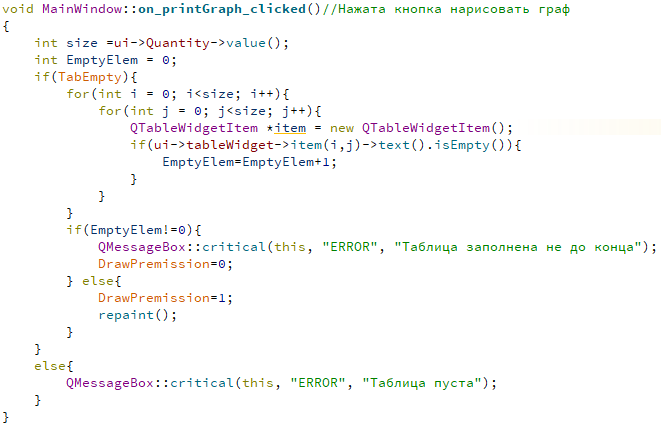


* 1. После создания таблицы, необходимо добавить функция для её удаления. Для этого создается кнопка “очистить таблицу”, при нажатии которой вызывается функция on\_clearTab\_clicked()



Данная функция выставляет нулевое значение количеству столбцов и строк таблицы, тем самым удаляя её.

* 1. Теперь необходимо реализовать функция рисования графа. Исходные данные для графа будут браться из таблицы.

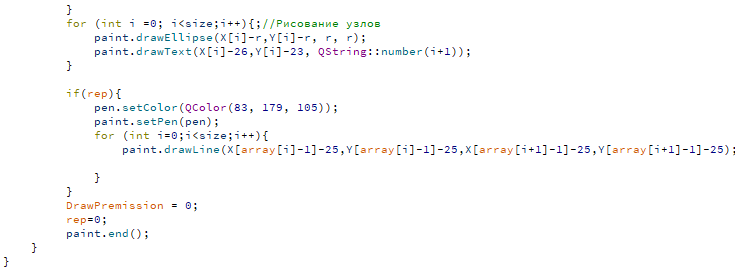
Создаётся кнопка “печать графа”, при нажатии которой вызывается функция on\_printGraph\_clicked().

В функции идёт проверка на пустоту всей таблицы и на наполненность её всех ячеек. Если вся таблица полна исполняется команда repaint(), которая вызывает метод paintEvent().

* 1. Метод paintEvent() переопределён в классе MainWindow.

paintEvent() рисует узлы графа и их ребра. Для всего графа задается центральная точка, радиус графа, радиус узлов, так как узлы изображаются в виде кругов. По формулам рассчитываются центры узлов и в этих точках рисуются. Также проверяется наличие связи между узлами по ранее созданной таблице смежности. Если связь присутствует, то узлы соединяются линией, то есть ребром.

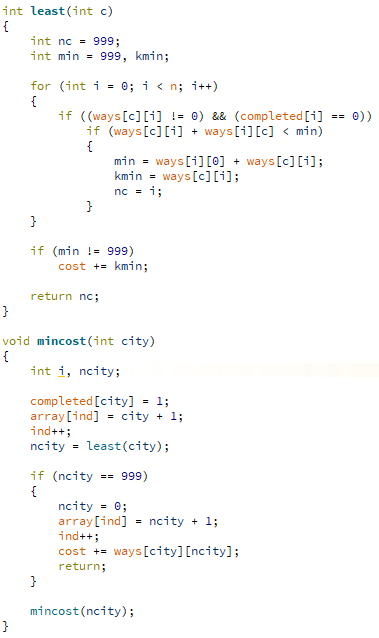




В методе также объекту QPainter задаётся цвет кисти, ручку и их стиль.

* 1. После того как были созданы функции для рисования графа, создается кнопка “найти кратчайший путь”. По нажатию вызывается функция on\_findMinWay\_clicked(). В функции проверяется условие не пуста ли таблица, после чего данные из таблицы заносятся в двумерный массив ways, заданный глобально. После чего идёт работа двух других функций: mincost() и least().





В функция mincost() принимает в качестве параметра номер узла с которого будет начат поиск кратчайшего маршрута. В массивы completed и array заносятся номера отработанных путей и узлов.

После чего вызывается функция least().

В функции least() идёт перебор дорог, и в переменную cost присваивается длина самого короткого маршрута. Данная функция возвращает номер узла, на котором остановилась работа цикла for.

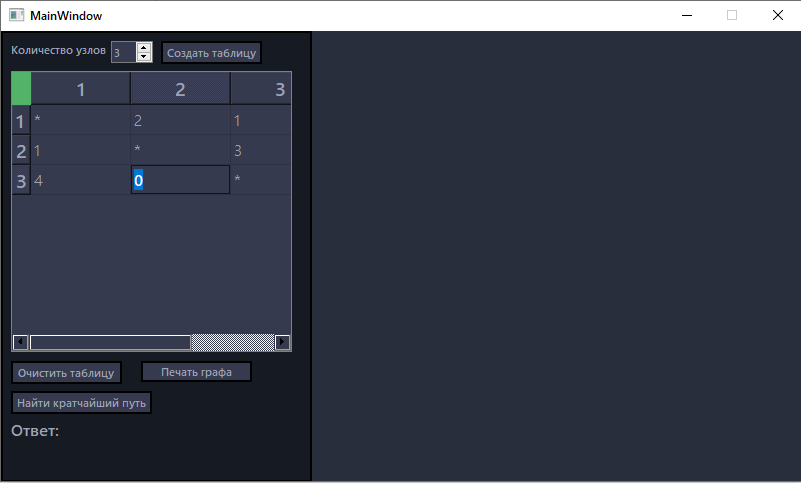
Функция mincost() будет вызывать рекурсивно, пока не будет найдем маршрут, проходящий через все узлы и возвращающийся в начальную позицию. Начальной позицией всегда является первый узел.

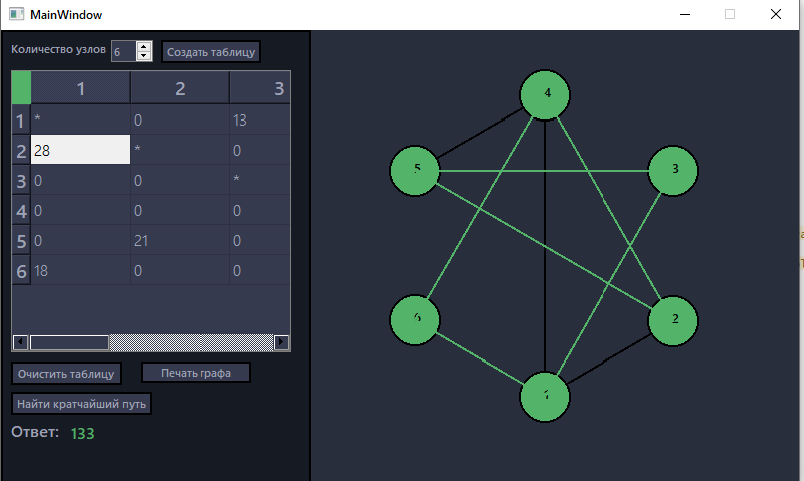
Если маршрут не удалось найти, то будет выведен ответ, что маршрута не существует, иначе будет выдан ответ – длина маршрута.

1. **Интерфейс приложения**

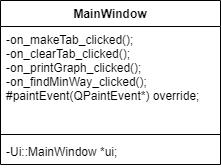
****

1. **Скриншоты работы приложения**

****

****

1. **UML-диаграмма**



**Ссылки:**

GitHub: <https://github.com/IsmagilovAinaz/Creative-Works>

Документация Qt: https://doc.qt.io