Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**Отчёт по творческой работе**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

Выполнил студент гр. ИВТ-21-1б

Ахунов Руслан Булатович

г. Пермь - 2022

**Калькулятор**

**Постановка задачи:**

1. Реализовать калькулятор Обыкновенных и десятичных дробей.
2. Создать интерфейс работы с калькулятором.

**Описание Решения**

Для создания интерфейса был выбран инструмент WindowsForms.

Трудности:

Встретились трудности работы с целой частью обыкновенной дроби, так как для целой части было создано отдельное поле, то нужно было правильно его обработать, особенно если число отрицательно, и правильно вывести в конце решения. Эта трудность была преодолена путем проверок дисплеев на отрицательные значения, и последующие изменения в поле числителя.

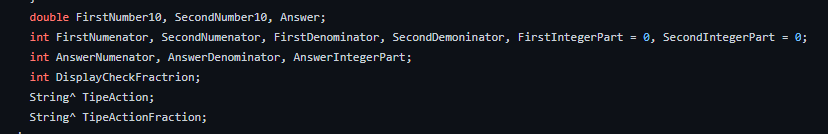
Описание Кода:

Рисунок 1 – Переменные калькулятора

В этом участке кода мы объявляем переменные для работы. FirstNumber10, SecondNumber10, Answer – для работы с десятичными значениями.   
FirstNumerator, SecondNumerator, FisrtDenominator, SecondDemoninator, FirstIntegerPart, SecondIntegerPart, AnswerNumenator, AnswerDeniminator, AnswerIntegerPart – для работы с обыкновенными дробями.   
DisplayCheckFraction – определяет тот дисплей (Числитель, Знаменатель, Целая часть), который в последний раз был активен.  
TipeAction, TipeActionFraction – записывается то действие, которое нужно выполнить. (Например: Сумма (+), Разность (-) и др.)

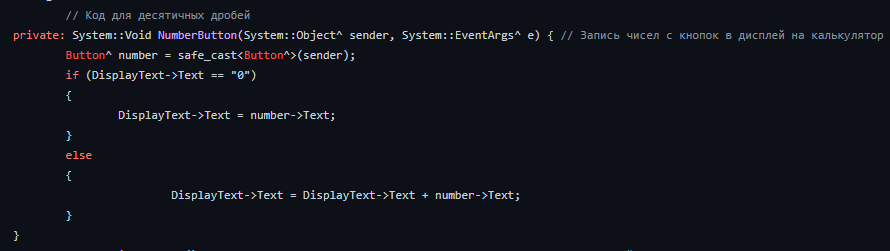


Рисунок 2 – Функция NumberButton

Функция NumberButton нужна для заполнения дисплея калькулятора соответствующими кнопками с цифрами на интерфейсе.

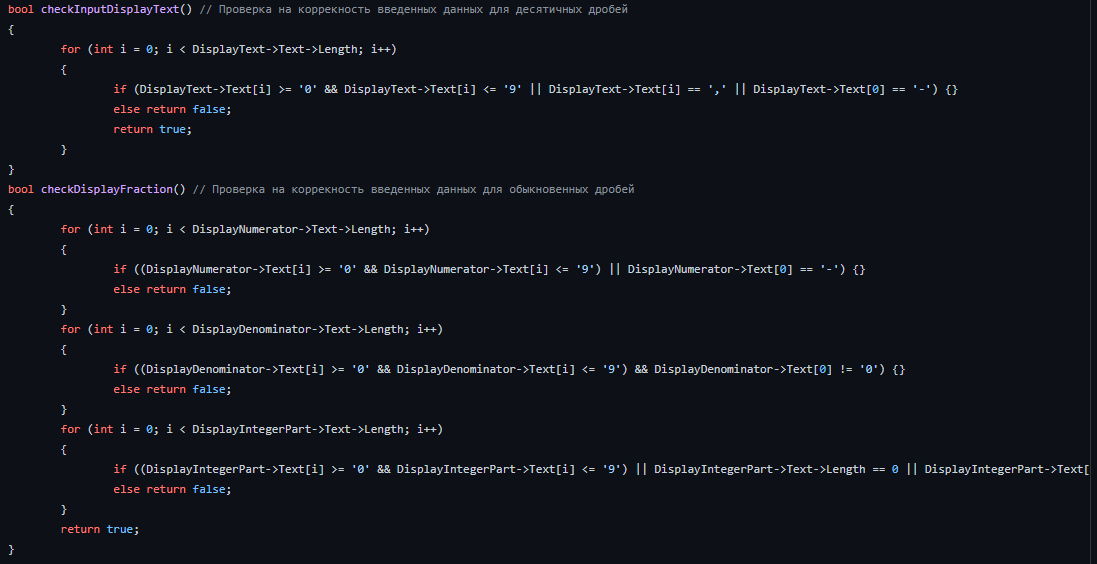


Рисунок 3 – Функции для проверки

Две функции checkInputDisplayText, checkDisplayFraction проверяют корректность введенных данных на дисплее, то есть чтобы не было никаких посторонних символов, которые нельзя преобразовать в число.

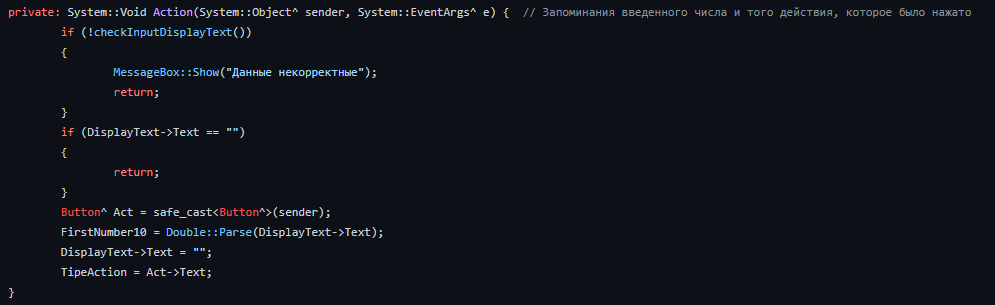


Рисунок 4 – Функция Action

Функция Action. В ней мы сперва проверяем на корректность первое число, затем запоминаем действие, которые мы нажали, и само первое число. В конце очищаем дисплей для следующего числа.

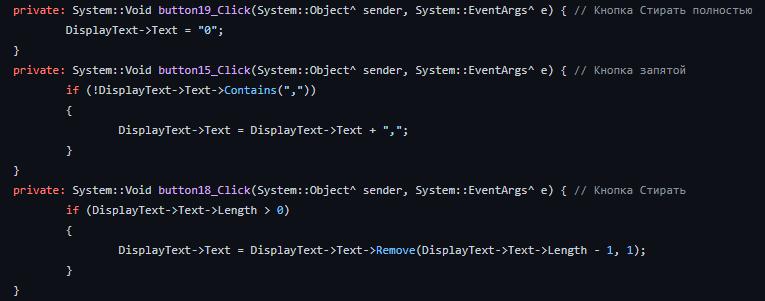


Рисунок 5 – Функции дополнительных кнопок

Функция button19\_Click – Очищает полностью дисплей, ставя в последствии 0.  
Функция button15\_Click – Кнопка постановки запятой.  
Функция button18\_Click – Убирает последнюю введенную цифру.

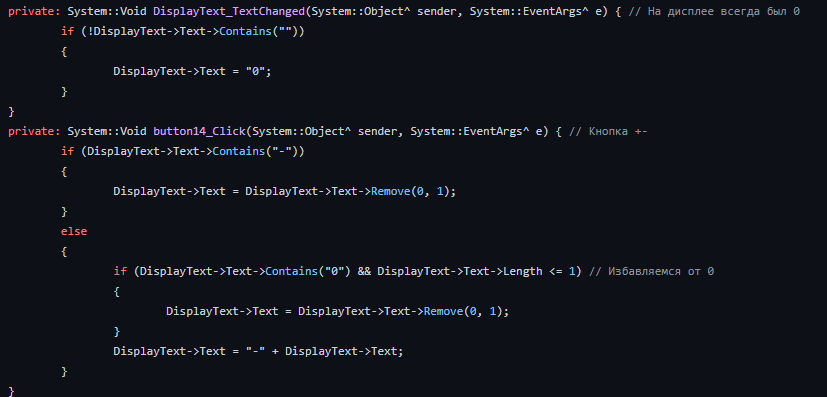
  
  
Функция DisplayText\_TextChanged – Если на дисплее нет никаких значений, то ставиться 0.

Рисунок 6 – Функции дополнительных кнопок

Функция button14\_Click – Кнопка +-, в котором проверяется есть ли минус уже в значении, если есть, то он убирается, а если его нет, то он, наоборот, ставиться, при этом убирает 0.



Рисунок 7 – Алгоритм при нажатии на кнопку =

В этой функции происходят все основные вычисления, как и в функции Action сперва идет проверка на корректность введенных данных, затем запоминается второе число. Уже дальше, с помощью переменной TipeAction, в которую мы записали действие, мы определяем какое арифметическое действие мы хотим совершить. Если сумма, то мы складываем 2 переменных FirstNumber10 и SecondNumber10, затем выводим ответ на дисплей. Такая же логика фигурирует в 3 дальнейших действия.

Дальше переходим к коду для обыкновенных дробей.

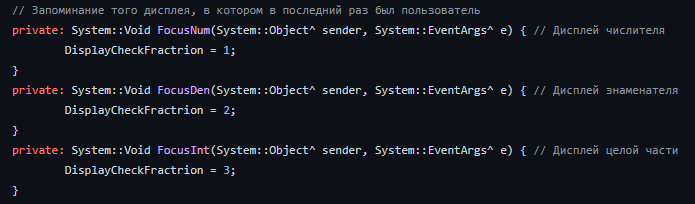
  
  
FocusNum, FocusDen, FocusInt соответственно дисплей числителя, знаменателя и целой части. Нужны для корректной работы функции NumberFraction.

Рисунок 8 – Фокусы для 3 дисплеев

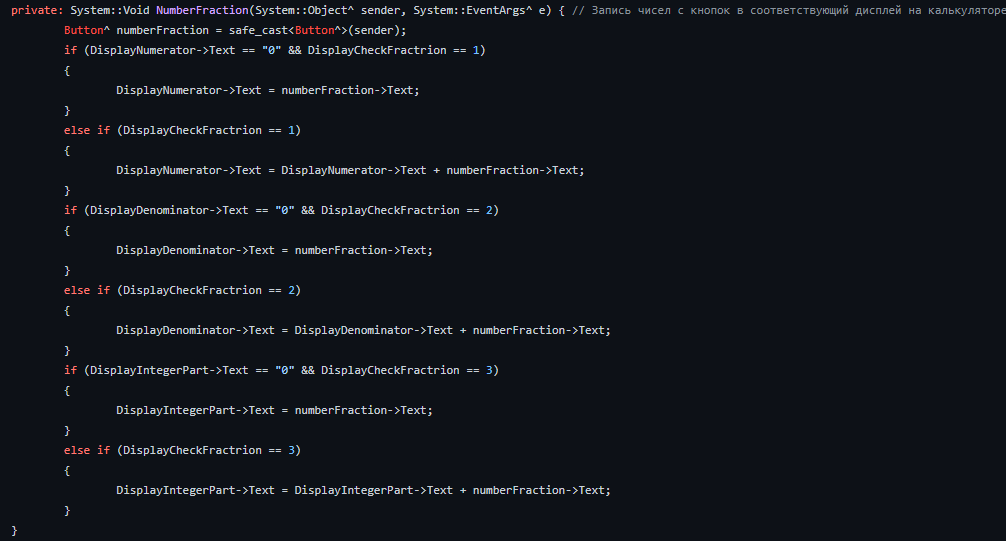


Рисунок 9 – Функция NumberFraction

Функция NumberFraction похоже на функцию NumberButton, нужна для заполнения дисплея с помощью кнопок с цифрами на интерфейсе, но при этом нужно проверять, какой из дисплеев был в последний раз активен, так как без этой проверки будут заполняться сразу 3 дисплея одновременно.

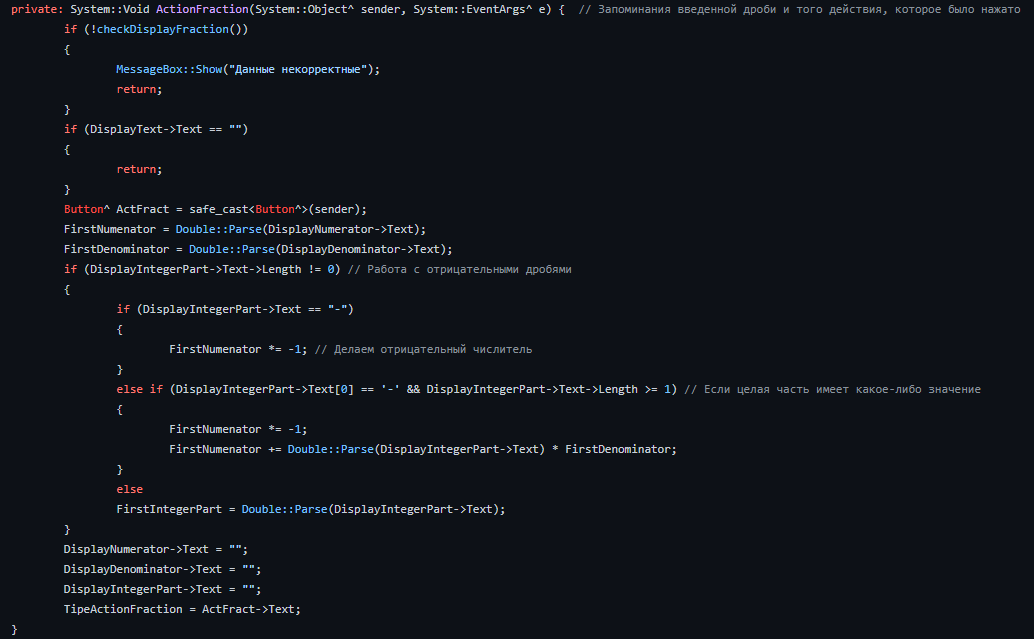


Рисунок 10 – Функция ActionFraction

Первым делом проверка на корректность введенных данных, затем запоминания числителя и знаменателя первой дроби. Для целой части идет проверка на длину и на знак, затем идет соответствующие действие с числителем или идет запоминание целой части. В конце идет стирание данных на дисплее.

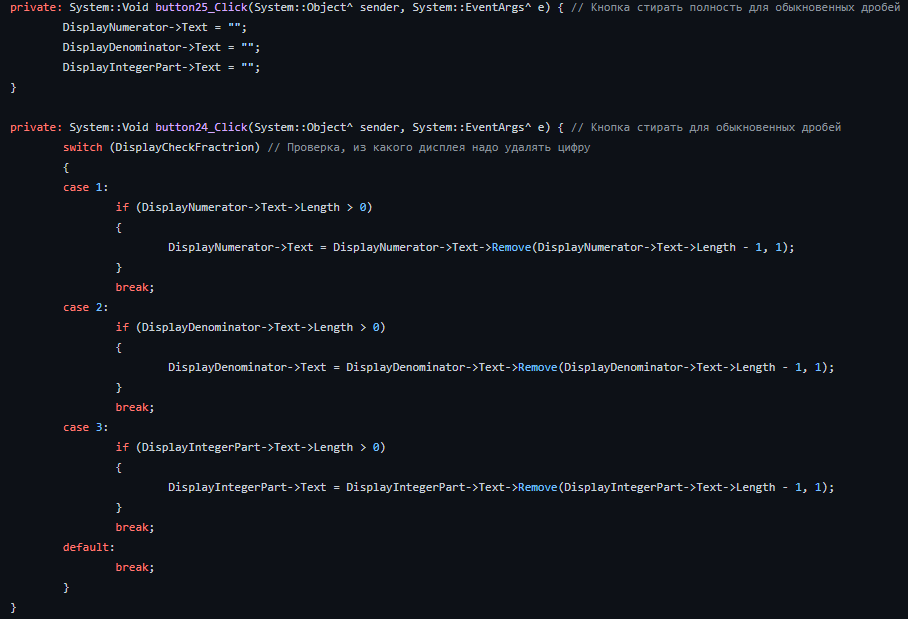


Рисунок 11 – Функции дополнительных кнопок

Функция button25\_Click – Полное стирание данных в 3 дисплеев.

Функция button24\_Click – Одиночное стирание значений, при этом нужна проверка на последний активный дисплей, так как иначе будут удаляться одновременно сразу с 3 текстовых полей.

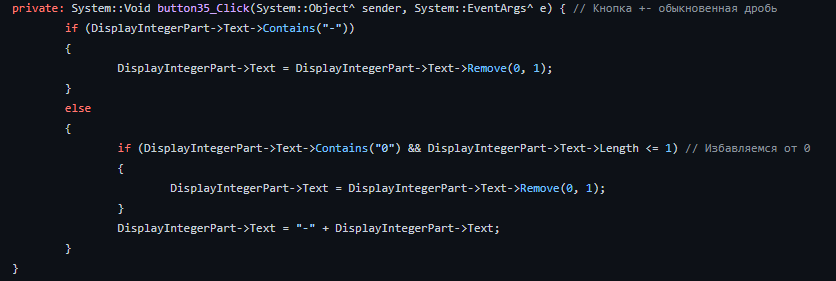
  
  
Функция button35\_Click – постановка знака +/- в целую часть дроби.

Рисунок 12 – Функции +/-

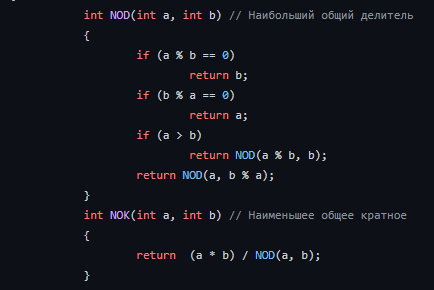


Рисунок 13 – Наибольший общий делитель и наименьшее общее кратное

Функции NOD, NOK, то есть Наибольший общий делитель и наименьшее общее кратное, нужны для нахождение общего знаменателя при сложении или вычитании обыкновенных дробей.

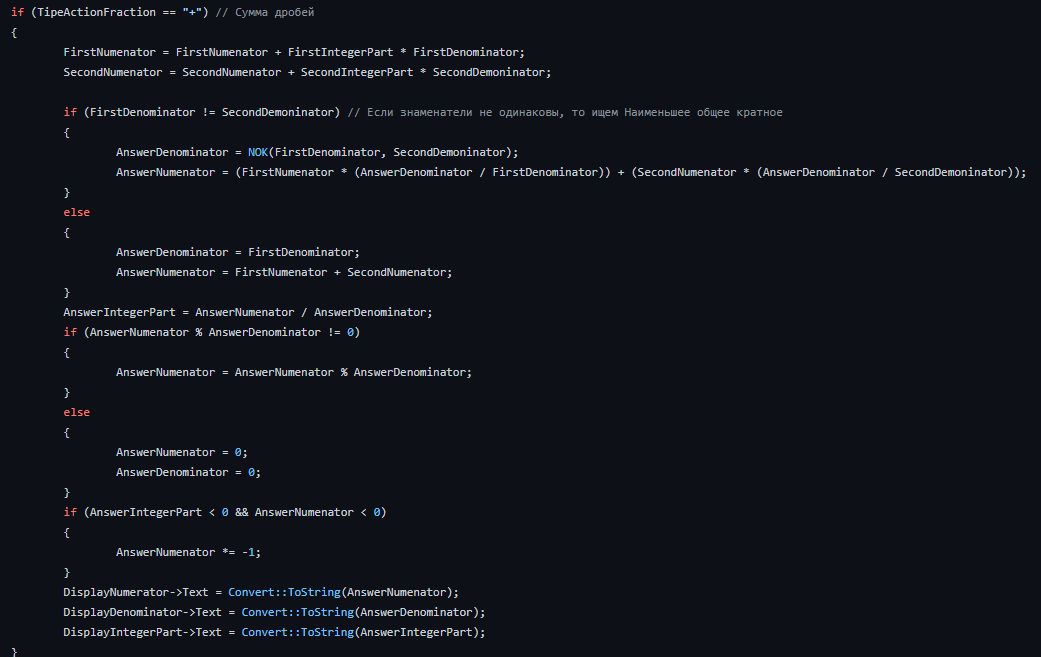
  
  
Первым делом избавляемся от целой части дроби, путем внесения её в числитель. Затем проверяем, одинаковы ли знаменатели, если нет, то ищем НОК, если да, то просто суммируем числители. В конце сложения нужно избавиться от неправильной дроби, это мы делаем с помощью целой части дроби, путем деления нацело. Дальше нужно избавиться от этого значения в числителе, используем деления с остатком. Весь этот результат потом выводится на 3 дисплеев.

Рисунок 14 – Алгоритм сложения в функции button37\_Click (=)

Алгоритм вычитания работает по такому же принципу.

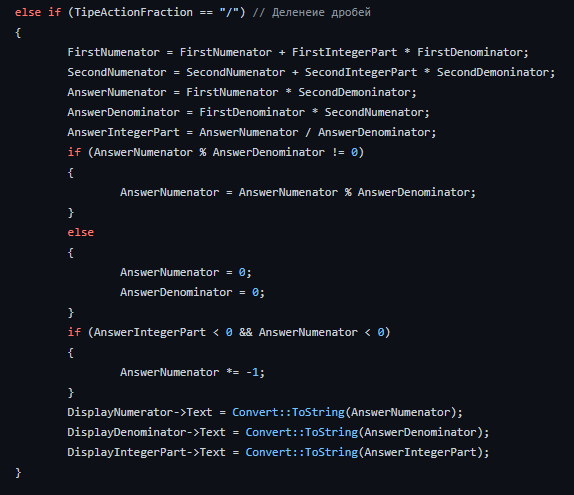
  
  
Деление дробей это тоже самое умножение, но нужно найти произведения первой дроби и обратной второй дроби. Это мы выполняем, просто перемножая FirstNumenator и SecondDemoninator, то есть числителя первой дроби и знаменателя второй.

Рисунок 14 – Алгоритм деления в функции button37\_Click (=)

Дизайн:

Я выбрал такой дизайн калькулятора, так мне нравиться более простой внешний вид программы, он очень похож на самый обычный калькулятор, и поэтому, чтобы начать работу с ним не нужно тратить время на его освоение. Так же хочу выделить интерфейс калькулятора обыкновенных дробей, в нем сразу же интуитивное понятно, за что отвечают дисплеи, так как находятся в привычных местах.

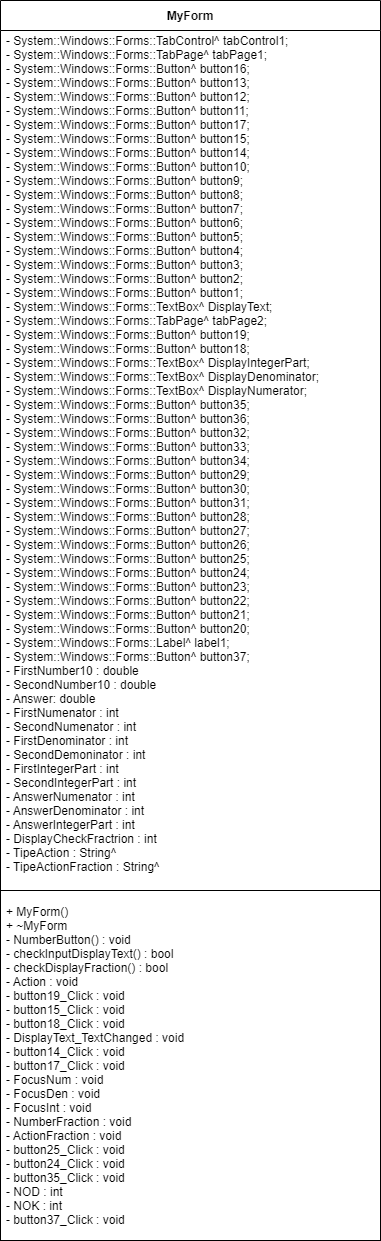
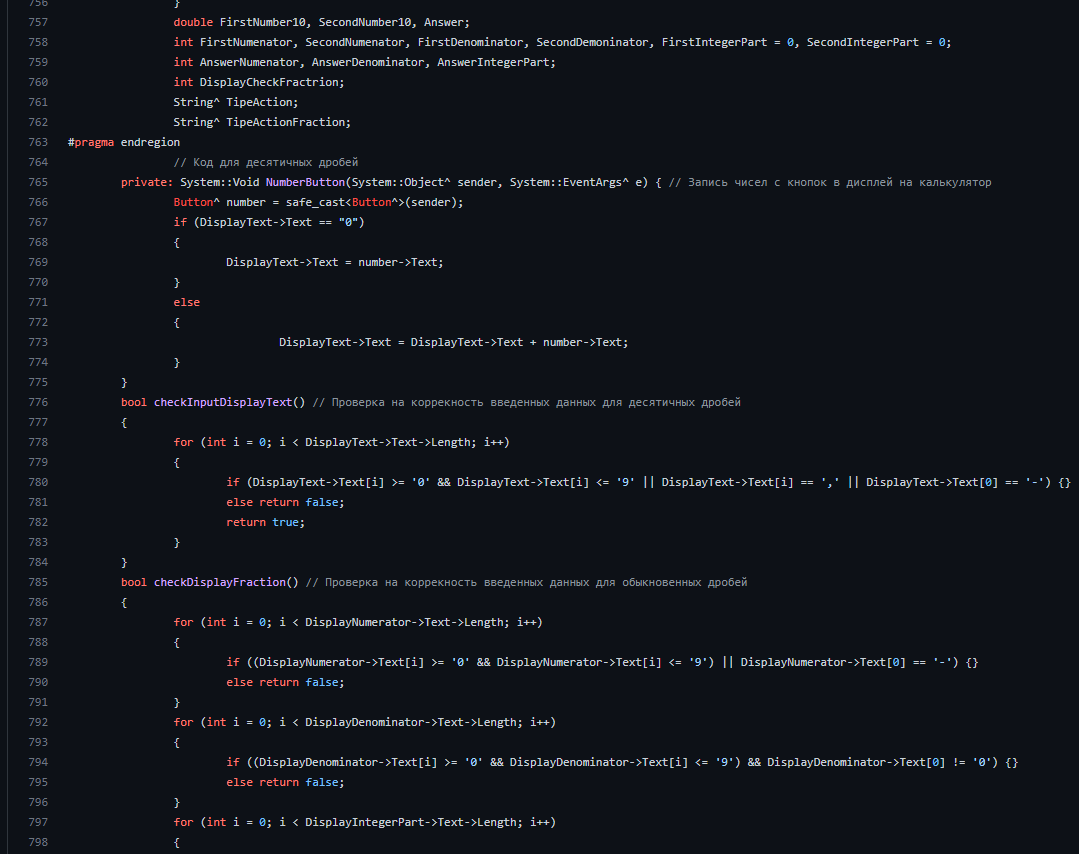
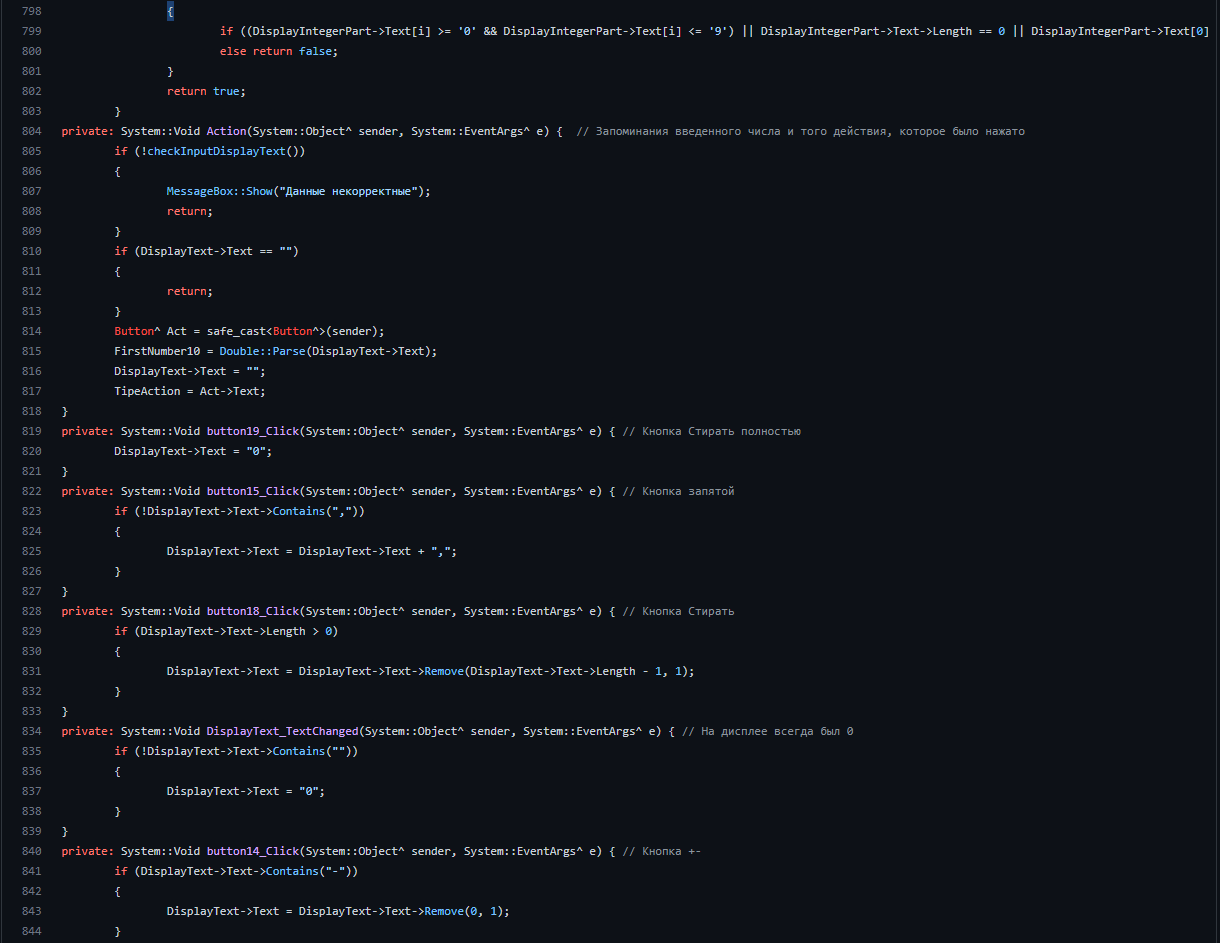
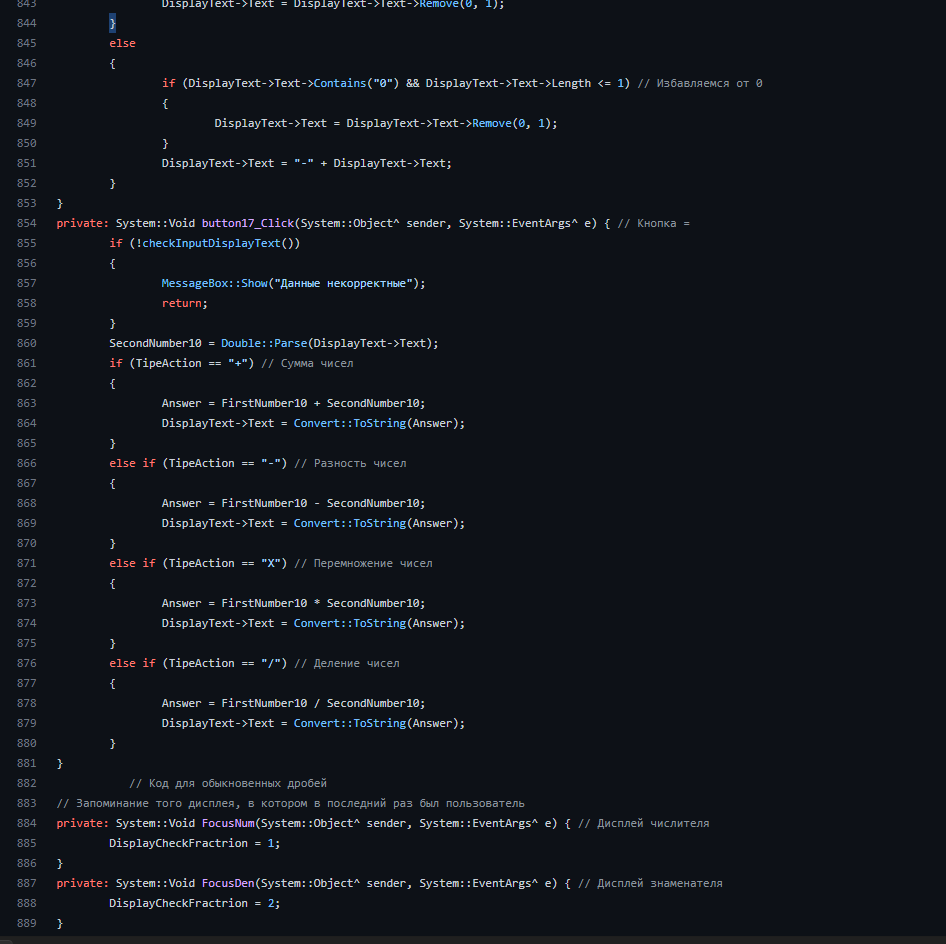
**Диаграмма класса**

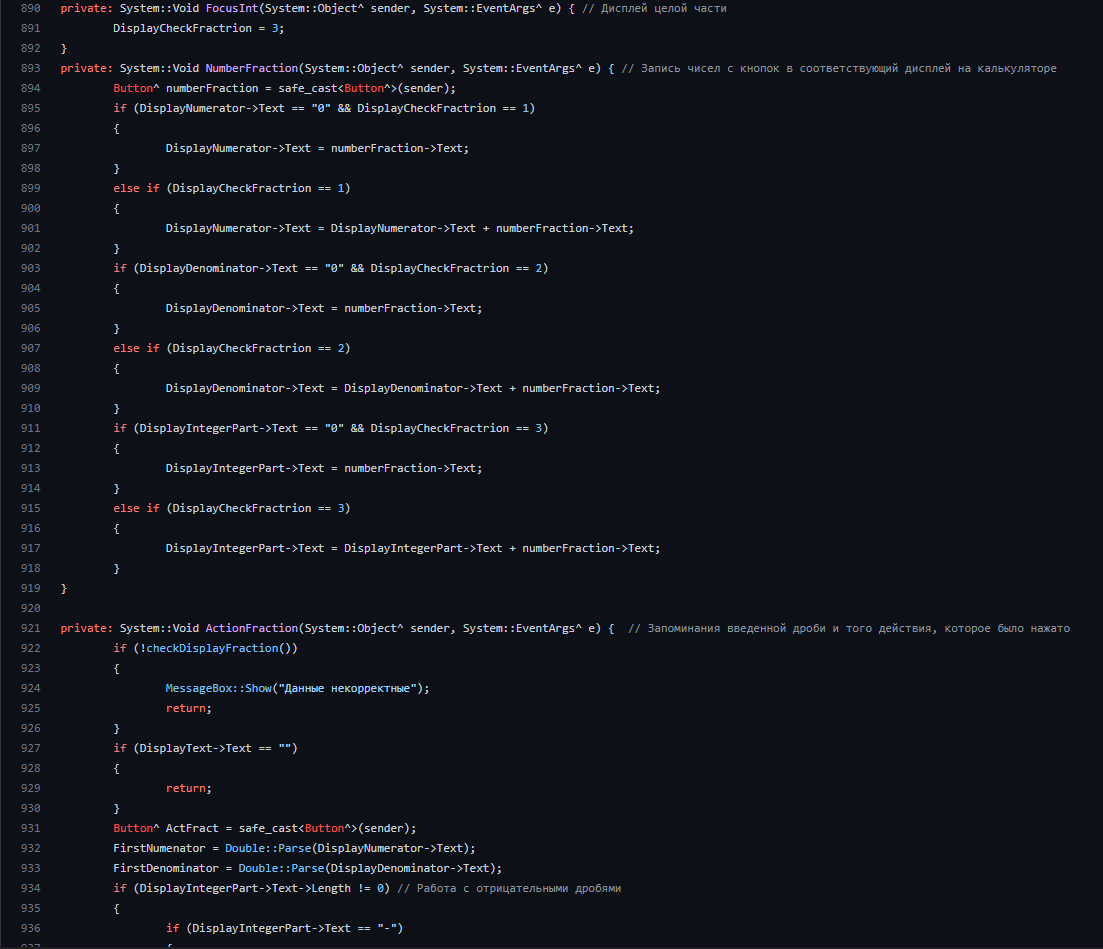
Рисунок 15 – Диаграмма классов Калькулятора

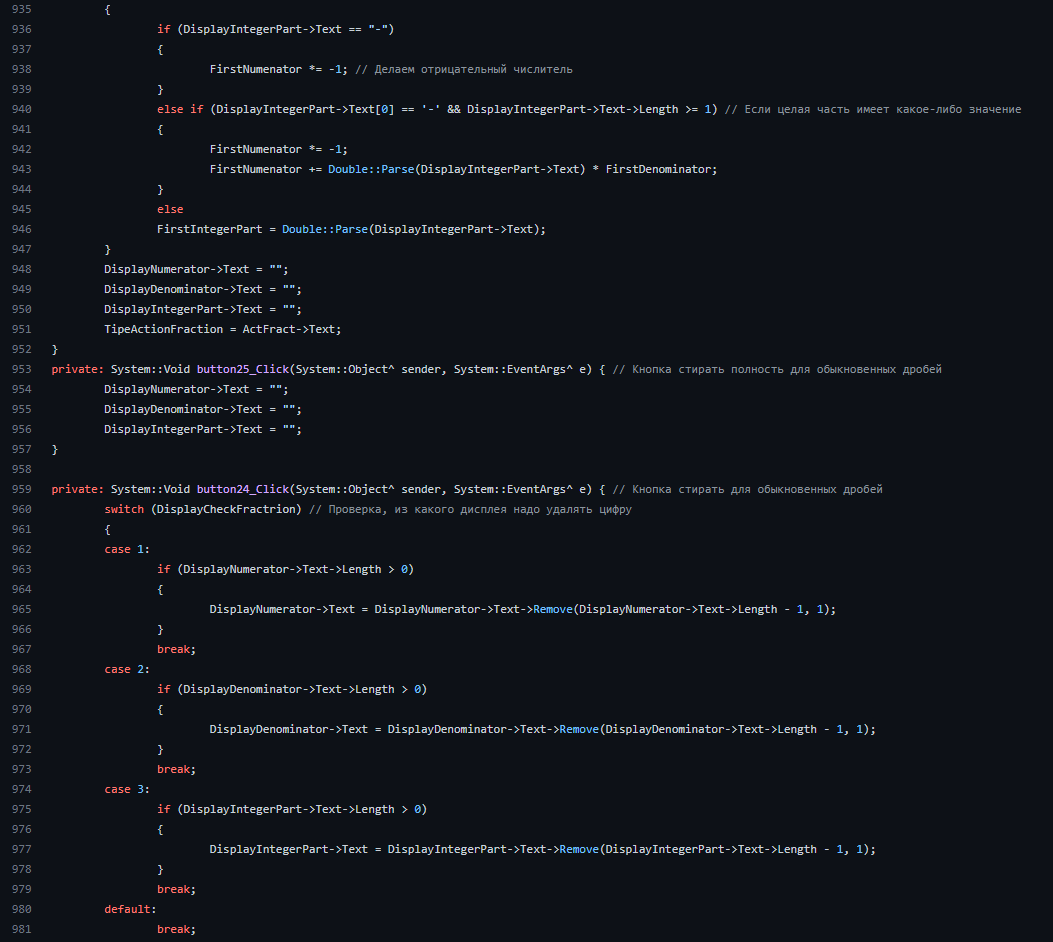
**Код программы**

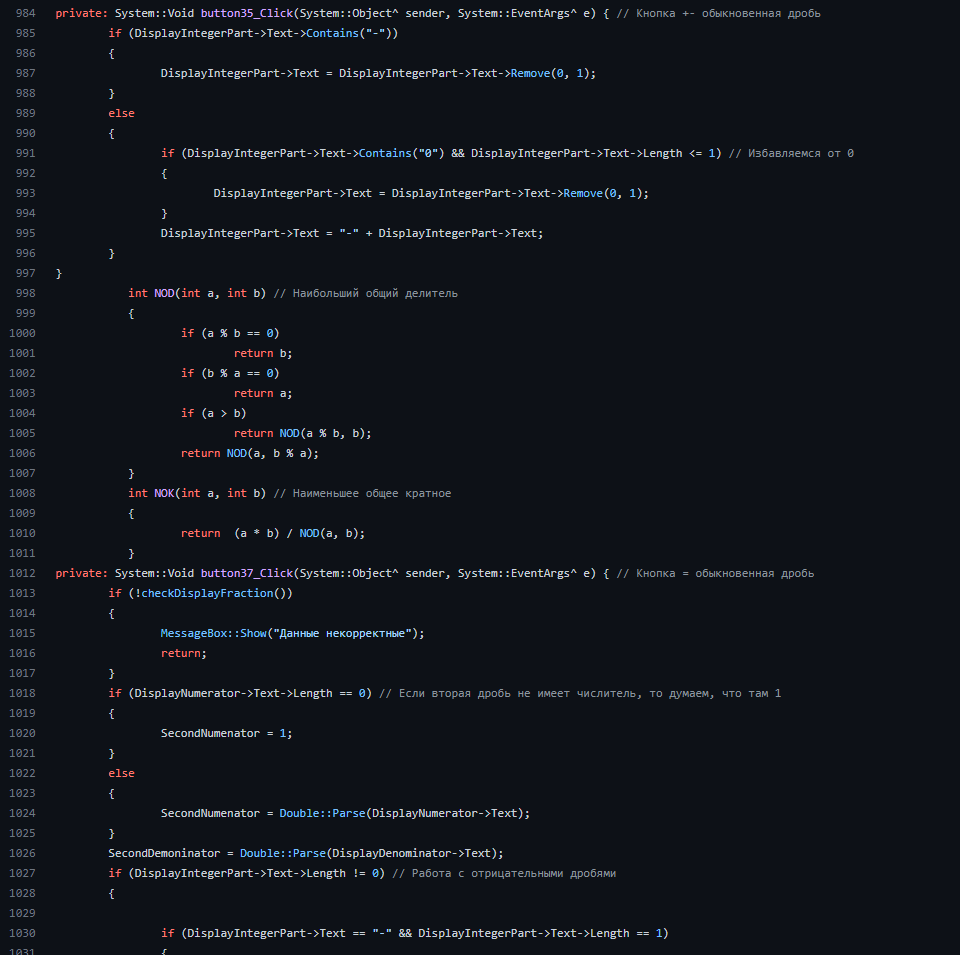
****

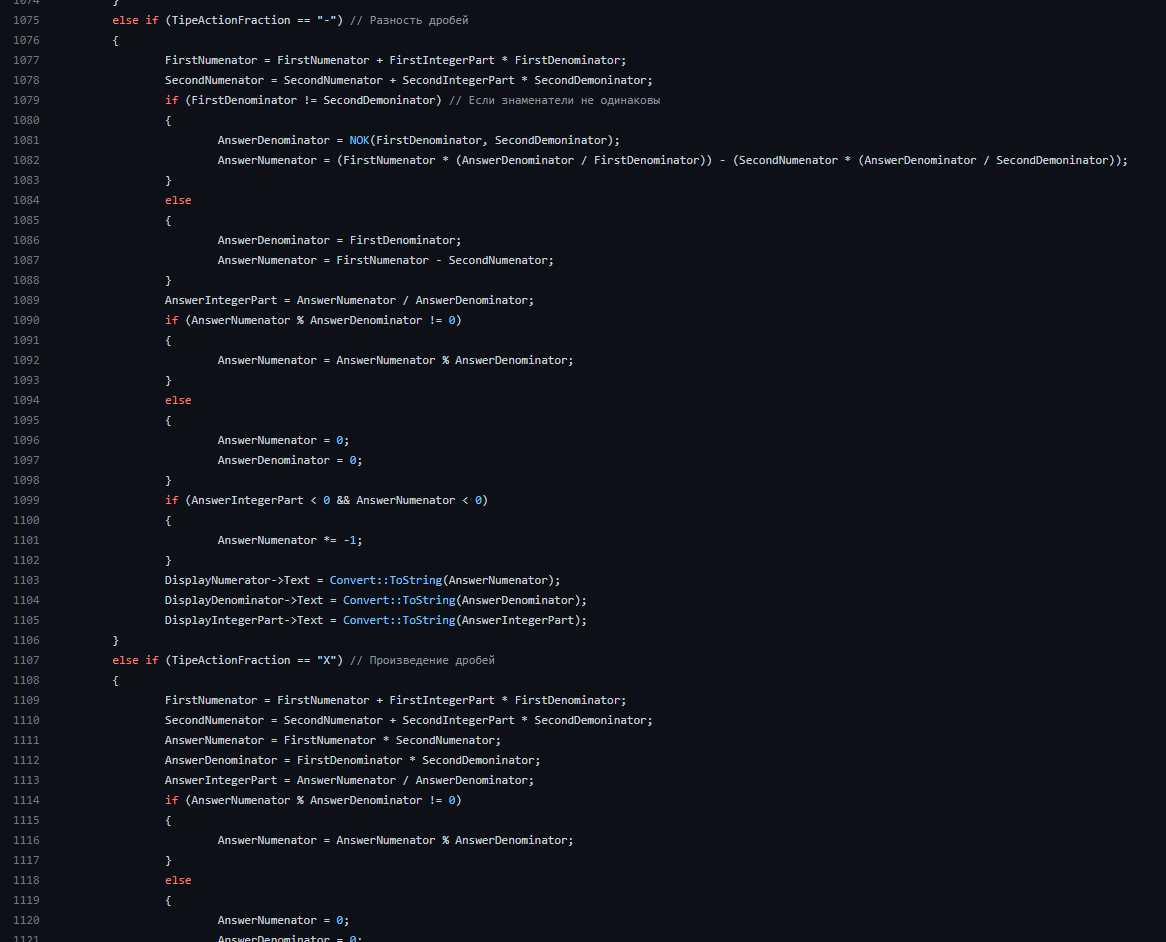
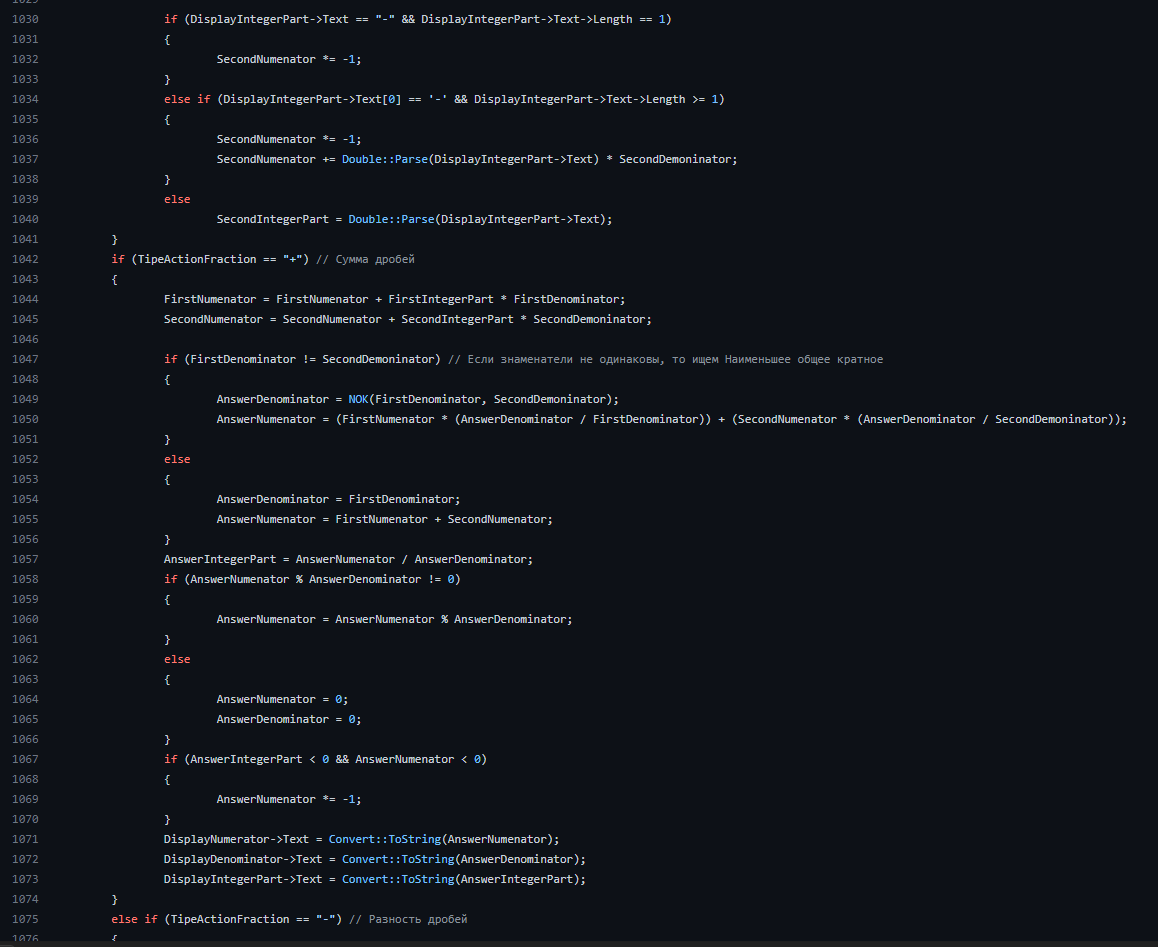
****

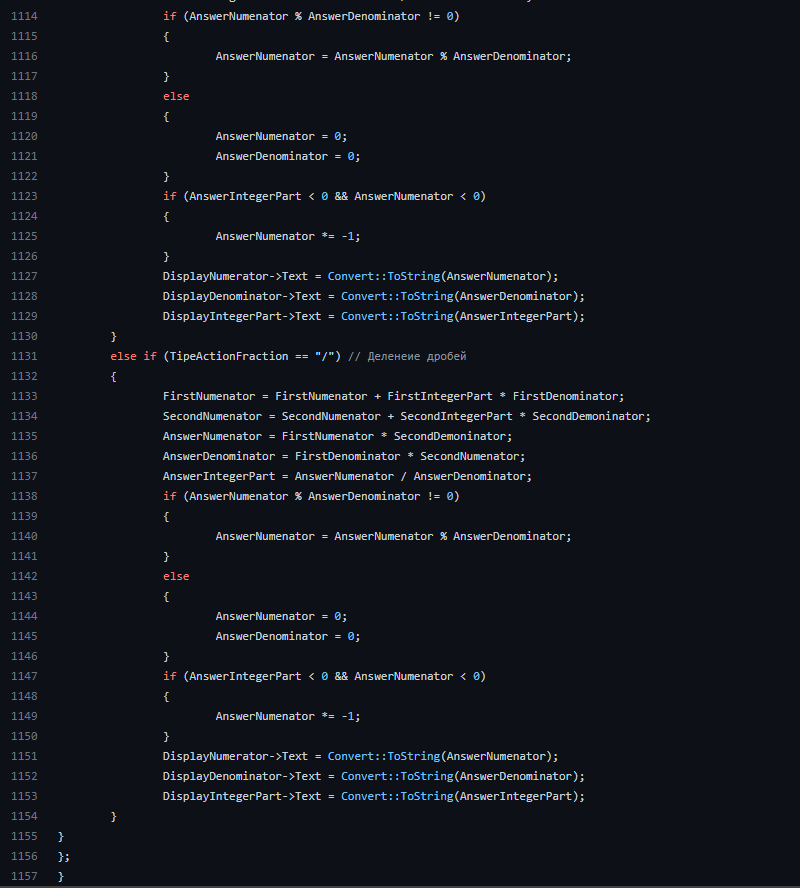
****

****

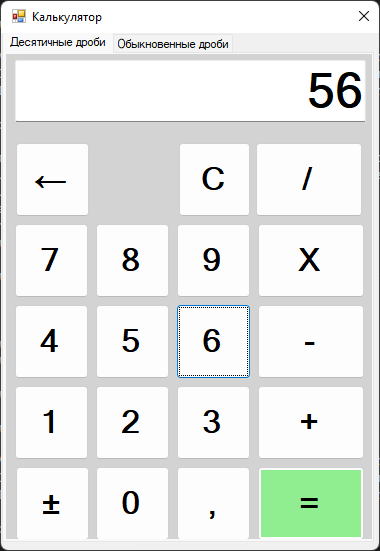
****

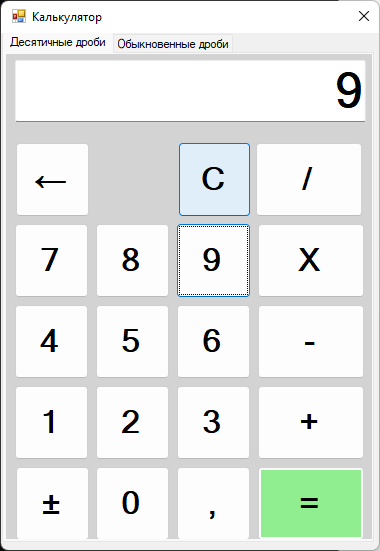
****

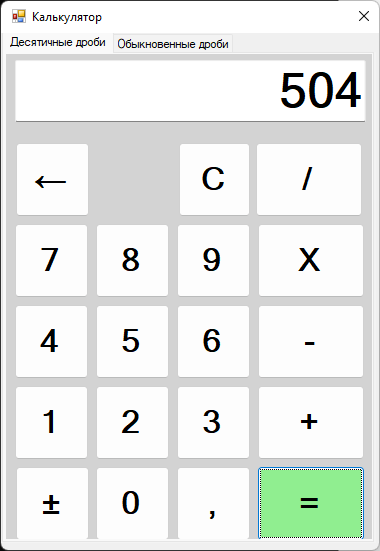
****

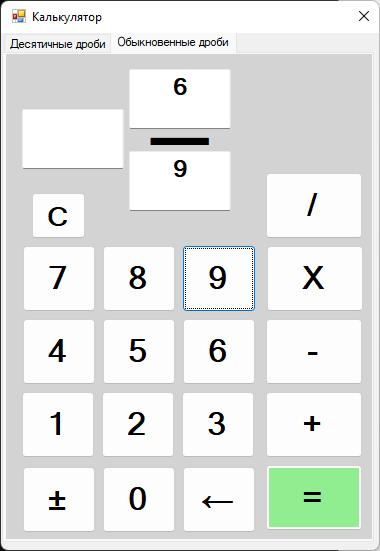
****

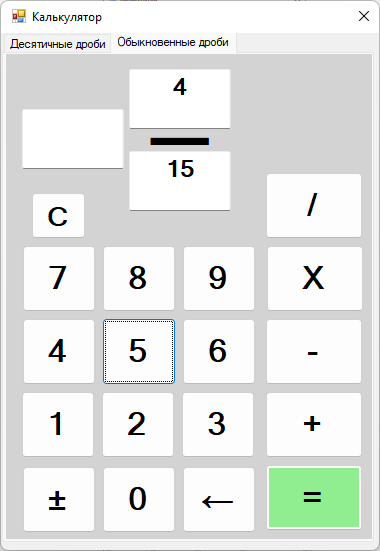
**Работа программы:**

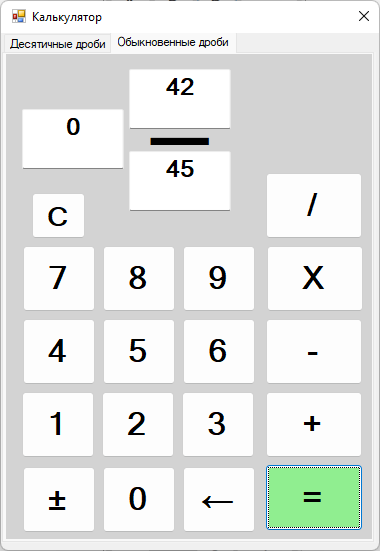
****

****

****

****

****

****

**Задача Коммивояжера**

**Постановка задачи:**

1. Реализовать решение задачи Коммивояжера методом Ветвей и Границ.
2. Визуализировать граф.
3. Добавить функционал добавления и удаления вершины.

**Описание решения:**

Для визуализации использовался инструмент OpenGL.

Трудности:

Как именно можно визуализировать граф. Но путем просмотра информации в интернете был найден способ визуализации с помощью окружности, а именно постановки вершин графа на эту самую окружность.

Работа с OpenGL. Как именно его устанавливать (Были некоторые проблемы с этим, но в последствии был найден способ все это решить), как работать с фигурами, координатами, цветами. Путем проб и ошибок разобрался со всем этим просто практикуясь.

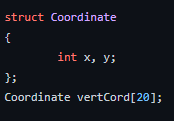
Описание кода:

Рисунок 16 – Структура Coordinate

Данная структура нужна для удобства работы, так как для OpenGL нужны координаты.



Рисунок 17 – Класс Graph

Граф выполнен через классы, функционал был взят из соответствующей лабораторной работы.



Рисунок 17 – Методы класса Graph

Метод InsertEdge – Добавить ребро между двумя вершинами.

InsertVertex – Добавить вершину.

DeleteVertex – Удалить последнюю вершину.



Рисунок 18 – Методы класса Graph

GetVertPos – Возвращает порядковый номер вершины.

GetNbrs – Возвращает список соседей этой вершины.

GetAmountVerts – Возвращает текущее кол-во вершин.

GetAmountEdges – Возвращает текущее кол-во рёбер.



Рисунок 19 – Методы класса Graph

GetWeight – Возвращает вес ребра.

IsEmpty – Возвращает пуст ли граф или нет.

IsFull – Возвращает заполнен ли граф до конца или нет.

Print – Выводит матрицу смежности.

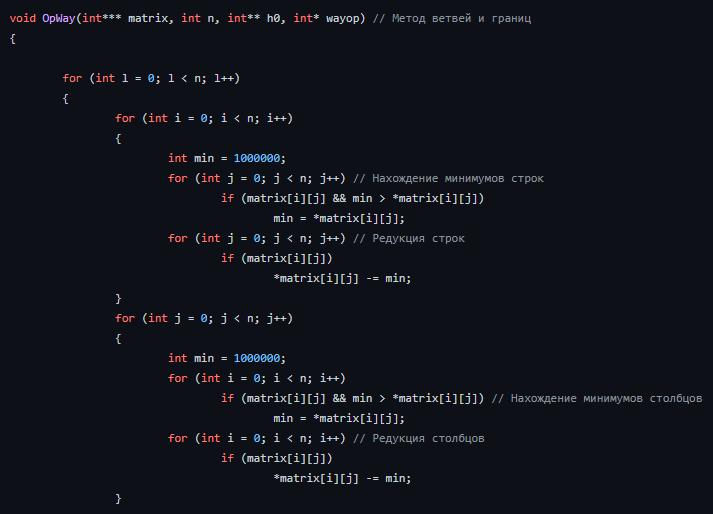


Рисунок 20 – Метод Ветвей и границ

Метод для решения задачи Коммивояжера это метод Ветвей и Границ. Поэтому функция OpWay нужна для этого метода. Сперва нужно найти минимумов по строкам, затем идет редукция этих строк. Следующий шаг — это нахождение минимумов по столбцам, и последующая редукция столбцов.

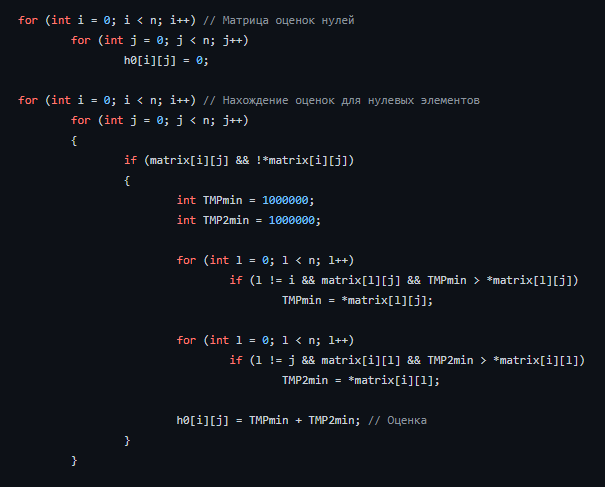


Рисунок 21 – Метод Ветвей и границ

Матрица h0 нужна для записи оценок нулевых элементов.   
Следующий шаг – это нахождение оценок для нулевых элементов и последующая запись оценок в матрицу h0.

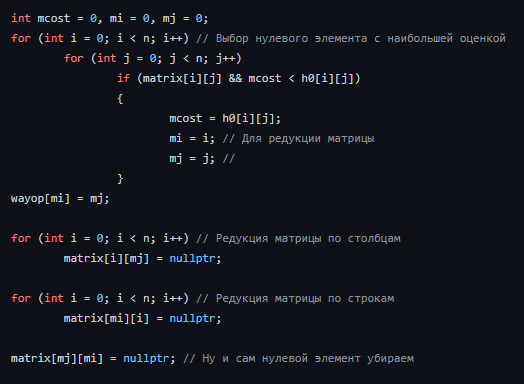


Рисунок 22 – Метод Ветвей и границ

Выбор нулевого элемента с наибольшей оценкой, запоминания столбца с помощью переменной wayop. Затем идет редукция матрицы.

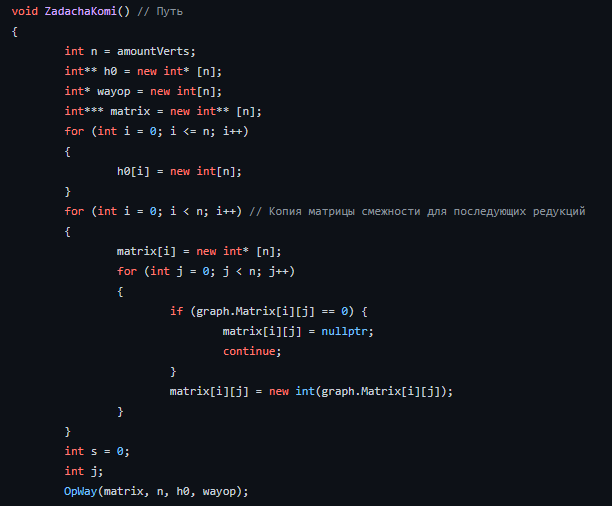


Рисунок 23 – функция ZadachaKomi

Функция ZadachaKomi это ответ на задачу Коммивояжера, сперва мы делаем копию изначальной матрицы смежности и потом отправляем её в функцию OpWay.

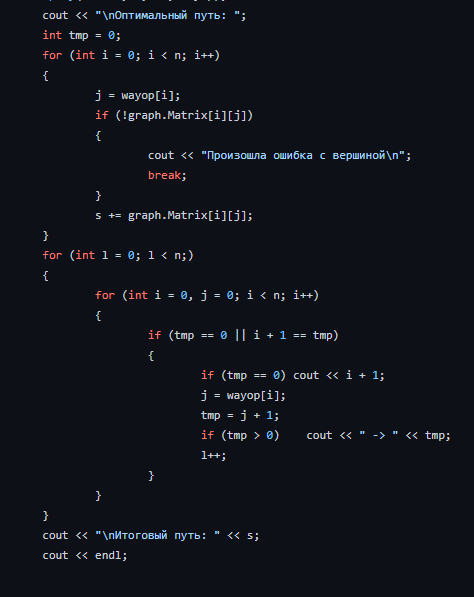


Рисунок 24 – функция ZadachaKomi

Идет подсчет расстояния пройденного пути и вывод его на экран, вместе с путём.

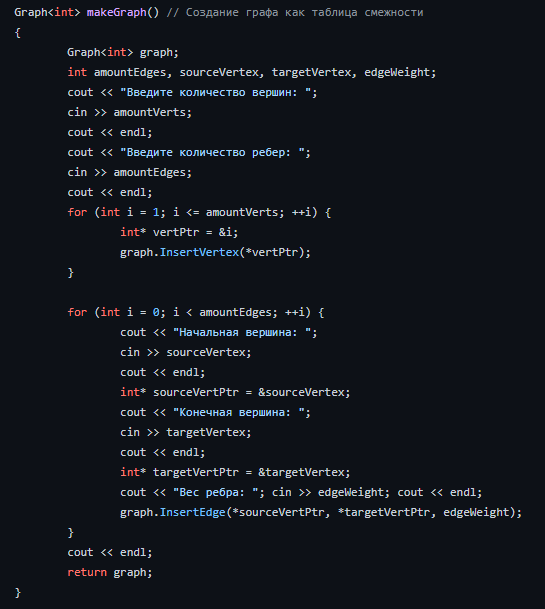


Рисунок 25 – метод makeGraph

Метод makeGraph нужен для создания матрицы смежности, и этот метод запускается при старте программы.



Рисунок 26 – функция GraphCircle

Один из способов отображения графа. Выполнен с помощью уравнения, благодаря которым мы можем получить точки на окружности, и эти точки являются вершинами графа.

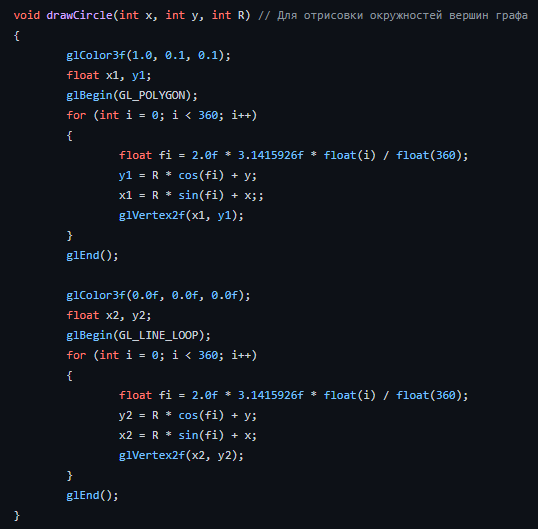


Рисунок 26 – функция drawCircle

Функция drawCircle – отрисовка окружностей вершин графа.

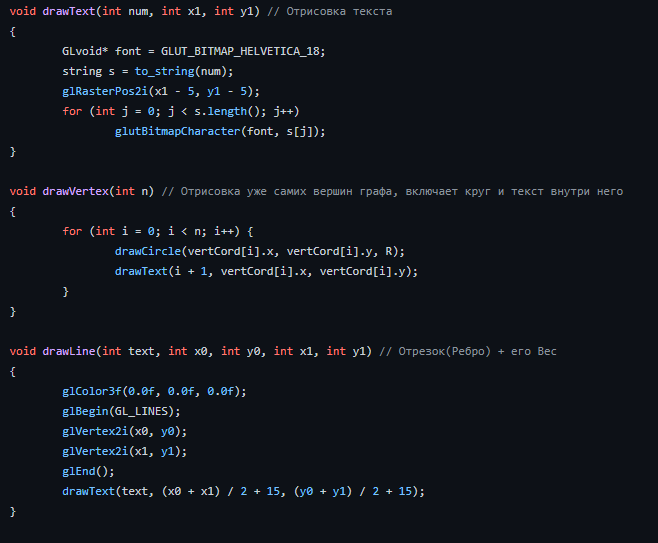


Рисунок 27 – функция drawText, drawVertex, drawLine

drawText – вывод текста для вершин графа.

drawVertex – Включает в себя две другие функции для отрисовки вершин графа.

drawLine – Отрисовка рёбер графа вместе с их весом.

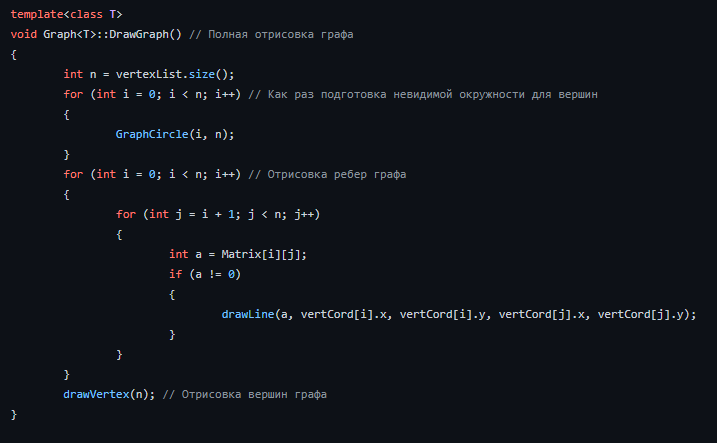
  
  
Метод для полной отрисовки графа, включает подготовку координат для вершин графа, отрисовка рёбер графа и вершин этого графа.

Рисунок 28 – метод DrawGraph

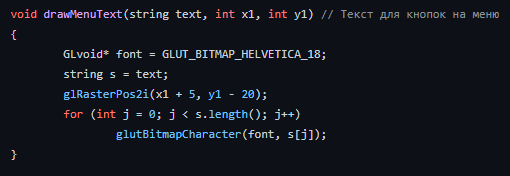
  
  
Функция drawMenuText – нужна для отрисовки текста кнопок меню.

Рисунок 29 – функция drawMenuText

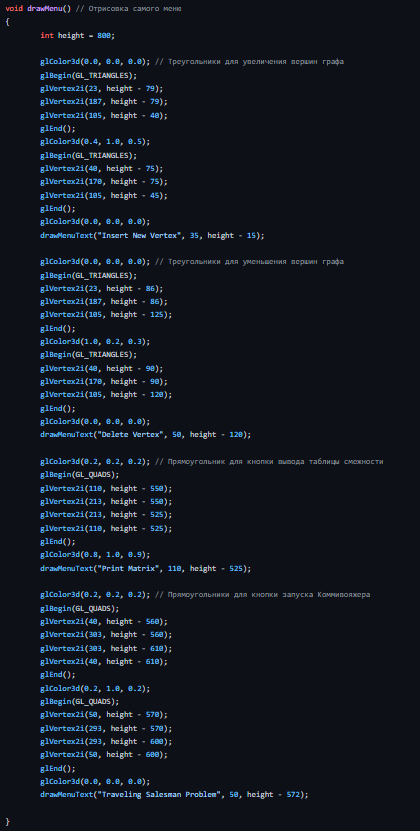


Рисунок 30 – функция drawMenu

Полная отрисовка меню, включает в себя 4 треугольника и 3 прямоугольника. В итоге получаем 4 кнопки для добавления и удаления вершин, вывода матрицы смежности и решения задачи Коммивояжера.

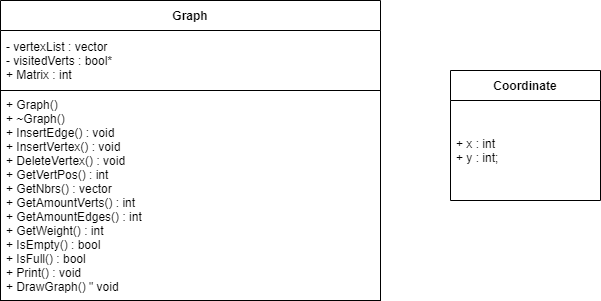
  
  
Функция MouseAction нужна для отслеживания нажатий пользователя. По определенным координатам, которые почти совпадают с координатами фигур в меню, происходит тот алгоритм, который написан на кнопках в меню.

Рисунок 31 – функция MouseAction

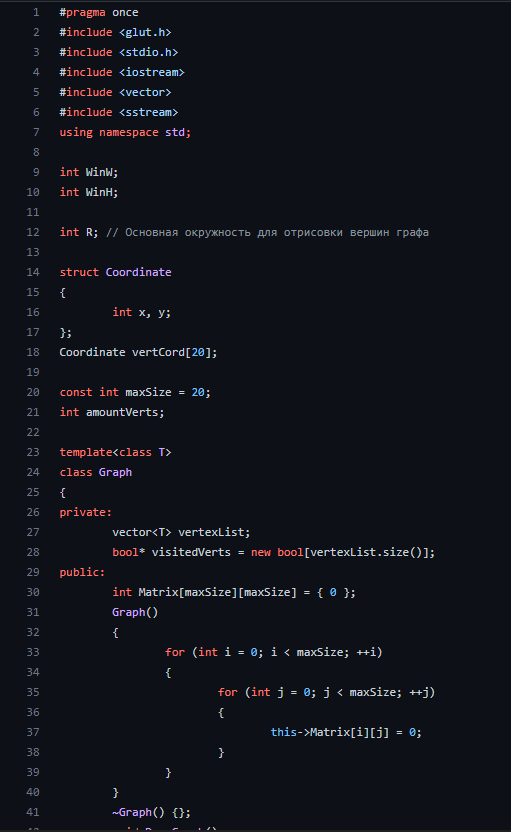
Дизайн:

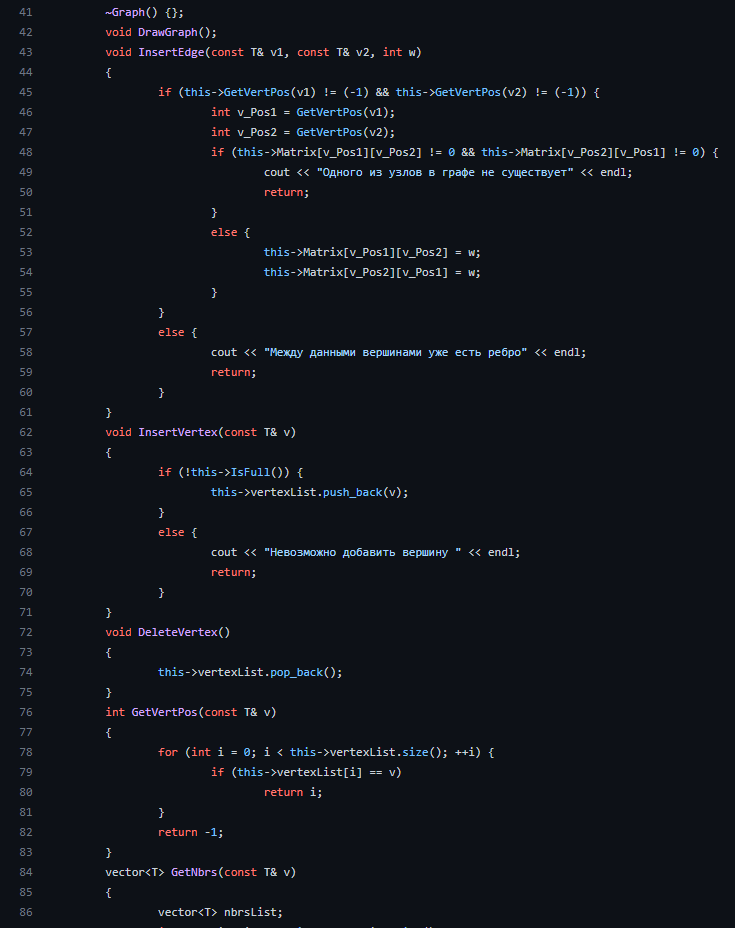
Мне понравился метод с отрисовкой вершин графа с помощью окружности, поэтому я использовал его в своей работе. Фон выбран был серый, так он наименее всего бросается в глаза, не стал оставлять белые, чтобы фон выделялся. Кнопки добавления и удаления были сделаны в виде треугольников, или можно сказать «стрелок», которые соответственно показывают, что они делают. Так же это можно сказать по их цвету, зеленый – добавить, красный – убрать. Кнопка для решения задачи Коммивояжера выделяется.

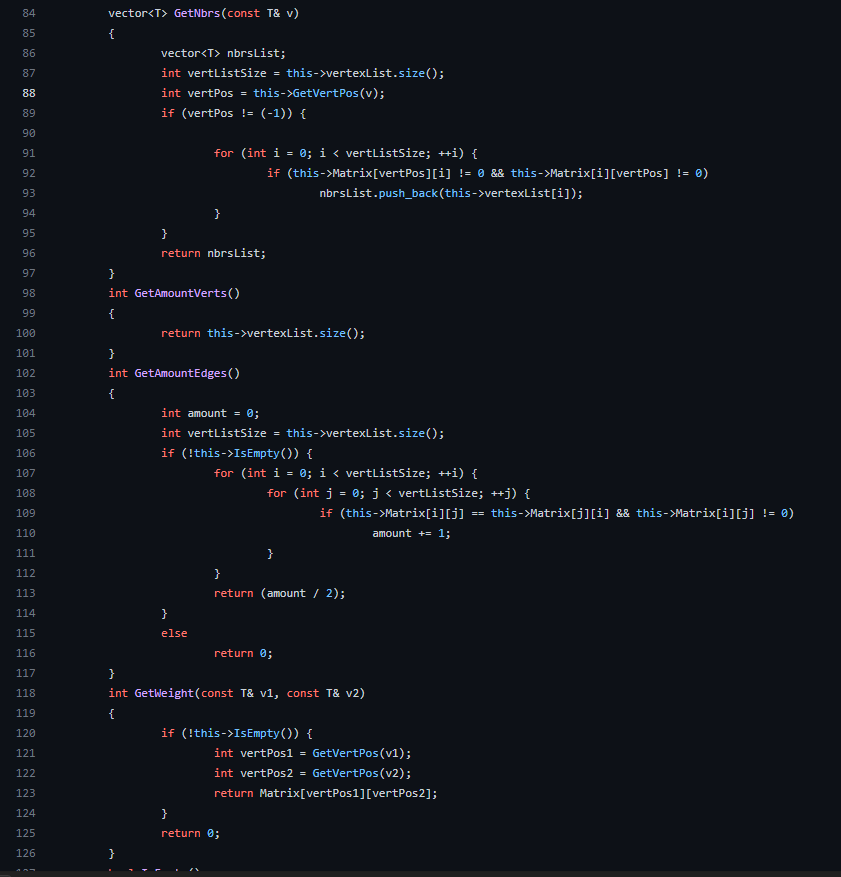
**Диаграмма класса**

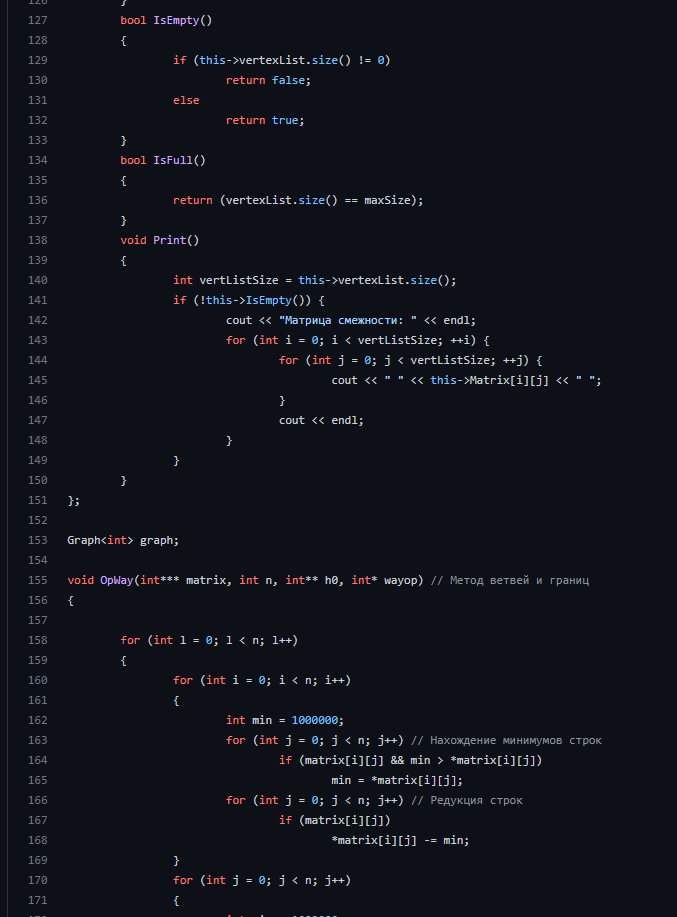
****

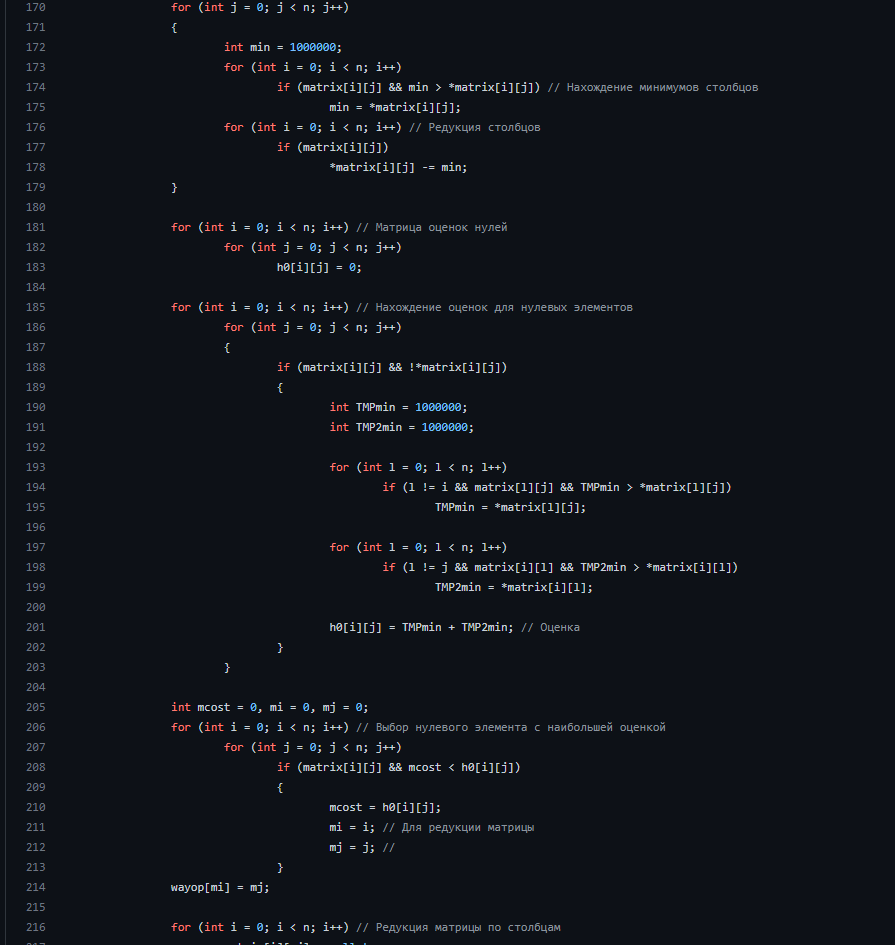
**Код программы**

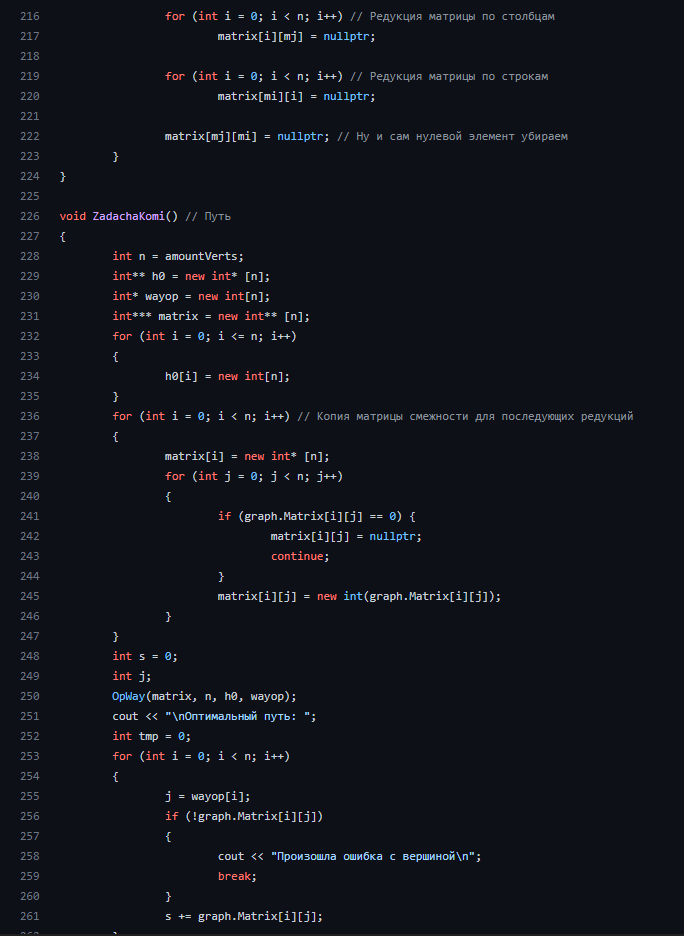
****

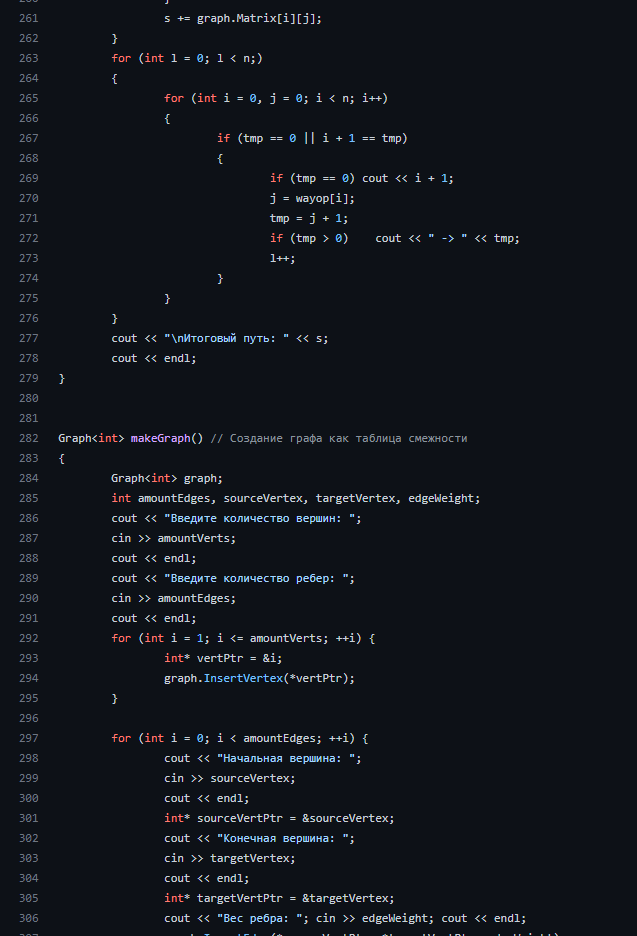
****

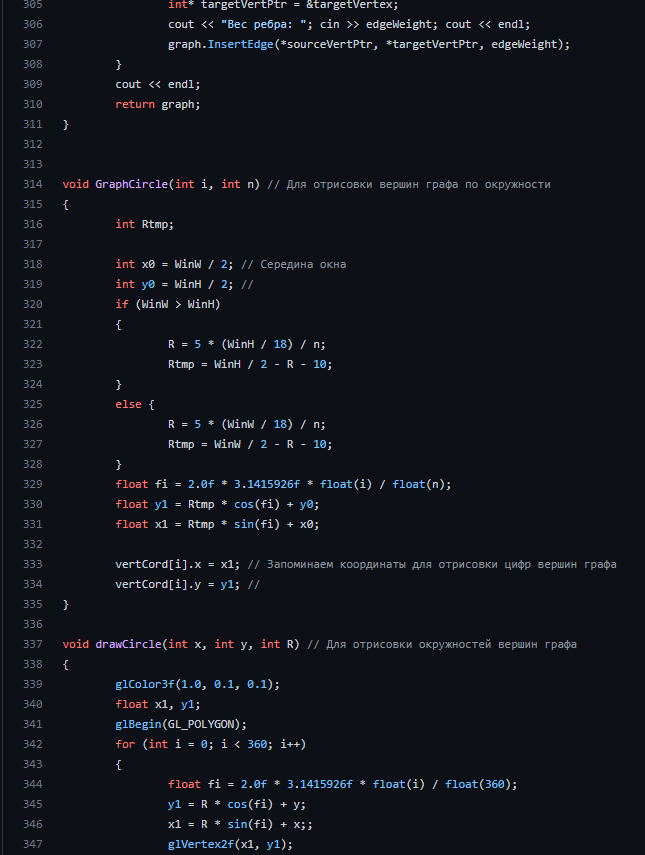
****

****

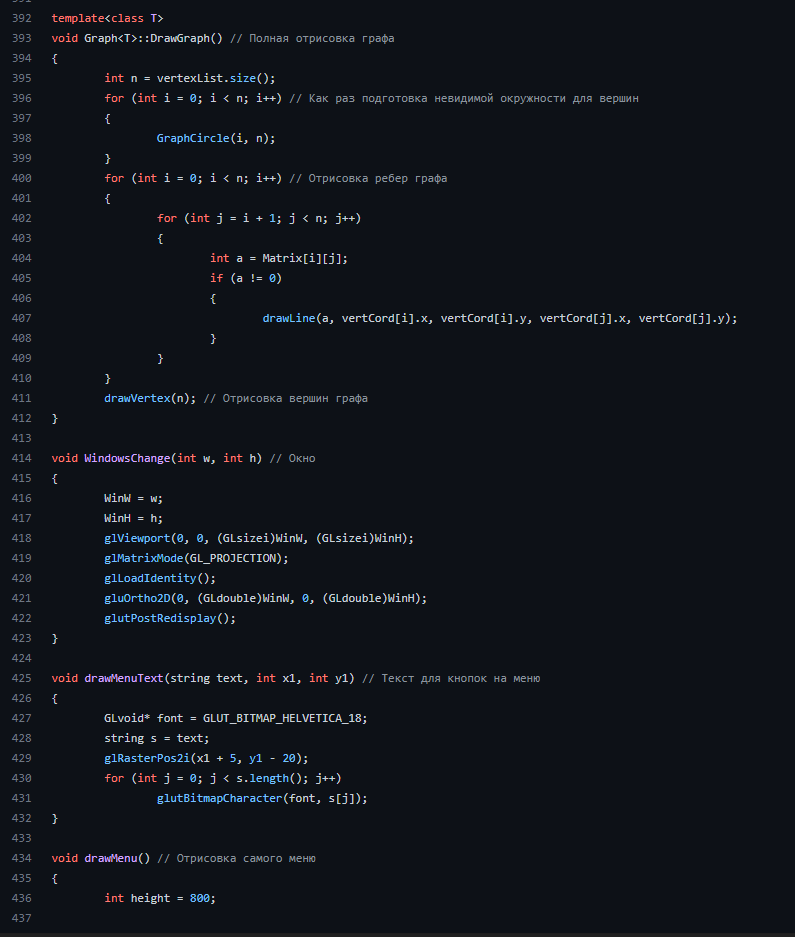
****

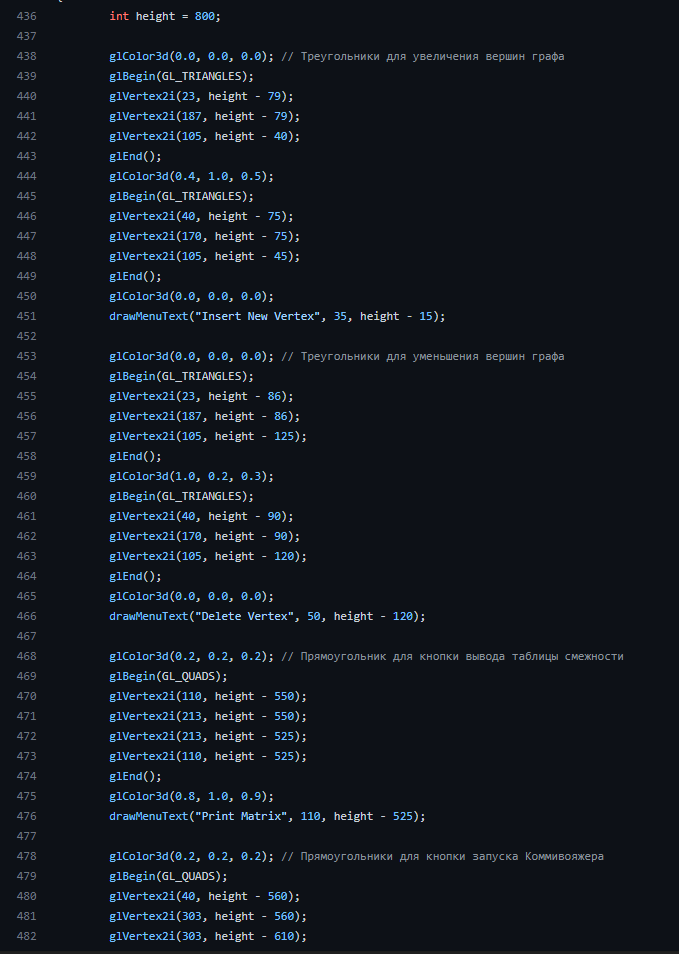
****

****

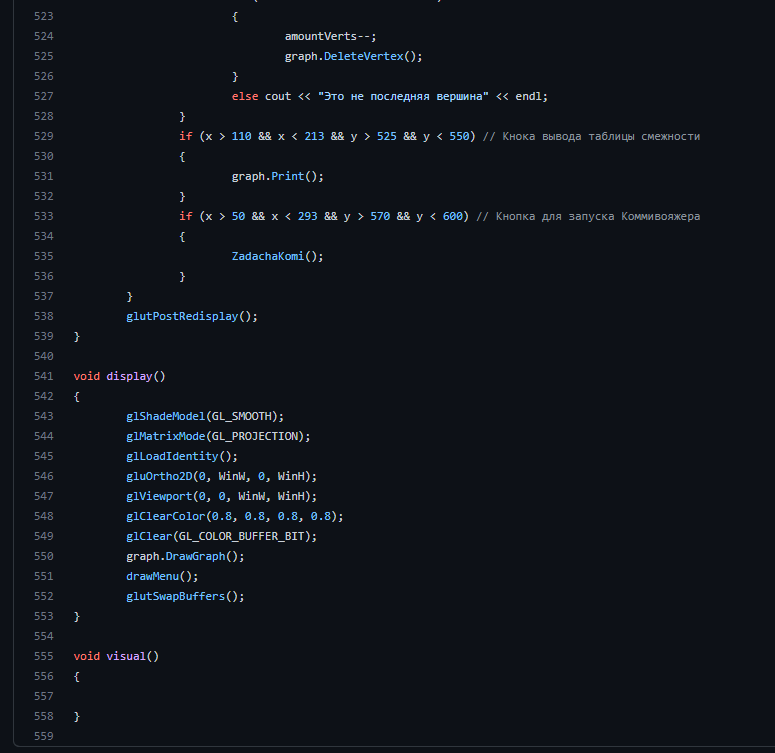






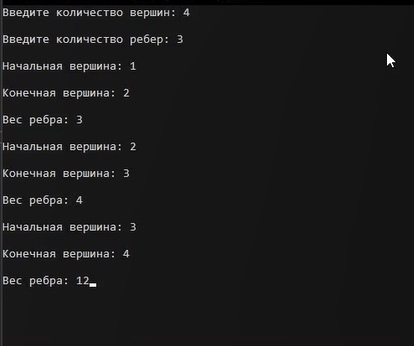


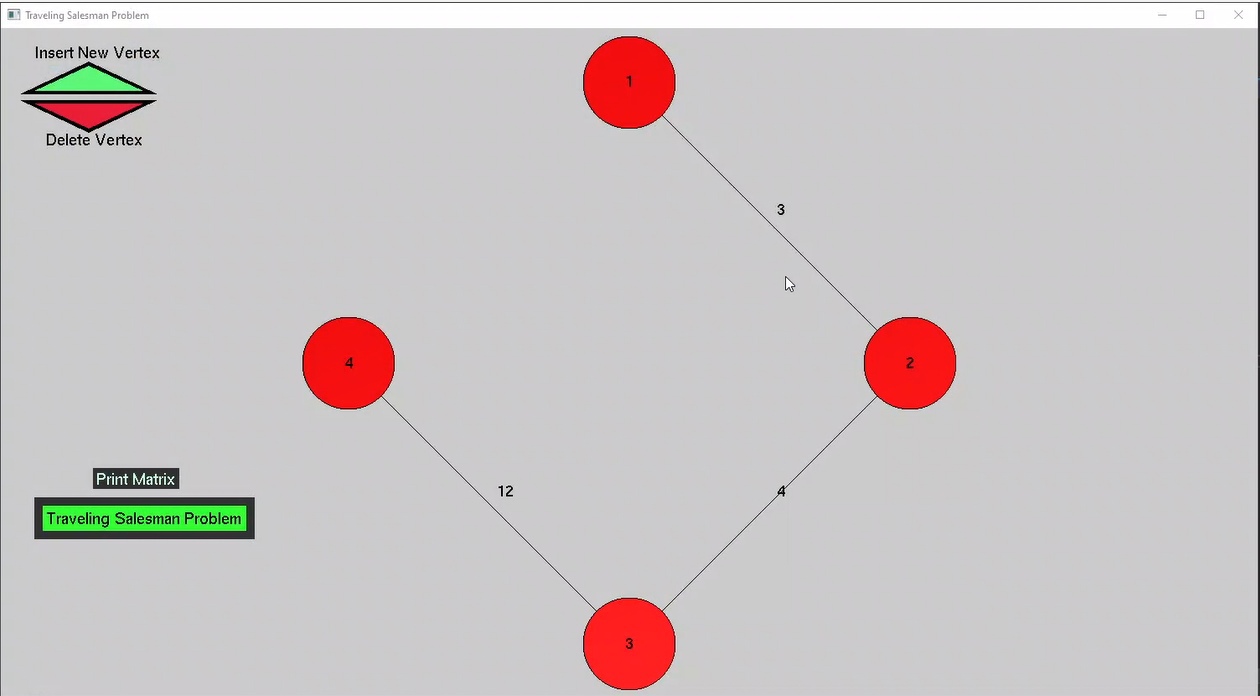


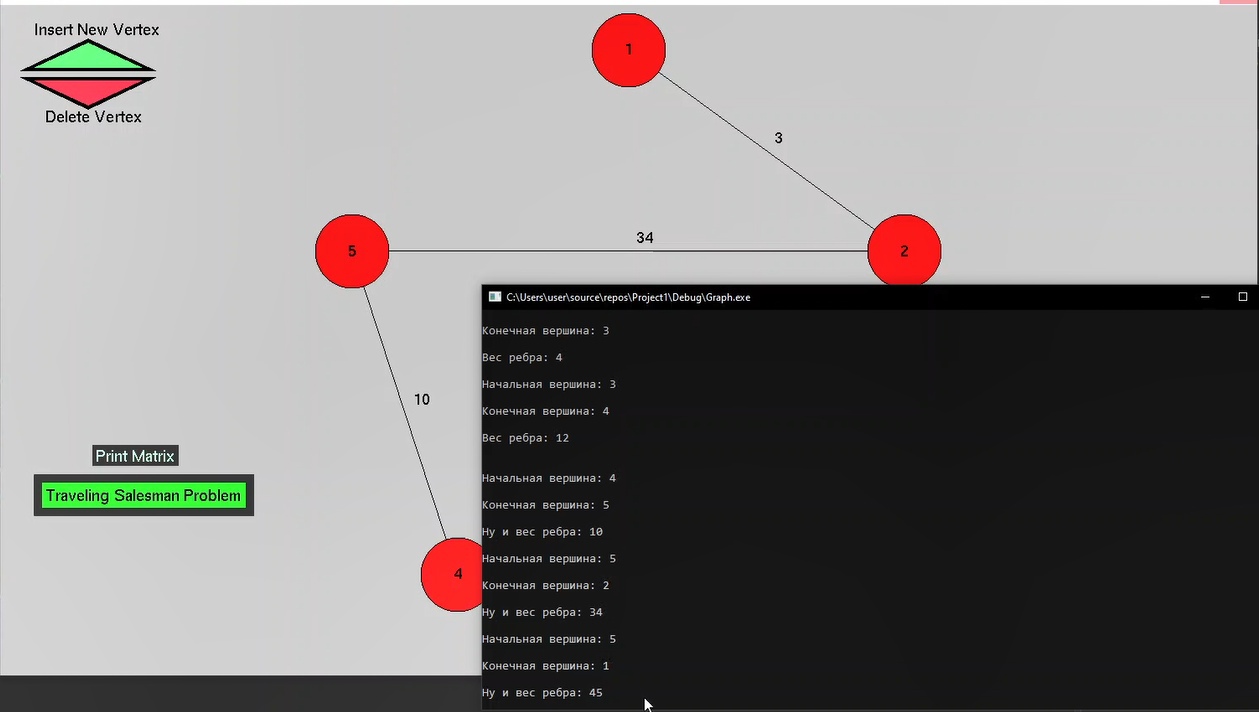


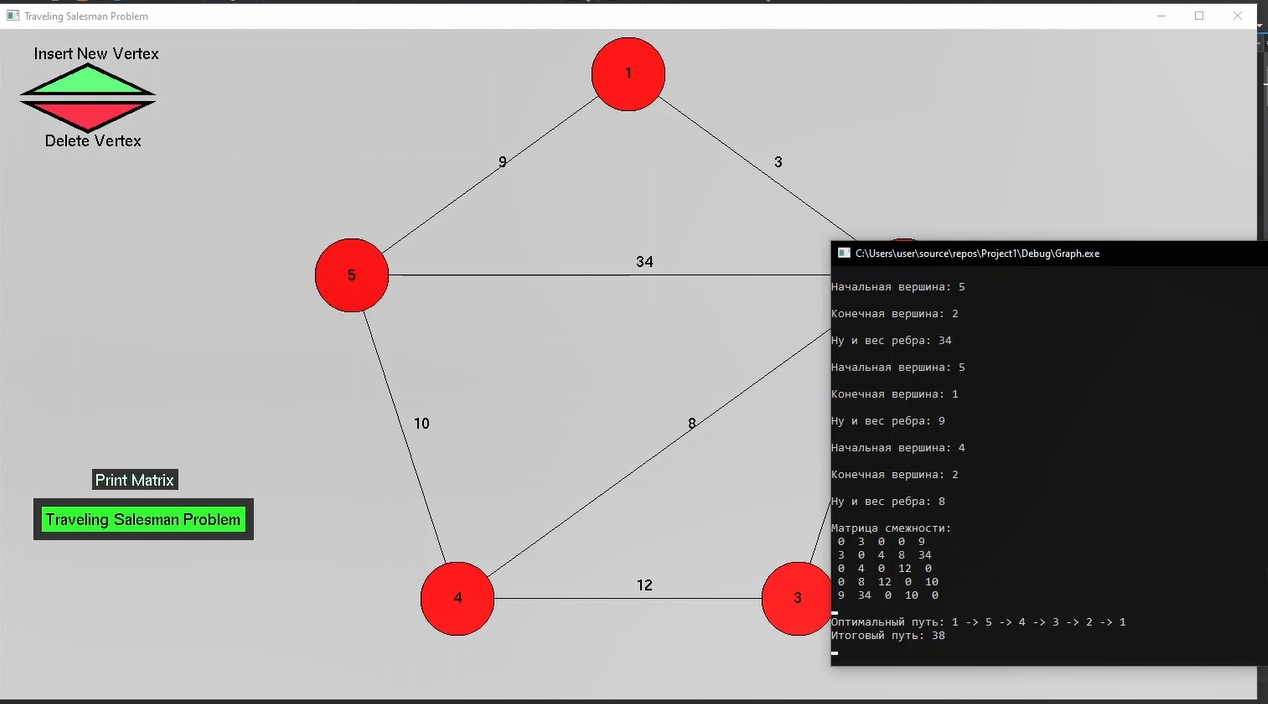


**Работа программы**









# **Ссылки**

Youtube - <https://www.youtube.com/watch?v=jbT_Fj4HqG0>

GitHub - <https://github.com/FarWG/TvorcheskayRabota>