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 “Коммивояжер и Калькулятор градусов”**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

Выполнил студент гр. ИВТ-21-1б

Волков Роман Алексеевич

Проверил:

(подпись) (оценка)

г. Пермь-2022

**Содержание:**

Задача коммивояжера...................................................................................................................3

Калькулятор градусов................................................................................................................26

Ссылка на GitHub и YouTube канал..........................................................................................41

**Коммивояжер:**

**1)Постановка задачи:**

1. Реализовать ввод матрицы смежности.
2. Реализовать функции для решения задачи методом ветвей и границ.
3. Реализовать нахождение наикратчайшего пути.
4. Реализовать отрисовку графа с помощью OpenGL.

**2)Решение задачи:**

Переменные:

sourceVertex - Начальная вершина.

targetVertex - Конечная вершина.

EdgeWeight - Вес ребра.

AmountVerts,n - Кол-во вершин.

AdjMatrix - Матрица смежности для вывода и счёта пути.

inter - Матрица смежности для метода ветвей и границ.

helpinter - Вспомогательная матрица.

path - Матрица для хранения пути.

vertC - переменная для хранения координат вершин.

string Text - для введения текста в вершинах и меню.

vertList - переменная вектор.

Также есть подключённая библиотека OpenGL <glut.h> для возможности отрисовки графа.

1) Класс Графа

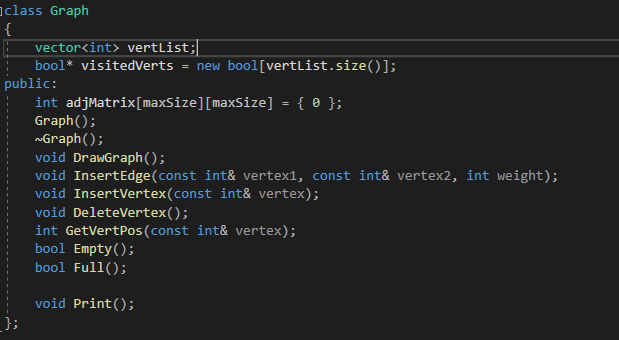


Рис. 1

2) Функции вставки и удаления вершин через библиотеку <vector>

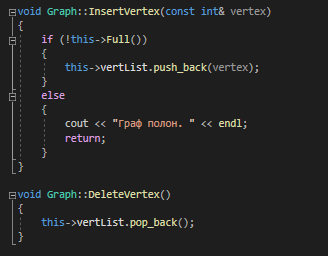


Рис. 2

3) Функция вставки ребра

Вставка происходит если вершины существуют и ребра ещё нет. Сначала берутся координаты вершин, а дальше идёт проверка на наличие ребра между вершинами, если же его нет, то происходит вставка веса ребра в матрицу смежности.

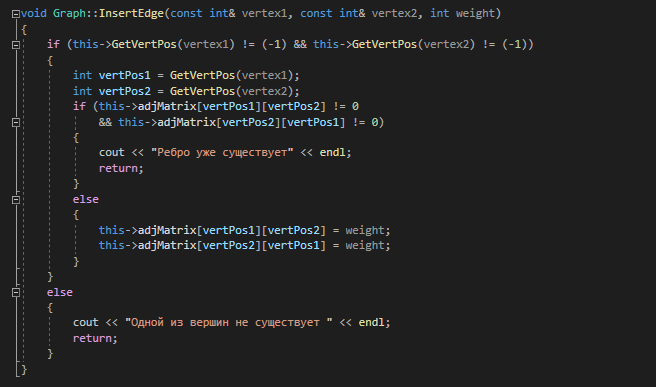


Рис. 3

4) Функции проверки пустоты и наполненности графа

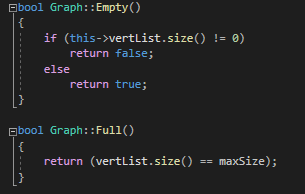


Рис. 4

5) Функция вывода матрицы смежности

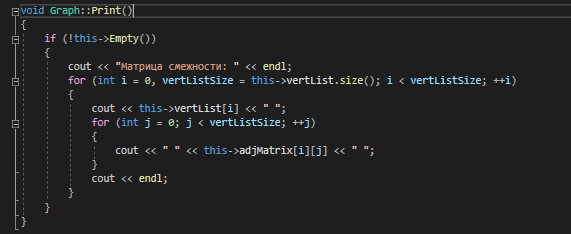


Рис. 5

Если граф не пустой, то происходит вывод матрицы

6) Функции создания графа

На ввод идёт кол-во вершин и рёбер, а дальше происходит их связывание.

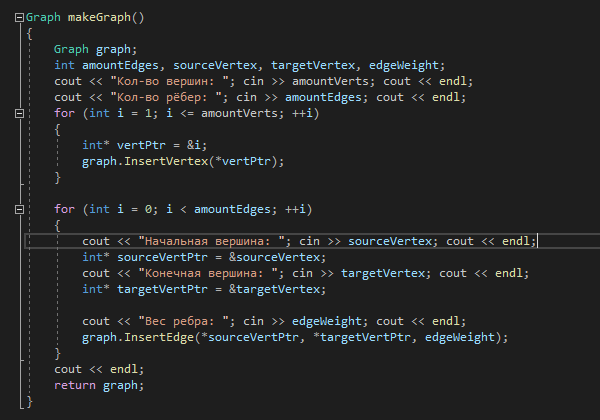


Рис. 6

7) Структура для координат вершин

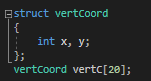


Рис. 7

8) Реализовать метод ветвей и границ

Все комментарии по методу находятся в пункте 4

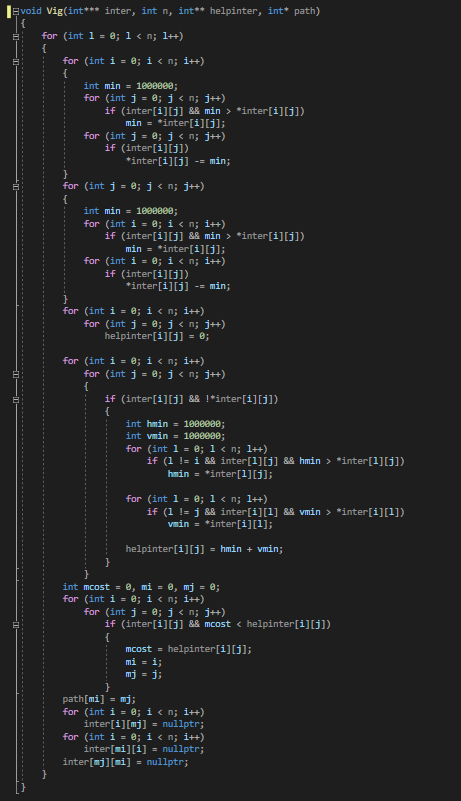


Рис. 8

9) Функция нахождения и вывода наикратчайшего пути

Переменная s хранит значение минимального пути

Путь выводится в формате “Номер начального пункта” - > “Номер конечного пункта”

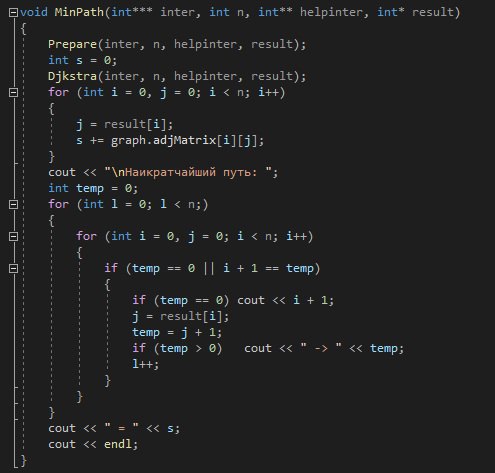


Рис. 9

10) Функция координат вершин и их радиуса.

Центр правильного вписанного многоугольника находится в центре экрана.

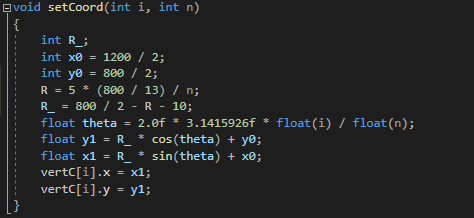


Рис. 10

11) Функции отрисовки текста в вершинах и меню

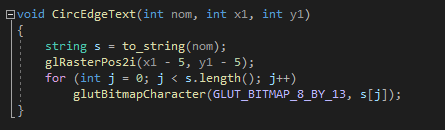


Рис. 11

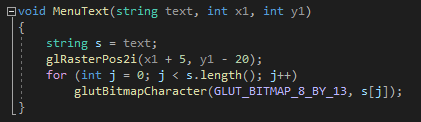


Рис. 12

12) Функции отрисовки вершин с координатами и связывающих их рёбер.

Круг рисуется в зависимости от кол-ва вершин, и чем больше вершин, тем меньше радиус.

Отрисовка линий и вершин происходят сразу с текстом.

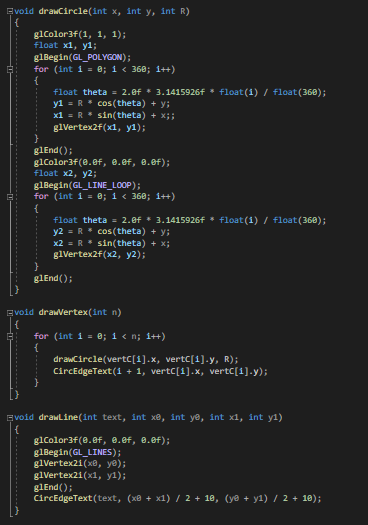


Рис. 13

13) Функция отрисовки графа.

Стояла проблема в расположении вершин, и поэтому было выбрано расположение по кругу на вершинах вписанного правильного многоугольника с n вершин, чем горжусь, поскольку реализуется достаточно легко, выглядит красиво, но есть проблемы с накладыванием текста рёбер, но это не критично.

Линии рисуются раньше вершин, поскольку нужно, чтобы вершины накладывались на них, а не наоборот.

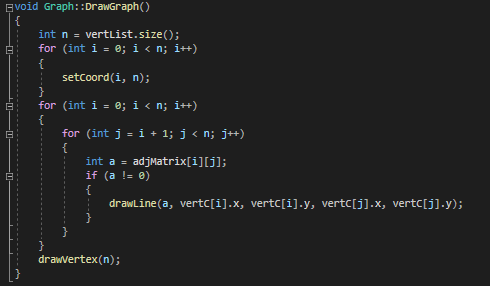
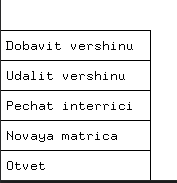


Рис. 14

14) Выполнить отрисовку кнопок и привязки к ним функций

Реализовано координатами положения мыши и функции GLUT\_DOWN считывающей нажатие.



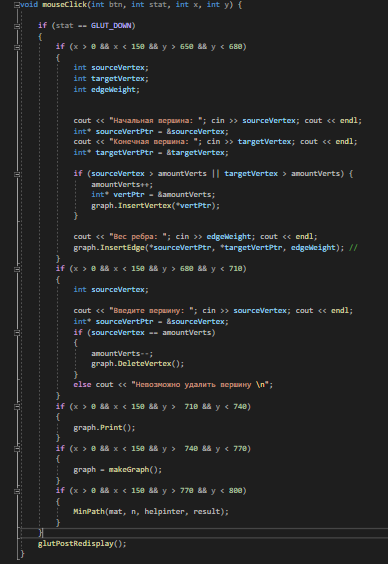


Рис. 15

15) Функция отрисовки меню в левом нижнем углу с разделением кнопок линиями

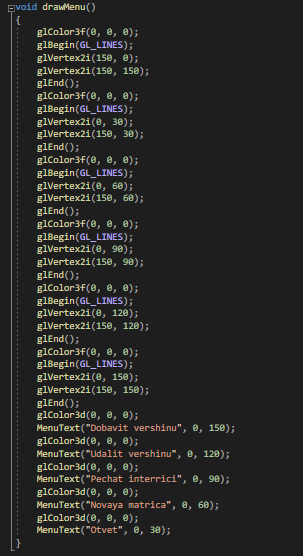


Рис. 16

16) Функция отрисовки окна с параметрами и функцией отрисовки графа

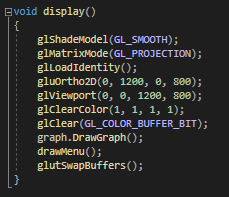


Рис. 17

**3) UML-диаграмма**

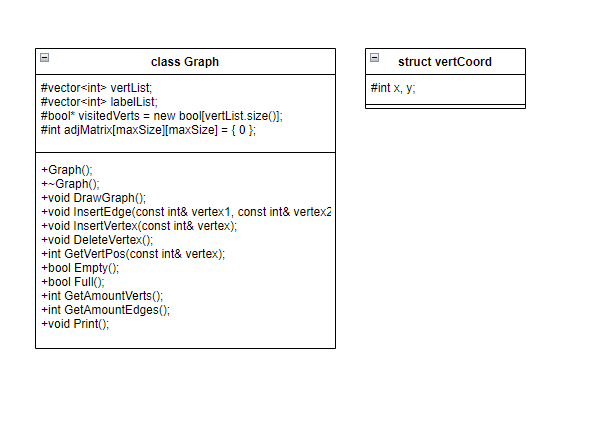


Рис. 18 UML-диаграмма класса Graph и структуры vertCoord

**4) Код с комментариями:**

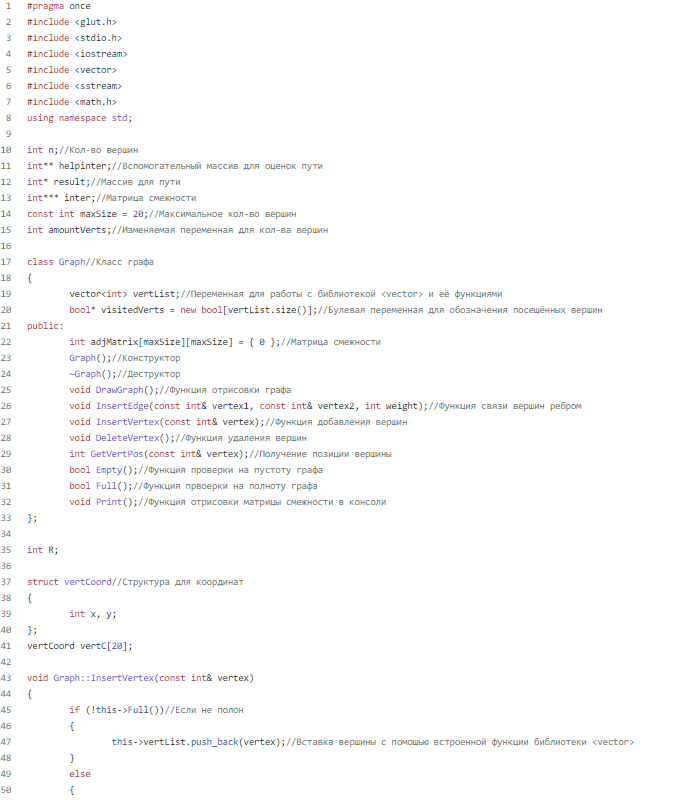












Рис. 19 Код программы

**5) Скриншот с введёнными данными и результатом:**

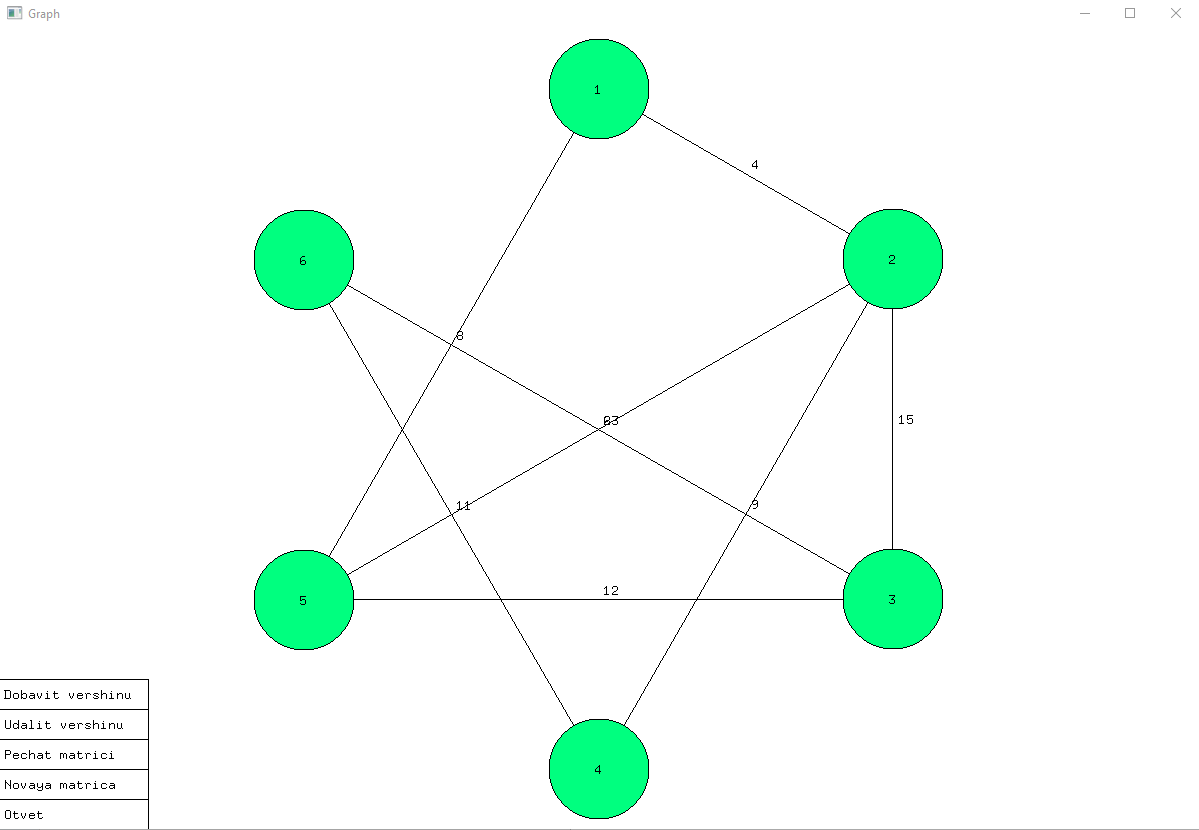


Рис. 20 Графическая отрисовка графа

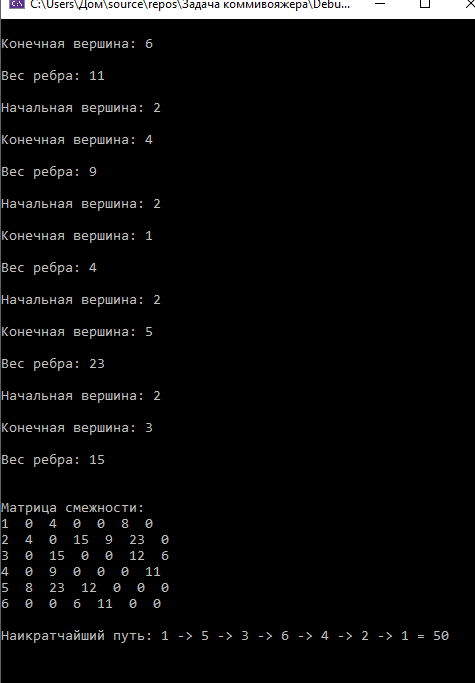


Рис. 21 Результат работы алгоритма

Дизайн минималистичен без всего лишнего, ведь нужно чтобы программа работала быстро и чисто

**Калькулятор градусов:**

**1)Постановка задачи:**

1. Создать в конструкторе оформление калькулятора.
2. Прописать функции для действий.
3. Проверить правильность выполнения всех действий.

**2)Решение задачи:**

Переменные:

A,b,c,d,f,g - хранение частей чисел first и second.

First,second,third - переменные для храения первого, второго числа и результата.

Var - переменная для хранения действий.

Plus,minus,mult,divide - хранение знаков действий для преобразования.

Решение было реализовано с помощью WindowsForms.

Трудностей в решении задачи не было встречено, поскольку WindowsForms лёгок и интуитивен.

1) Внешний вид запущенного калькулятора и в конструкторе.

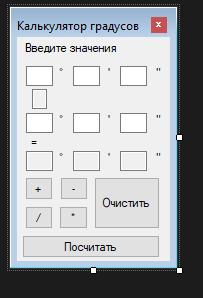
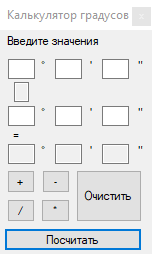


Рис. 22

2) С введёнными данными и полученным результатом.

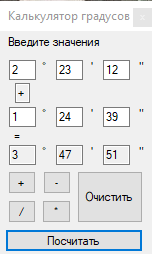


Рис. 23

Такой дизайн был выбран для минимализации используемого пространства без добавления лишнего, что можно записать в достоинства.

3) Функции сложения, вычитания, деления и умножения. Здесь сразу производится проверка на пустоту в поле 2-го числа. Проверка на пустоту полей после проверки на пустоту полей 2-го числа с заменой их на 0, если они пустые, т.к. программа не будет работать без этого.

Дальше представлен алгоритм перевода и вычисления итогового числа с помощью функций преобразования System::Convert::Тип\_Данных из string в int

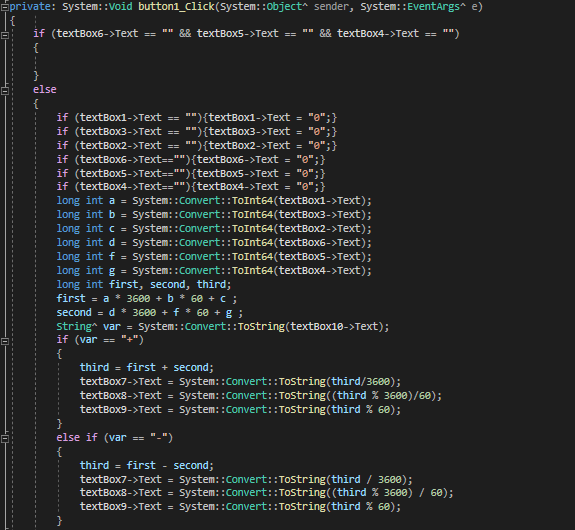


Рис. 24

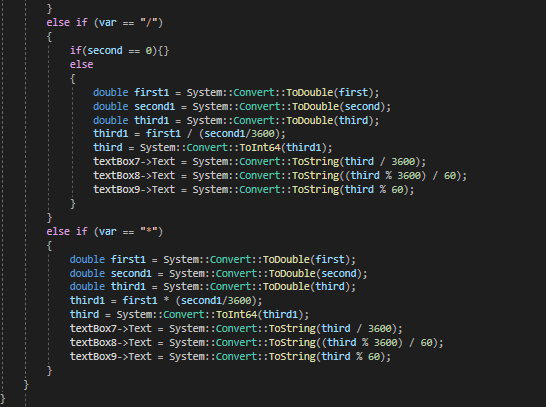


Рис. 25

4) Функция очистки полей

После нажатия на кнопку все поля со значениями очищаются

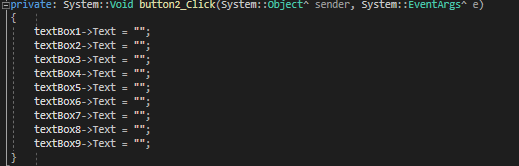


Рис. 26

5) Функции ввода для поля знаков действий, самих знаков действий

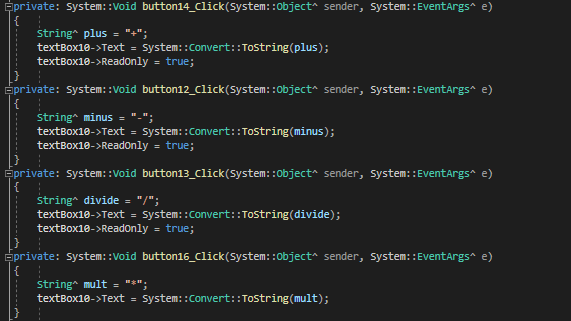


Рис. 27

**3) UML-диаграмма**

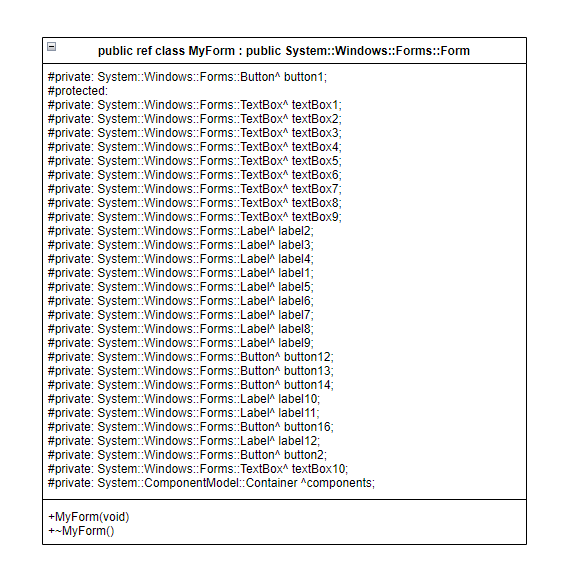
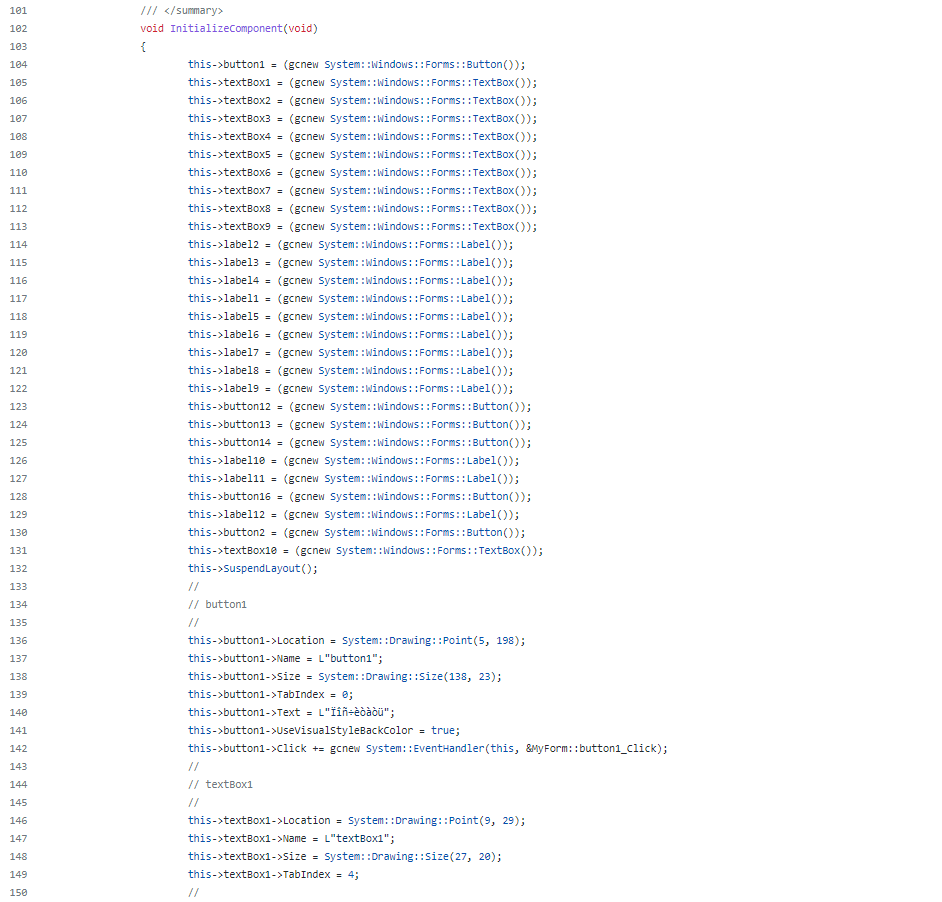
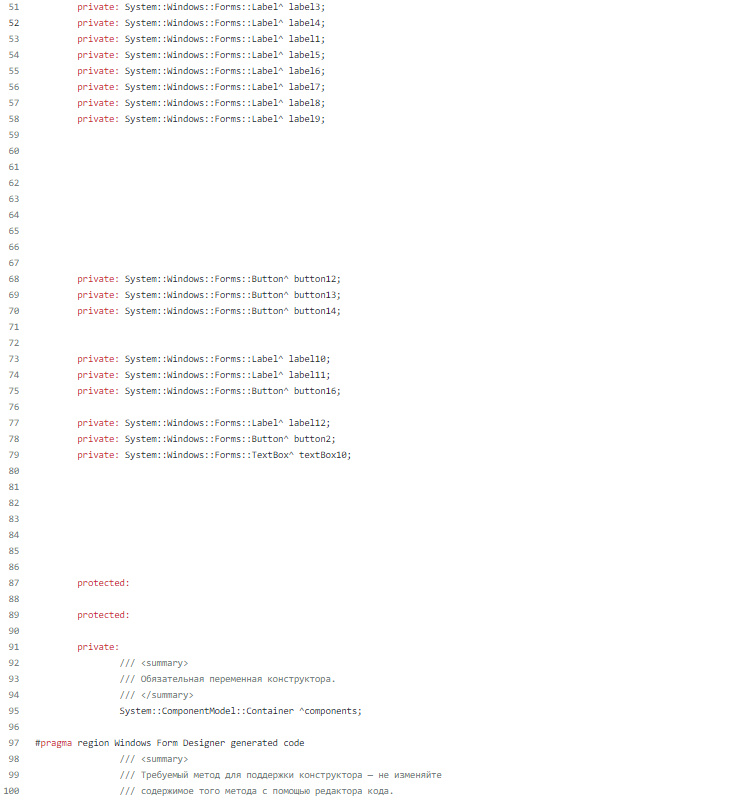


Рис. 28 UML-диаграмма класса MyForm

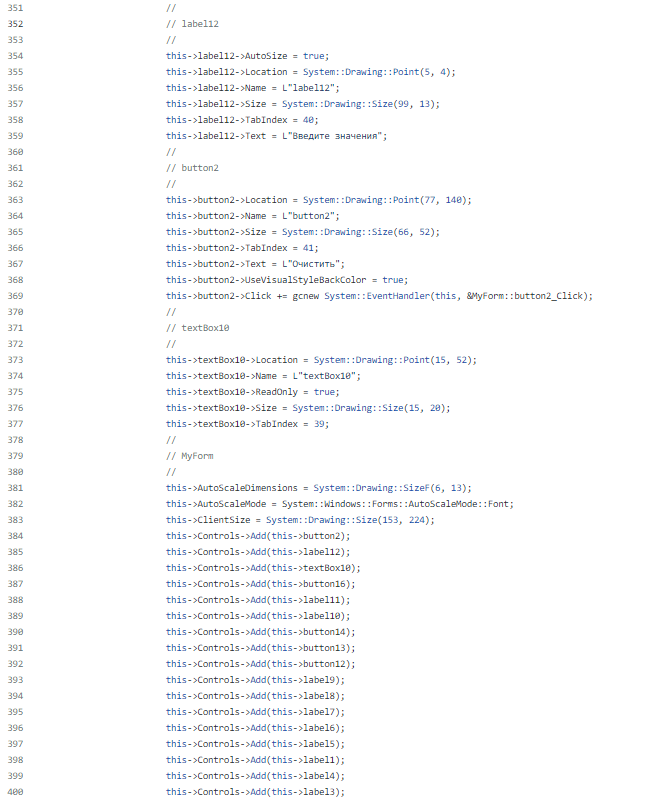
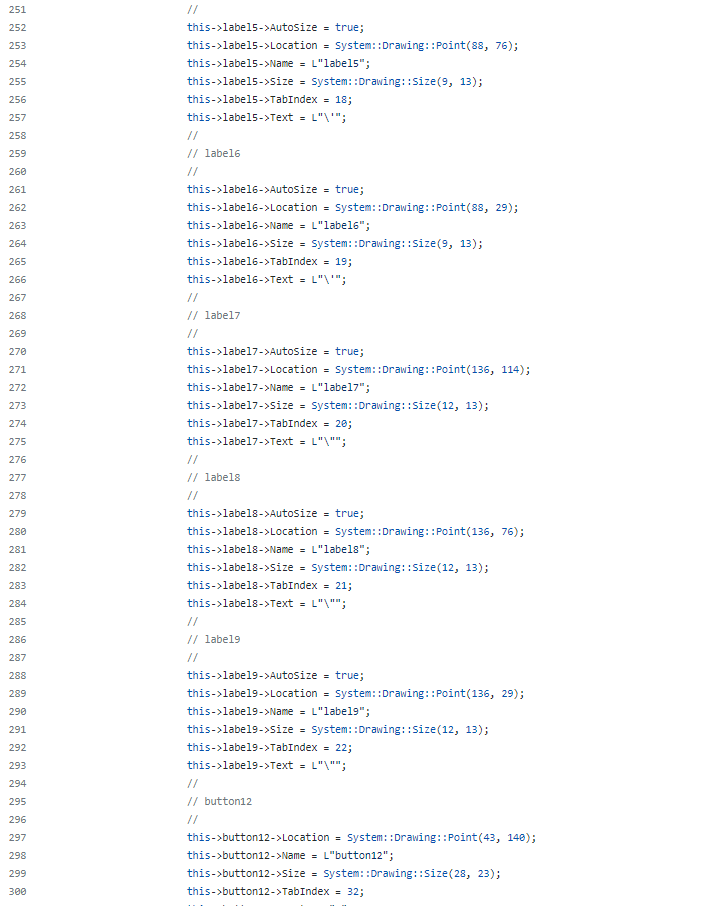
**4) Код с комментариями:**









Рис. 29 Код калькулятора

**6) Ссылки на GitHub и YouTube**

**YouTube:**

https://www.youtube.com/channel/UC2MuNrHJoBIkbPxQGk42-3w

**GitHub с творческим заданием:**

https://github.com/ChinaRBM/Labs\_PSTU/tree/main/Программы/Творческое%20задание