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

Выполнила:

студентка группы РИС-20-2б

Сахарова Алина Романовна

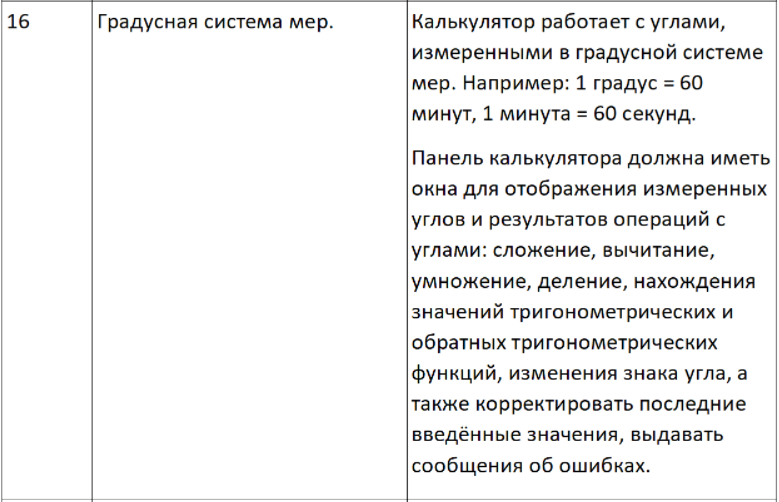
**Пермь, 2021**

**Первая часть. Калькулятор**

**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.

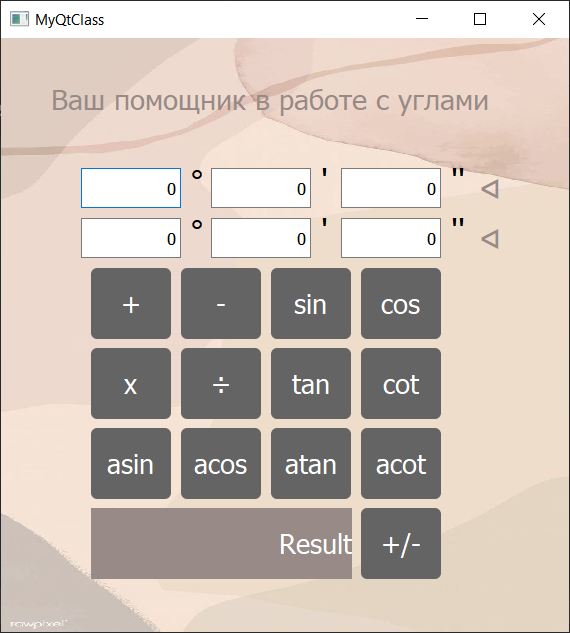
Вариант 16.



**Анализ задачи**

Реализация калькулятора была начата с создания интерфейса. Были добавлены две строки для ввода пользователем необходимых значений (в каждой строке по 3 поля: для ввода значения градусов, минут и секунд), кнопки для необходимых математических действий (всего кнопок 13: для действий смены знака числа, сложения, вычитания, умножения, деления, нахождения тригонометрических и обратных тригонометрических функций), поле для вывода полученного в ходе вычислений значения выбранной операции, кнопки удаления введенных значений для всей строки, а так же для удобства восприятия возле каждого окна для ввода добавлен специальный символ, показывающий, что в данное поле вводятся либо градусы, либо секунды, либо минуты.

Интерфейс получился таким:



Как же работает данный калькулятор? Представьте, что Вы пользователь, которому необходимо выполнить какую-либо операцию с углами. Итак, Вы начинаете свою работу с калькулятором и заполняете поля необходимыми значениями. Рассмотрим несколько возможных ситуаций дальнейшего развития событий:

1. По случайности вводятся запрещенные для ввода символы (разрешенными символами считаются цифры, все остальные символы считаются программой запрещенными). В этой ситуации ничего не произойдет и символы просто не появятся в окне, так как заранее предусмотрен ограничитель для вводимых значений, который не позволяет ввести те самые запрещенные символы;
2. На ввод поступают цифры, но Вы пытаетесь ввести вещественное число. Для такой ситуации предусмотрен тот же ограничитель, что и для запрещенных символов, который не даст поставить точку и записать вещественное число. Почему вводить можно только целые числа? Например, в поле для значения градусов поступает вещественное число 54, 26; Целая часть этого числа-это градусы (54), оставшуюся часть числа умножаем на 60 и получаем 15,6, где целая часть числа-это минуты (15), повторив предыдущее действие, получаем число 36-это секунды. Из примера можно сделать вывод, что значение не может быть вещественным числом;
3. Теперь Вы вводите целое число, но видите ошибку, которая сообщает о том, что значение введено некорректно. В этом случае нужно проверить знак каждого записанного значения угла. Если значение градусов было положительным числом, то программа ждет на ввод положительные значения минут и секунд, и напротив – если значение градусов отрицательное, то значения минут и секунд тоже.

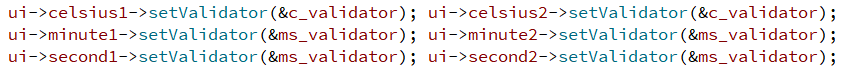
Все ситуации с возможными ошибками рассмотрены. Теперь Вы вводите все правильно и корректно: на ввод поступают числа целого типа, которые имеют верные знаки. В таком случае доступно выполнение математических, тригонометрических и обратных тригонометрических операций. Для математических (сложение, вычитание, умножение и деление) необходимо заполнить значениями обе строки, результат отобразится в поле result. Для тригонометрических и обратно тригонометрических операций, для операции смены знака необходимо заполнить только верхнюю строку, поэтому при несоблюдении этого условия выйдет ошибка, сообщающая о том, что пользователю необходимо ввести единственное значение угла в верхнюю строку.

Это то, что видит пользователь при работает в калькуляторе. Перейдем непосредственно к самой программе, которая позволяет выполнять все описанные выше действия. Пойдем по порядку:

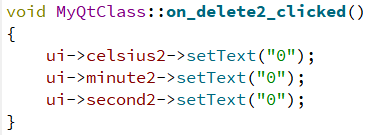
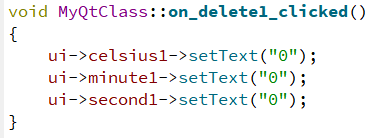
1. Ограничение для вводимых пользователем данных было реализовано с помощью валидатора. Для этого в заголовочном файле программы подключается библиотека QIntValidator (int, потому что значения могут быть только целыми) и в класс добавляются переменные типа QIntValidator.



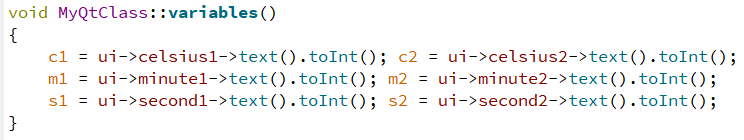
Для того, чтобы ограничитель работал в конструкторе класса необходимо добавить следующее:



1. Действия кнопок удаления реализованы следующим образом: если нажата верхняя кнопка удаления, то во все поля соответствующей нажатой кнопке строки устанавливаются значения 0, то есть первоначальные значения, если нажата нижняя кнопка, то, соответственно, те же действия для нижней строки.

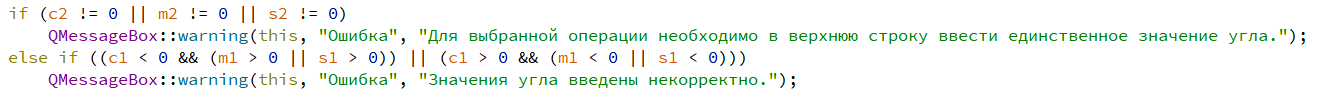


1. Как программа получает введенные пользователем значения? Через следующие действия:

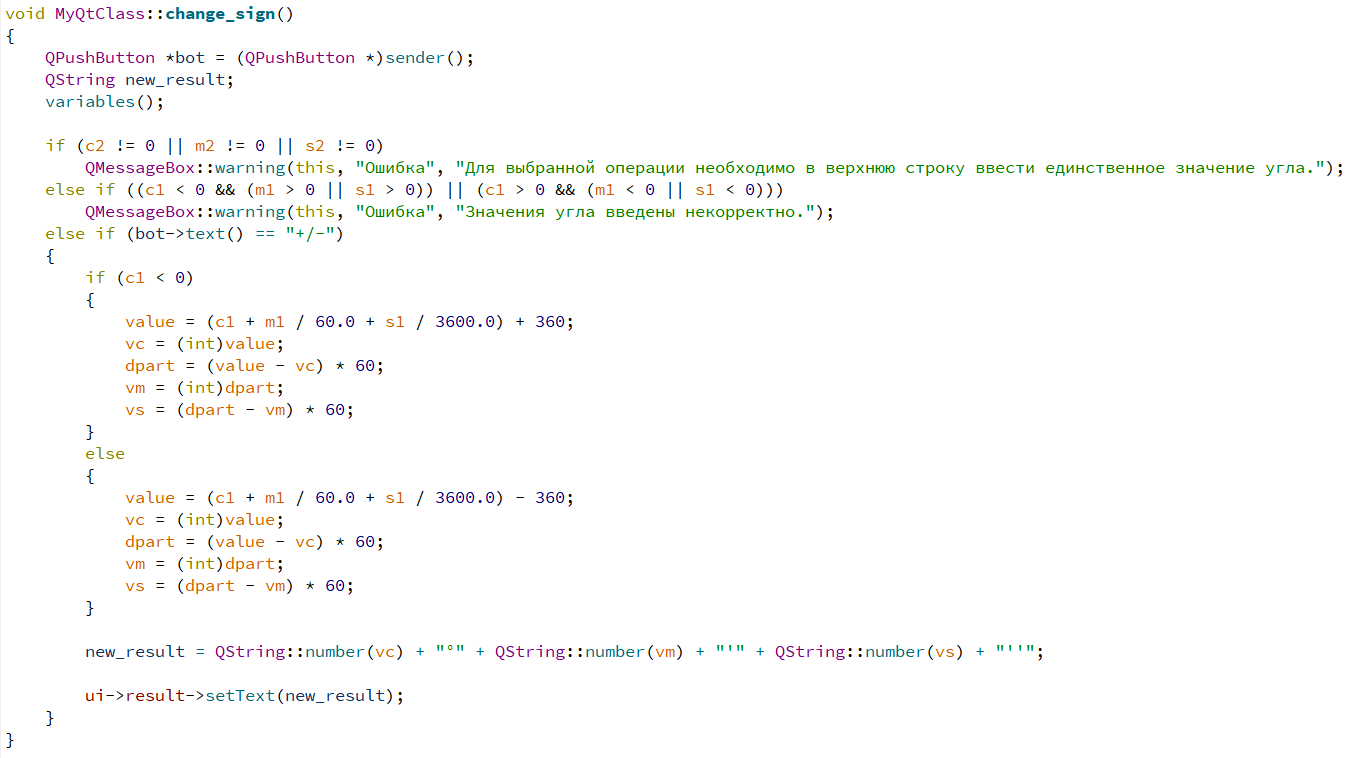


Переменной типа int, объявленной глобально, присваивается введенное значение, преобразованное к тому типу, что у переменной. Далее эти переменные используются для действий со значениями углов.

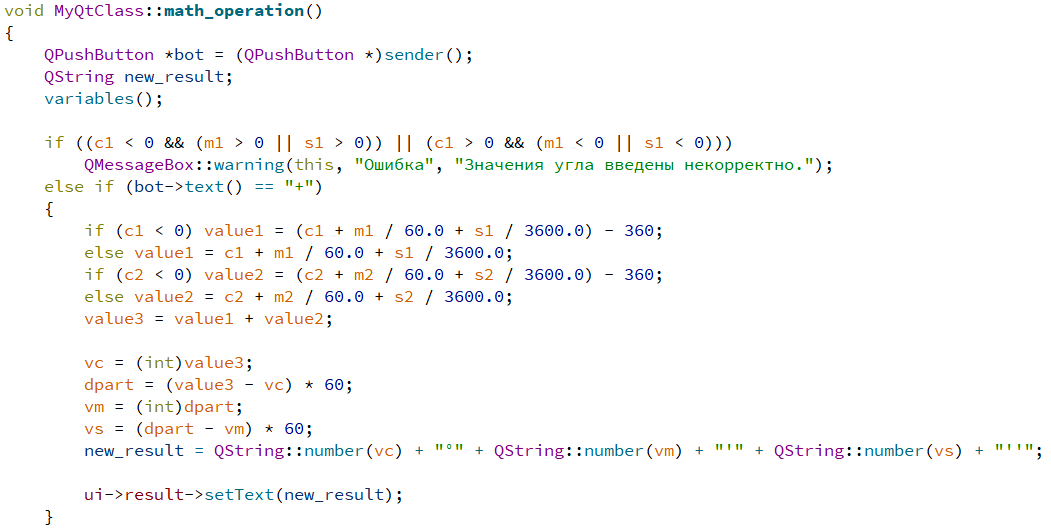
1. Остались операции. Первая операция – операция смены знака угла. В этой функции предусмотрен вывод ошибок. Для того, чтобы они выводились, когда это необходимо, добавлена библиотека QMessageBox, а в самой функции прописаны определенные критерии, которые проверяют введенные значения на неправильность, и если введенные данные удовлетворяют заданным критериям, то делается вывод ошибки.



Если же введенные значения критериям не удовлетворяют, то есть являются верными, то тогда выполняется действие смены знака. Прежде всего вызывается функция, в которой переменным присваиваются введенные значения (см. пункт 3). Как реализовано данное действие: для введенного угла вычисляется эквивалентный ему положительный или отрицательный (в зависимости от знака введенного значения) угол.

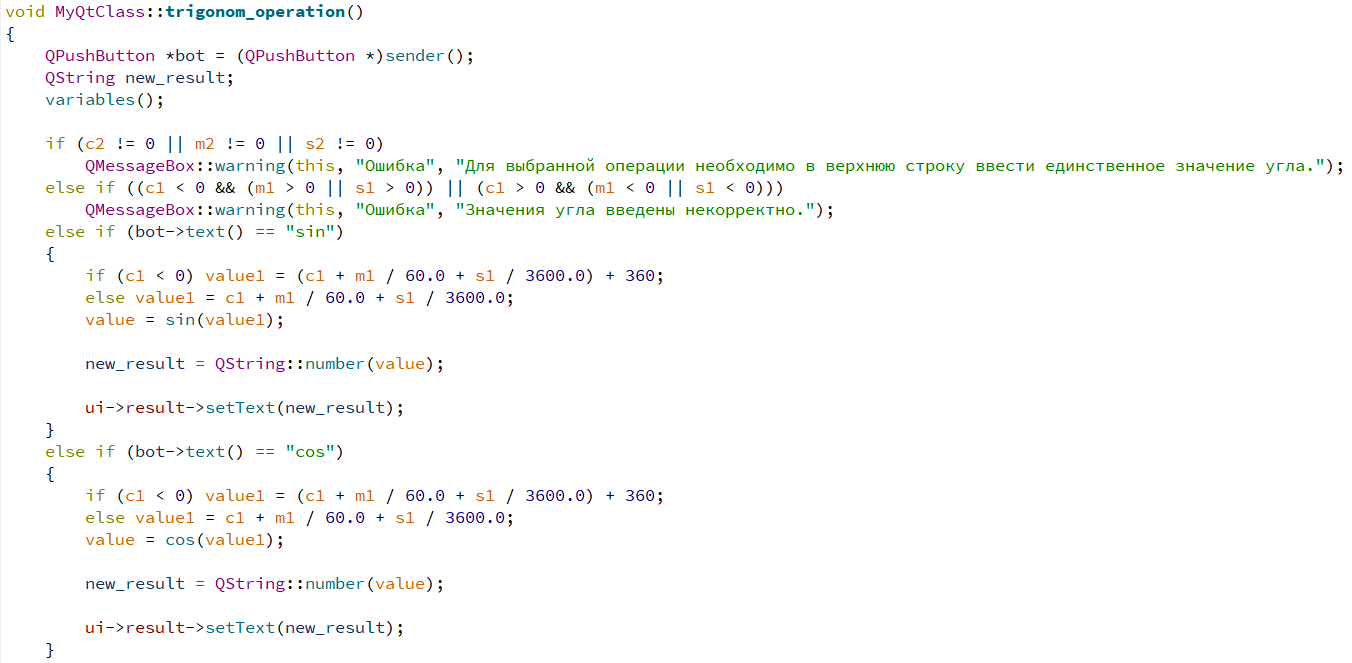


1. Следующие операции – математические. В них так же присутствует проверка введенных значений на корректность, если они не удовлетворяют им и ошибка не была вызвана, то тогда выполняются действия сложения, вычитания, умножения и деления.

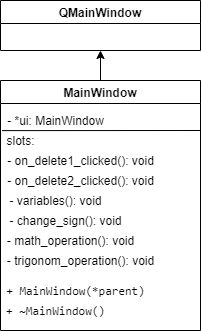


Прежде всего вызывается функция, в которой переменным присваиваются введенные значения (см. пункт 3), полученные значения используются в дальнейших вычислениях. Программа понимает, какое действие нужно выполнять по тексту на кнопке, которая была нажата. Для положительного угла программа находит значение угла в градусах, выполняет необходимо математическое действие, заново считает новые полученные значения градусов, минут и секунд; для отрицательного угла программа первоначально находит эквивалентный ему положительный угол, а далее повторяет действия для положительного угла (на скриншоте рассмотрена операция сложения, так как остальные операции выполняются аналогично). Метод перевода значений в градусы и обратно в отдельные значения градусов, минут и секунд позволяет избежать ситуации, когда при сложении, например, минут получатся такое значение, в котором можно выделить какое-либо значение градусов. Например, при сложении 30 и 50 минут получаем значение 80, можем выделить 60 минут, которые равны 1 градусу и оставшиеся 20 минут.

1. Следующие операции тригонометрические и обратно тригонометрические. Все действия аналогичны математическим операциям.



**UML-диаграммы**

****

**Вторая часть. Задача коммивояжера**

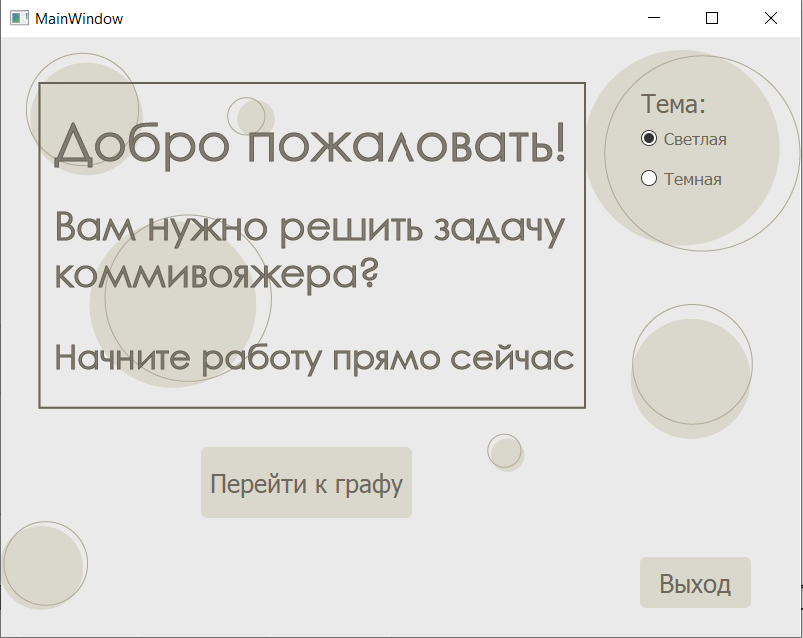
**Постановка задачи**

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

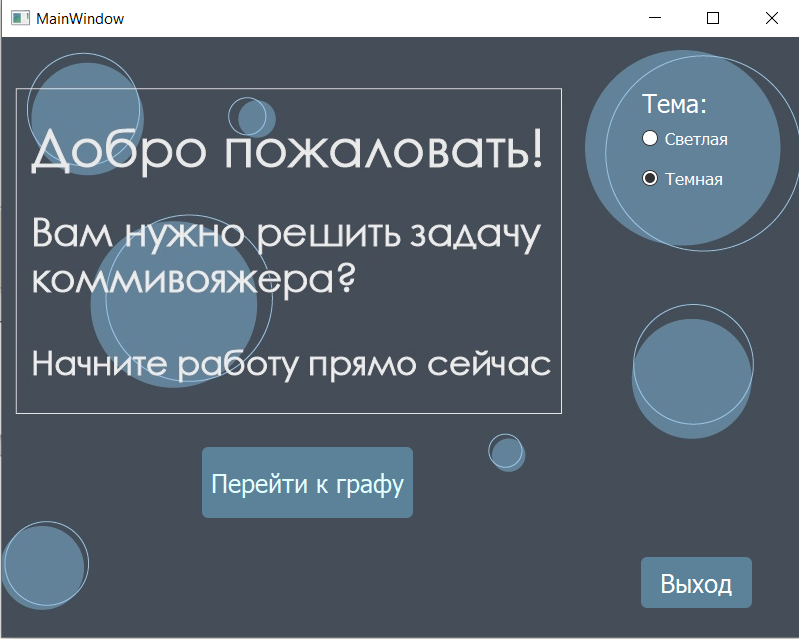
**Анализ задачи**

Рассмотрение выполненного задания начнем с интерфейса. Открывая проект, пользователь попадает в главное окно. В это окно были добавлены кнопки закрытия данного окна и перехода к окну с графом, а так же возможность смены темы во всем проекте.

Интерфейс главного окна со светлой темой (по умолчанию выбрана светлая тема):



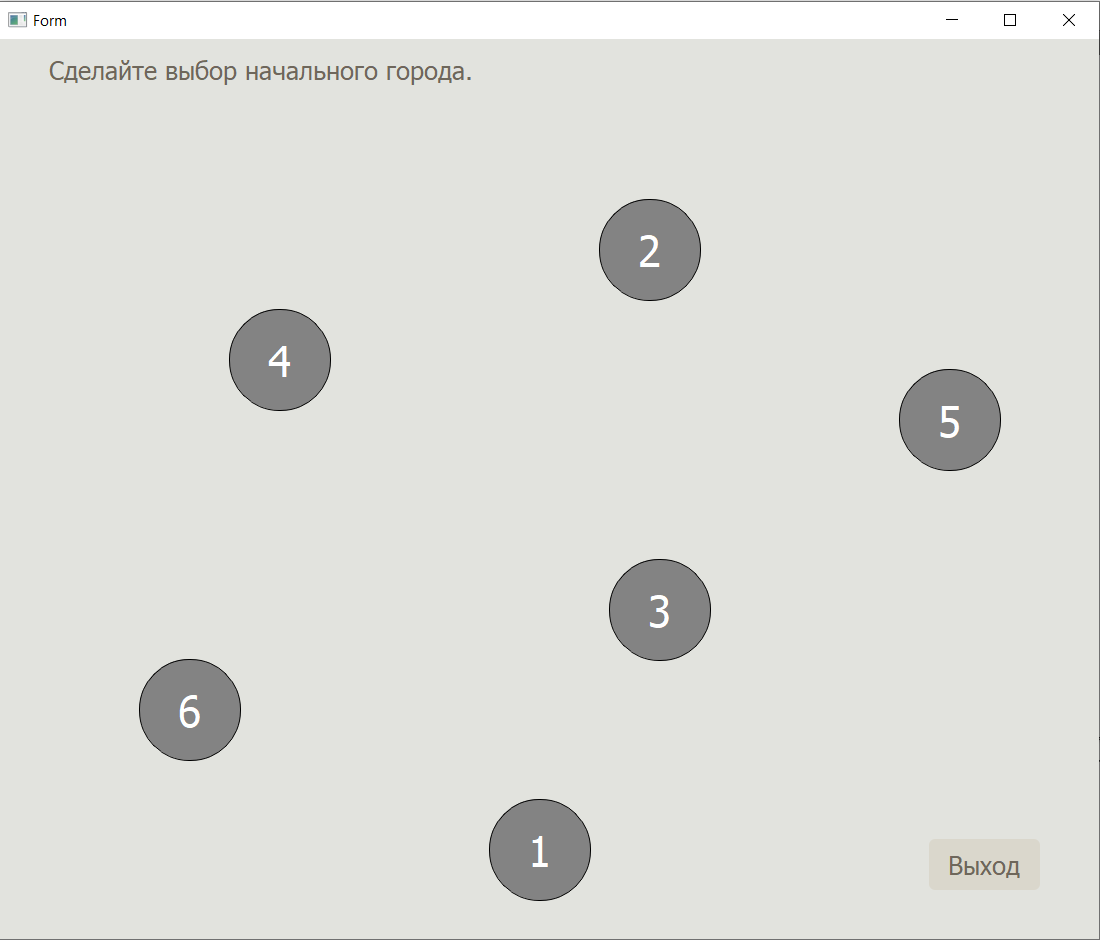
И с темной:



После нажатия пользователем кнопки «перейти к графу» откроется второе окно, в котором происходит решение задачи и отрисовка дорог в графе.

Окно имеет 6 кнопок (узлов) графа: первые две нажатые кнопки становятся начальным и конечным городами соответственно. После выбора пользователем городов, для которых находится кратчайшее расстояние, появляется поле ввода и кнопка «Ввести» (до этого момента они скрыты от пользователя). После этого можно начать осуществлять ввод необходимых расстояний, одновременно с введенным значением будет рисоваться дорога, проходящая через те, города, номера которых были указаны в заголовке при вводе значения. Окно с графом имеет кнопку для возможности создания нового графа, после чего можно будет заново выбрать необходимые начальный и конечный города, ввести новые значения расстояний и решить для этого графа задачу коммивояжера, а так же кнопку выхода, при нажатии которой пользователь возвращается в главное окно. Мной, как автором, предусмотрена такая ситуация, когда после закрытия окна с графом и возвращения на главный экран пользователь снова выберет действие открытия окна с графом, поэтому в таком случае так же доступно создание графа и решение для него задачи коммивояжера.

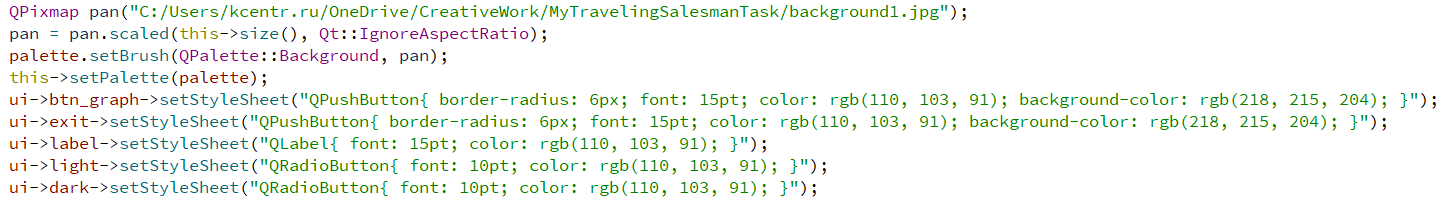
Интерфейс второго окна до выбора городов и ввода значений (без отрисованных линий):



Перейдем к программной реализации проекта.

1. Главное окно. Имеет функцию смены темы, закрытия окна и перехода ко второму окну с графом. Рассмотрим их реализацию по порядку:

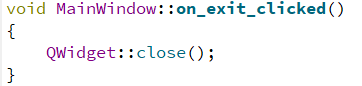
1. Функция смены темы. Добавить тему для окна несложно, достаточно прописать необходимые изменения таким образом:



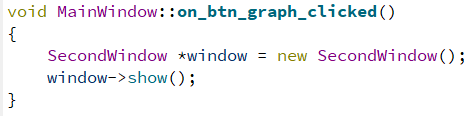
Но что было использовано для того, чтобы выбранная пользователем в главном окне тема изменялась и во втором на соответствующую? Ответ: добавлена глобальная переменная типа bool, которая принимает значение 1, если выбрана первая тема, и значение 0, если выбрана вторая. Во втором окне в конструкторе добавлена проверка, которая определяет значение добавленной переменной и в зависимости от него устанавливает необходимую тему, которая будет соответствовать той теме, которая установлена в главном окне.



2. Функция закрытия окна. Содержит одно действие, которое закрывает виджет.



3. Функция перехода ко второму окну (открытие окна с графом).



1. Второе окно с графом. Содержит ограничение для вводимых пользователем значений, предоставляет возможность решения задачи коммивояжера на заданном графе, создания нового графа, для которого так же решается задача коммивояжера, и закрытия окна. Рассмотрим их программную реализацию по порядку:

1. Ограничение для вводимых данных реализовано с помощью валидатора. Для этого в заголовочном файле программы подключается библиотека QIntValidator (int, потому что значения могут быть только целыми) и в класс добавляются переменная типа QintValidator.



Чтобы ограничение применить к полю для ввода необходимо сделать следующие действия в конструкторе класса:

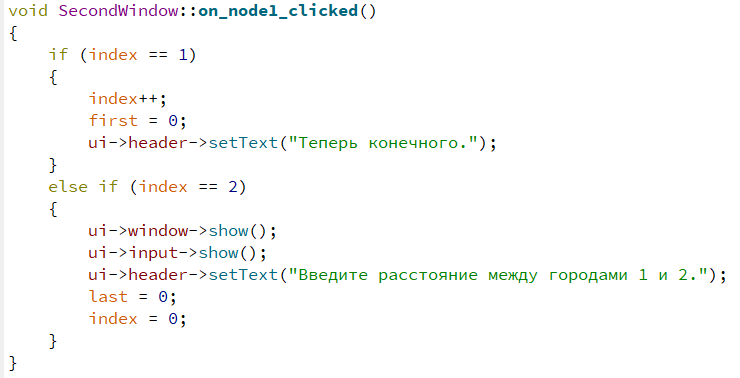


Так же можно установить разрешенный диапазон для вводимой информации:



Минимальное значение 0, так как значение пути не может быть отрицательным числом, а 0 означает, что дороги нет.

2. Решение задачи коммивояжера. Рассмотрим подобные функции для выбора начального и конечного городов (на примере рассмотрим первую):



Если индекс равен 1, то выбирается начальный город, если 2 – конечный, если индекс равен 0, то ничего не происходит. После выбора начального города в заголовок устанавливается новый текст (просьба выбрать конечный город). После того, как пользователь выбирает конечный город, то индекс становится равным 0, чтобы последующее нажатие на кнопки стало бесполезно, а так же становится доступной линия ввода и кнопка «Ввести» (изначально они скрыты).

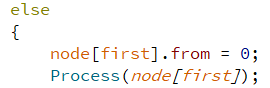
В последующих функциях выполняются аналогичные действия для остальных узлов.

После того, как пользователь нажимает кнопку «Ввести», вводятся

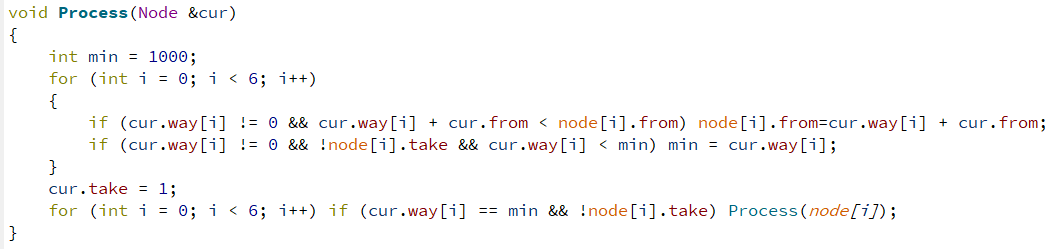
расстояния для городов, номера которых указаны в заголовке.



Когда все расстояния заполнены, вызывается функция, определяющая наименьшее расстояние между выбранными городами (первым выбранным городом устанавливается начальный, для него вызывается функция).



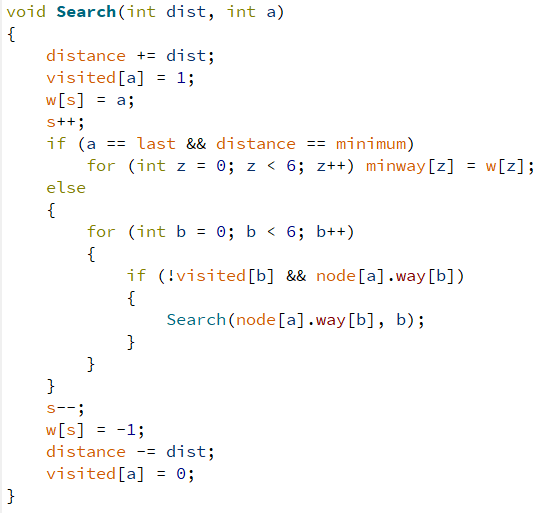
Функция работает по алгоритму Дейкстры и находит расстояния от первого выбранного города до всех остальных.

Поскольку необходимо найти расстояние до второго выбранного города, то минимальному значению присваивается расстояние от первого выбранного узла до второго.



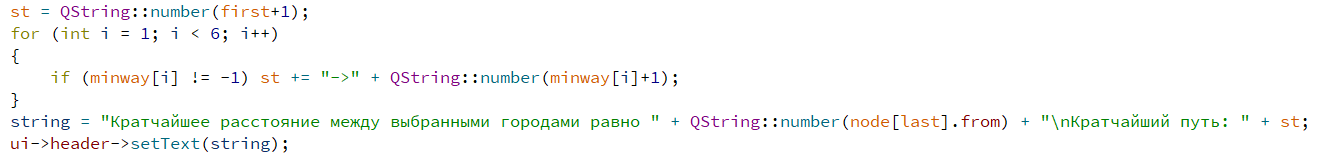
Переменная from есть у каждого узла. Она указывает на расстояние от этого узла до первого выбранного.

Затем вызывается функция Search.



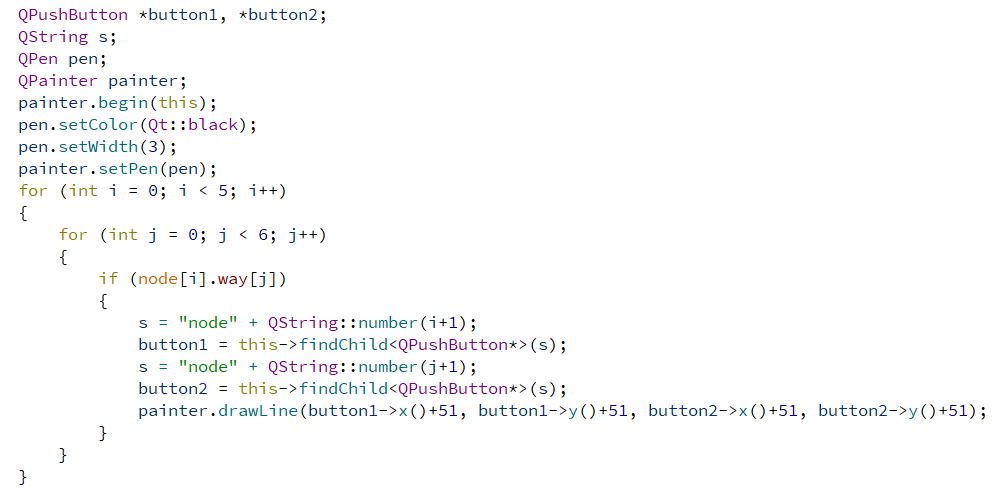
Она проходит по всем существующим маршрутам из первого выбранного узла. В течение этого прохождения маршрут записывается в массив w. Если найден маршрут, конечный узел которого соответствует последнему выбранному, и расстояние до него ранее вычисленному минимальному значению, то из массива w он сохраняется в массив minway.

Когда кратчайший путь найден, то он записывается в строчную переменную и выводится в заголовок.



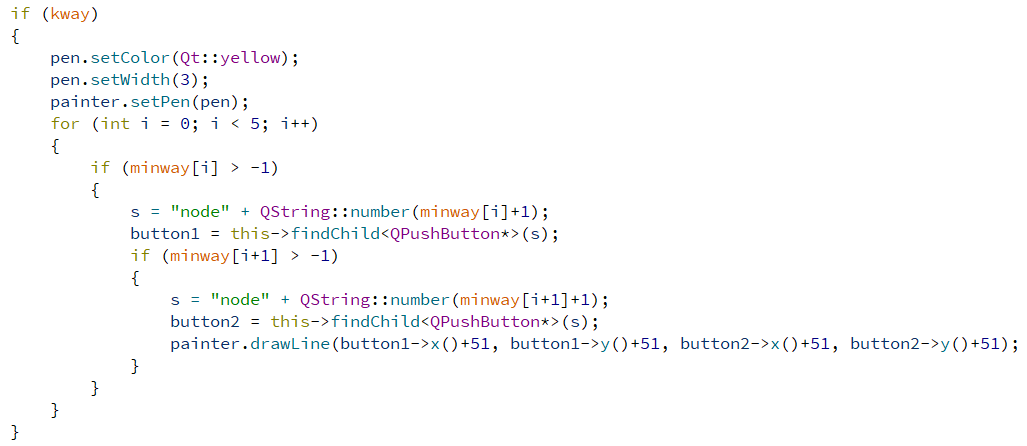
Пути между годами рисуются в функции paintEvent.

Для обычных путей в этой функции выполняется проверка существования пути между городами, после этого в строку заносится название узла и с помощью функции findChild находится объект класса, указанный в < >, то есть кнопка. Адрес найденной кнопки сохраняется в указатель \*button1 (аналогично с \*button2).

Для того, чтобы выделить кратчайший путь, нужно добавить переменную типа bool, после нахождения кратчайшего пути присвоить ей значение true и вызвать функцию paintEvent (this→repaint), иначе рисунок останется старый и кратчайший путь не будет выделен.

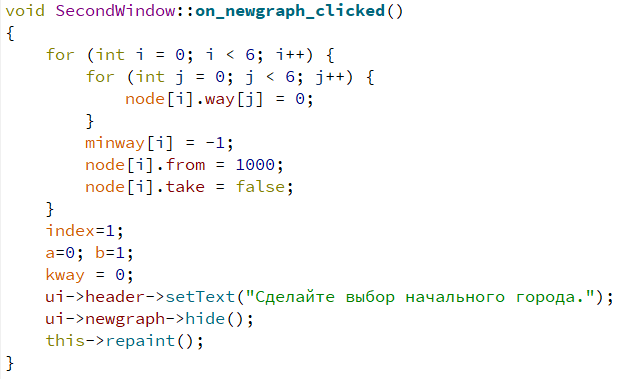


После этого в функции paintEvent добавить проверку, которая будет разрешать перерисовку линий кратчайшего пути другим цветом для его выделения.

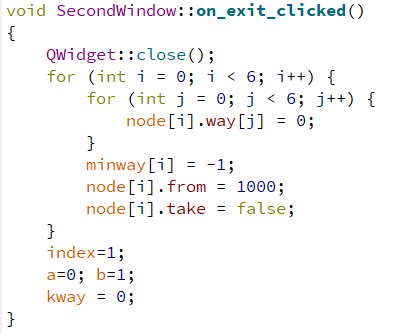


Функция drawLine рисует линию по четырем координатам. В качестве аргументов в эту функцию передается сумма координат кнопок и их радиуса, чтобы линия шла к центру кнопки.

1. Функция создания нового графа. При выборе пользователем данного действия все значения меняются на первоначальные, стираются линии и устанавливается заголовок, который установлен при первичном входе в окно с графом. Это сделано для того, чтобы после нажатия кнопки пользователь смог выбрать новые города, ввести новые значения и для нового графа решить задачу.



1. Функция выхода из окна с графом. Действия в этой функции аналогичны действиям при создании нового графа, но добавлено действие закрытия окна. Данные так же становятся равными первоначальным для того, чтобы пользователь имел возможность заново решить задачу коммивояжера, если, при попадании на главный экран после закрытия окна с графом, он снова выберет действие «Перейти к графу».



**UML-диаграммы**

