Министерство образования и науки Российской Федерации

ФГБОУ ВО

«Саратовский государственный технический университет им. Гагарина Ю.А.»

Кафедра «Информационная безопасность автоматизированных систем»

Специальность 10.05.03 Информационная безопасность автоматизированных систем

**Расчетно-графическая работа**

по дисциплине «Языки программирования»

**Сети, определение ключевых узлов, которые нужно**

**разрушить для разделения сети на сегменты**

|  |
| --- |
| **Выполнил**: студент 1 курса  учебной группы с-ИБС11  очной формы обучения  Филенко И.А. **Руководитель работы:**  Романчук С.П. |

Расчетно-графическая работа защищена на оценку \_\_\_\_\_\_\_\_\_\_\_\_\_

Саратов 2020

Оглавление

[**Задание** 3](#_Toc43360920)

[**Введение** 4](#_Toc43360921)

[**Теоретическая часть** 5](#_Toc43360922)

[**Инструменты для реализации задачи** 6](#_Toc43360923)

[**Описание алгоритма работы** 7](#_Toc43360924)

[**Практическая часть** 8](#_Toc43360925)

[**Пример работы функции** 9](#_Toc43360926)

[**Заключение** 10](#_Toc43360927)

[**Приложение** 11](#_Toc43360928)

[**Меню приложения** 16](#_Toc43360929)

[**Пример работы программы** 17](#_Toc43360930)

# **Задание**

Суть задания данной РГР — написать программу, которая представляет собой простейшую локальную компьютерную сеть. Где будет определяться минимальное и максимальное число вершин, которые нужно будет уничтожить для разрывания сети. Результат работы будет выводить на экране локальную сеть, с минимальными вершинами удаления, для разрыва. Возможный формат которого описан ниже:

N – число вершин.

X1 Y1 // координаты вершины

X2 Y2

...

XN YN

M – число связей.

N1 K1 // какие вершины связываются данной связью

N2 K2

...

NM KM

# **Введение**

Эта РГР была написана студентом кафедры «Информационная безопасность автоматизированных систем» группы с-ИБС 11.  
Суть данной работы – создание простейшей локальной компьютерной сети, для узнавания минимальное количества вершин для разрыва связи.  
Практическая польза написанной программы заключается в том, что опираясь на код и структуру выполненной работы, можно создать с большей компьютерной сетью и усложненным функционалом разрыва связей.

# **Теоретическая часть**

Задача о “’разрыве локальной сети” остается актуальной во все времена и

находит применение во многих областях нашей жизни. Данная задача началась ещё в 1652 году так называемая “ Задача о кёнигсбергских мостах”. Суть задачи составляла пройти по мостам, но не пройти из них дважды. Тогда решение этой задачи придумал Леонардо Эйлер.

На упрощённой схеме города (графе) мостам соответствуют линии (ребра графа), а частям города — точки соединения линий (вершины графа). В ходе рассуждений Эйлер пришёл к следующим выводам:

* Число нечётных вершин (вершин, к которым ведёт нечётное число рёбер) графа должно быть чётно. Не может существовать граф, который имел бы нечётное число нечётных вершин.
* Если все вершины графа чётные, то можно начертить этот граф без отрыва карандаша от бумаги, при этом можно начинать с любой вершины графа и завершить его в той же вершине.
* Если ровно две вершины графа нечётные, то можно начертить этот граф без отрыва карандаша от бумаги, при этом нужно начинать с одной из нечётных вершин и завершить его в другой нечётной вершине.
* Граф с более чем двумя нечётными вершинами невозможно начертить одним росчерком.

Граф кёнигсбергских мостов имел четыре нечётные вершины (то есть все) — следовательно, невозможно пройти по всем мостам, не проходя ни по одному из них дважды. После решения этой задачи, появляется смысловое название “Графы”.

**Граф**– это, в общем случае, математический объект (или геометрическая схема), который представляет собой совокупность вершин, соединенных рёбрами.

Вершины, в зависимости от контекста задачи, могут изображать точки назначения (города, острова, местоположения людей и т.п.), узлы связи (в компьютерных сетях), конкретных людей или адресатов и т.д. Значения рёбер также зависит от условий задачи – они могут обозначать как пути между вершинами, так и связи разного рода (социальные, экономические, физические и т.п.). Поэтому сейчас все чаще выделяют особые виды графов в рамках конкретных областей применения: социальные, молекулярные, веб‑графы и др.

# **Инструменты для реализации задачи**

Критериями к реализации служили 2 фактора:

1. IDE – Visual Studio
2. Язык программирования С++

Для реализации задачи подойдет любая графическая библиотека С++, однако на чистом WinApi программа также имеет возможность написания.

От выбора библиотеки зависят лишь такие факторы:

1. Сложность реализации и размер кода
2. Время на разработку проекта
3. Скорость выполнения/отрисовки

После изучения описания существующих графических библиотек, выбрал Visual Studio.

Преимущества Visual Studio:

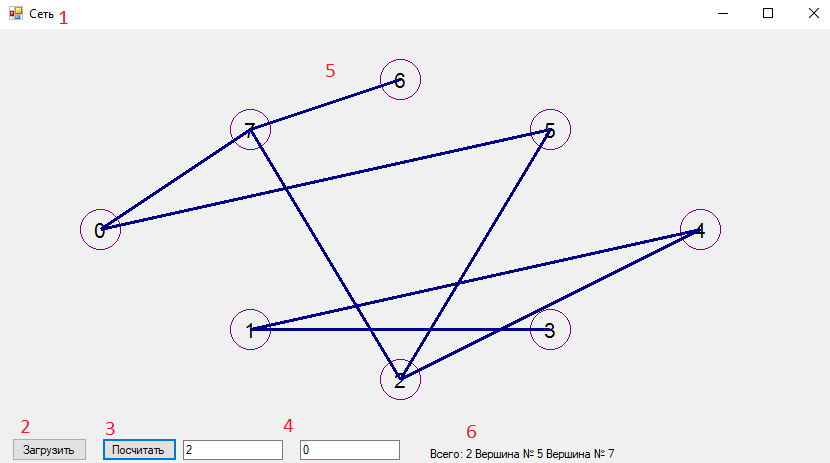
1. Удобное межпроцессное взаимодействие (сигналы/слоты). С появлением С++11 это неактуально.
2. Низкий порог вхождения.
3. Переносимость на уровне исходного кода. (Компиляция под Windows, Linux, Mac, QNX, Android, IOS, WinRT)
4. Хорошая документация.
5. Множество примеров.
6. Поддержка почти всех языков программирования.

Недостатки Visual Studio:

1. Большой вес приложений. (библиотеки, в зависимости от того, что вы используете, будут весить от 15 Мб и больше. Особенно неудобно для мобильных платформ).

# **Описание алгоритма работы**

Для понимания дальнейшей структуры алгоритма произведем демонстрацию всех используемых объектов:



Где:

1. WorkspaceForm – объект главного окна программы
2. Download – загрузка локальной компьютерной сети
3. To count – быстрый расчёт минимальной вершины для разрыва связи
4. Menu – пункт для ввода вершин, между которыми нужно разорвать связь
5. Computer Network – нарисованная локальная компьютерная сеть
6. Conclusion – вывод количества вершин, которые нужно удалить для разрыва связи

# **Практическая часть**

Алгоритм:

1. Заполняем вершины, к которым хотим рассчитать связи.

2. Происходит инициализация городов (вершин) и расстояний.

3. Далее выполняется шаг алгоритма.

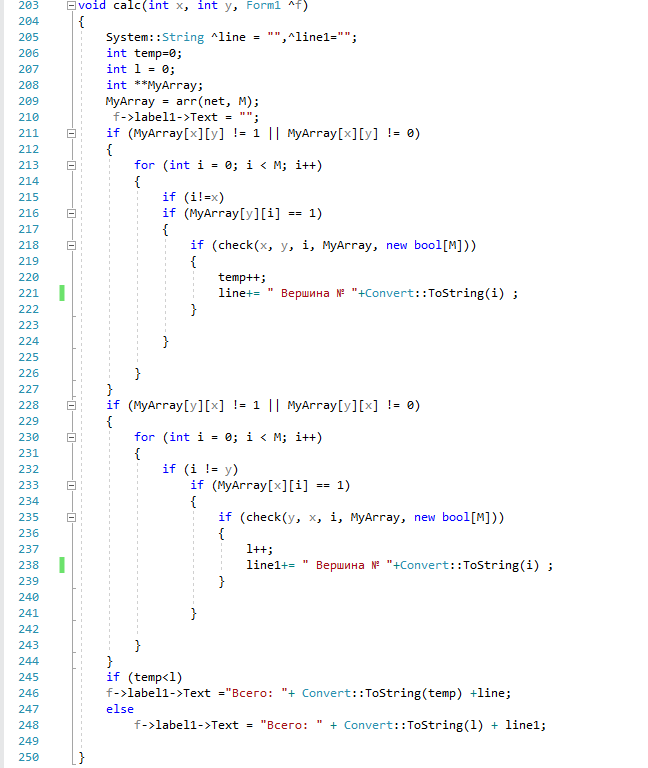
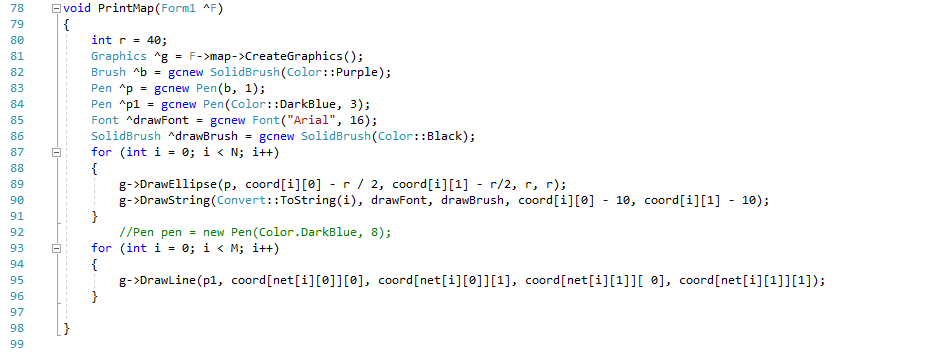
4. Вывод возможных для удаления вершин.

Используемые библиотеки:

1. #include "Form1.h" - заголовочный файл.
2. #include <string> - заголовочный файл для работы со строками.
3. #include <iostream> - заголовочный файл библиотеки для работы с потоками ввода-вывода консоли.
4. #include <fstream> - заголовочный файл библиотеки для работы с файловыми потоками.

Классы: отсутствуют.

# **Пример работы функции**

****

# **Заключение**

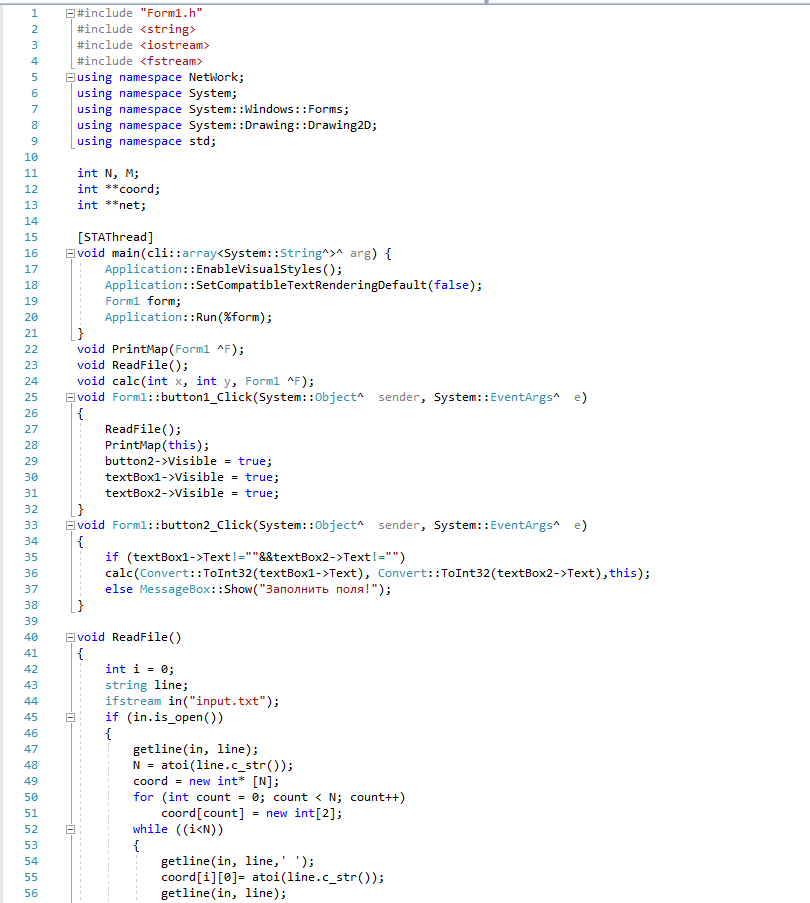
В ходе выполнения расчётно-графической работы было выполнено задание в соответствии с номером варианта, которое включало в себя составление рабочей программы на языке программирования С++.  
      В результате  были выполнены следующие этапы:

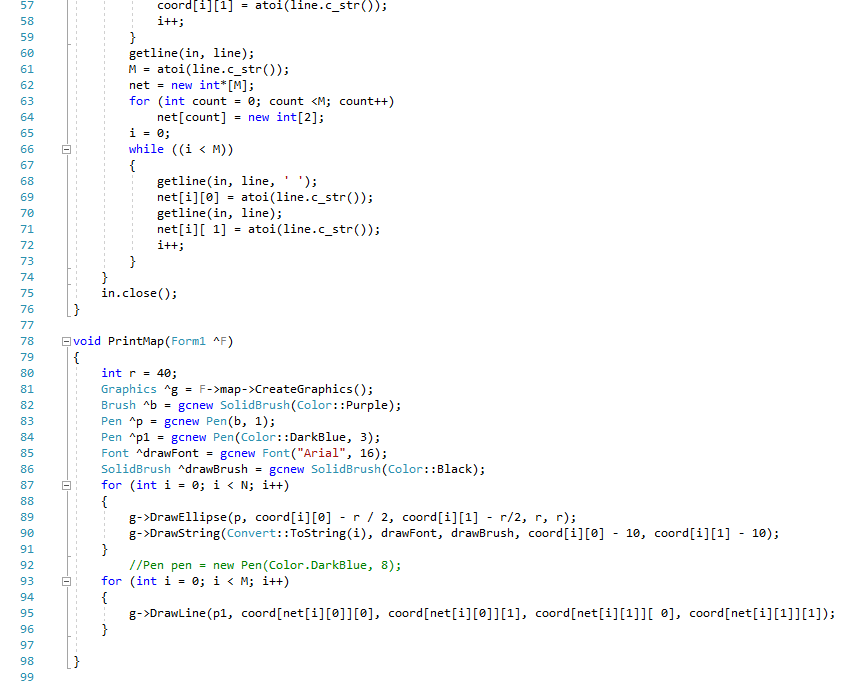
1. Обследование и разработка технического задания
2. Построение алгоритма программы
3. Разработка программы, для решения разных типов построения графов
4. Тестирование и устранение ошибок

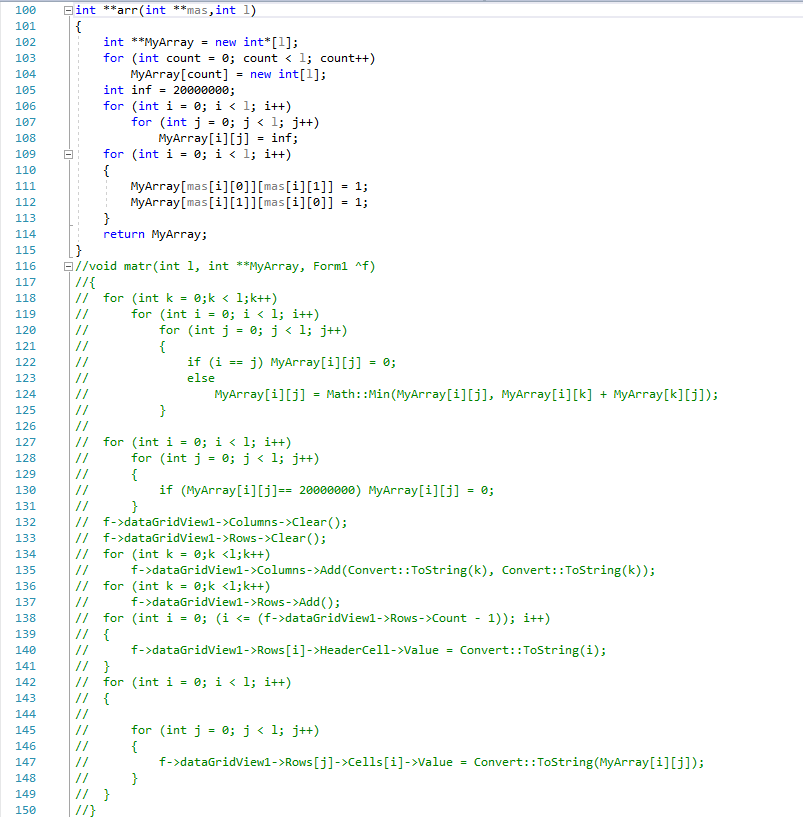
В процессе написания и отладки программ были закреплены навыки работы с разными возможностями языка C++. Итоговое приложение было протестировано, и результат проверки дал положительный результат. Данный код не представляет никакого труда человеку, чтобы понять его, изменить или дополнить часть его функционала.

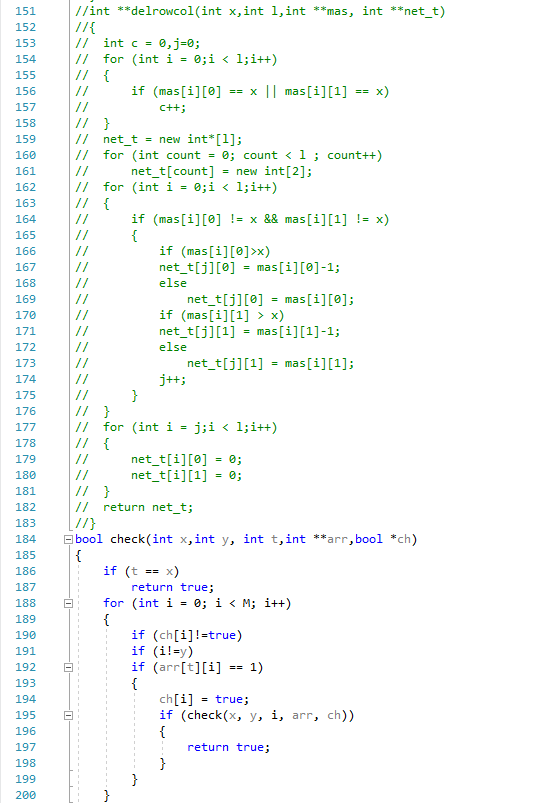
 Таким образом, мы приходим к следующему выводу - использование разработанной программы и доработка ее, заметно упростит передачу или хранение каких-то файлов по локальной компьютерной сети.

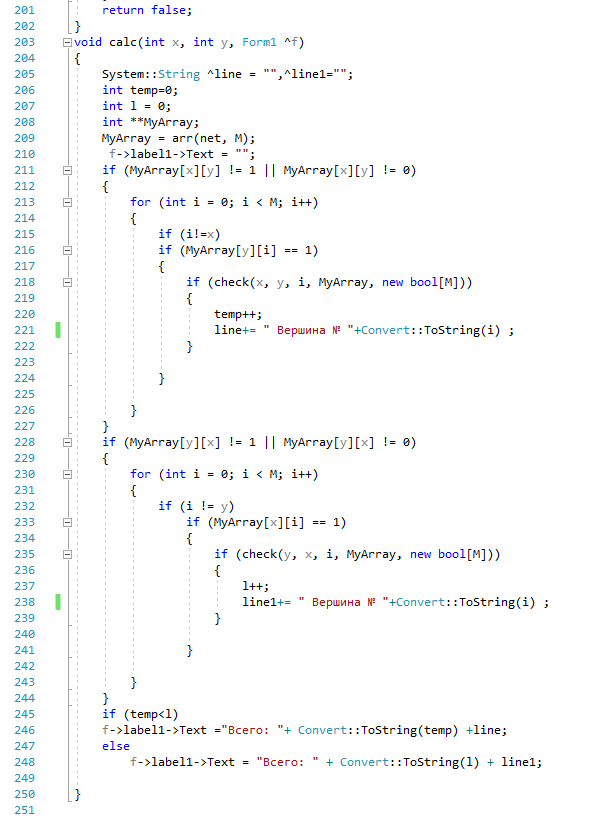
# **Приложение**

****

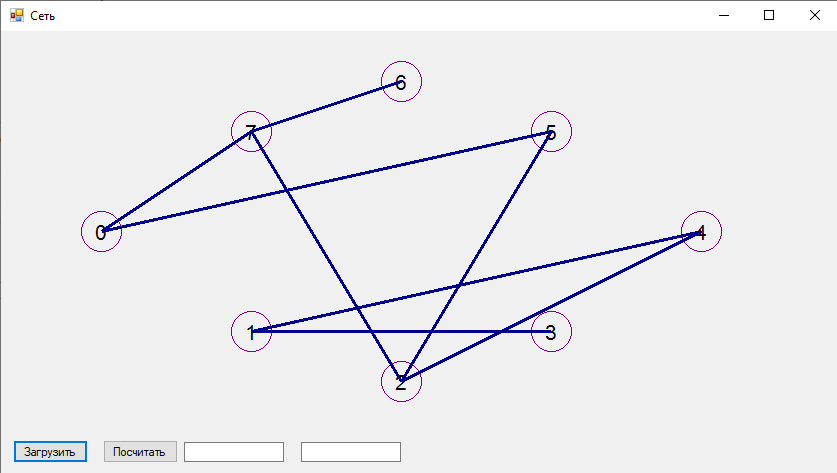
****

****

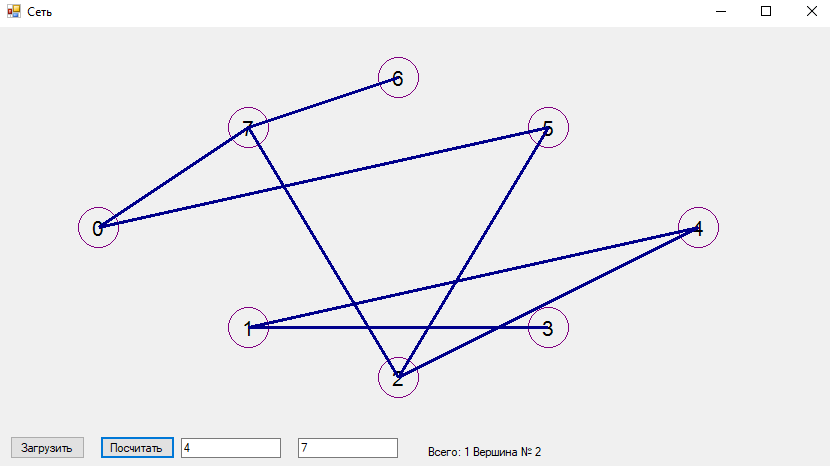
****

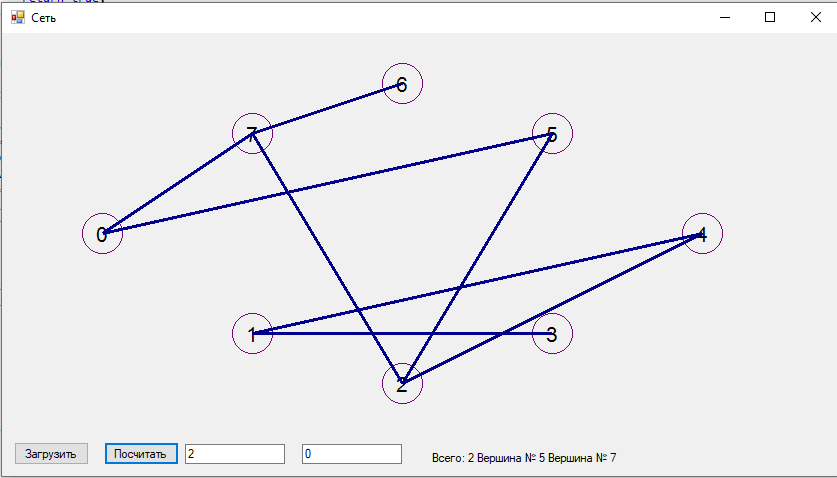
****

# **Меню приложения**

****

# **Пример работы программы**

****

****