МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационных систем

ОТЧЕТ

по практической работе №7

по дисциплине «Объектно-ориентированное программирование»

Студент гр. 8374 _____ Пихтовников К.С.

Студент гр. 8374 _____ Подсекин Г.С.

Преподаватель _____ Егоров С.С.

Санкт-Петербург

2021

Задание на практическую работу

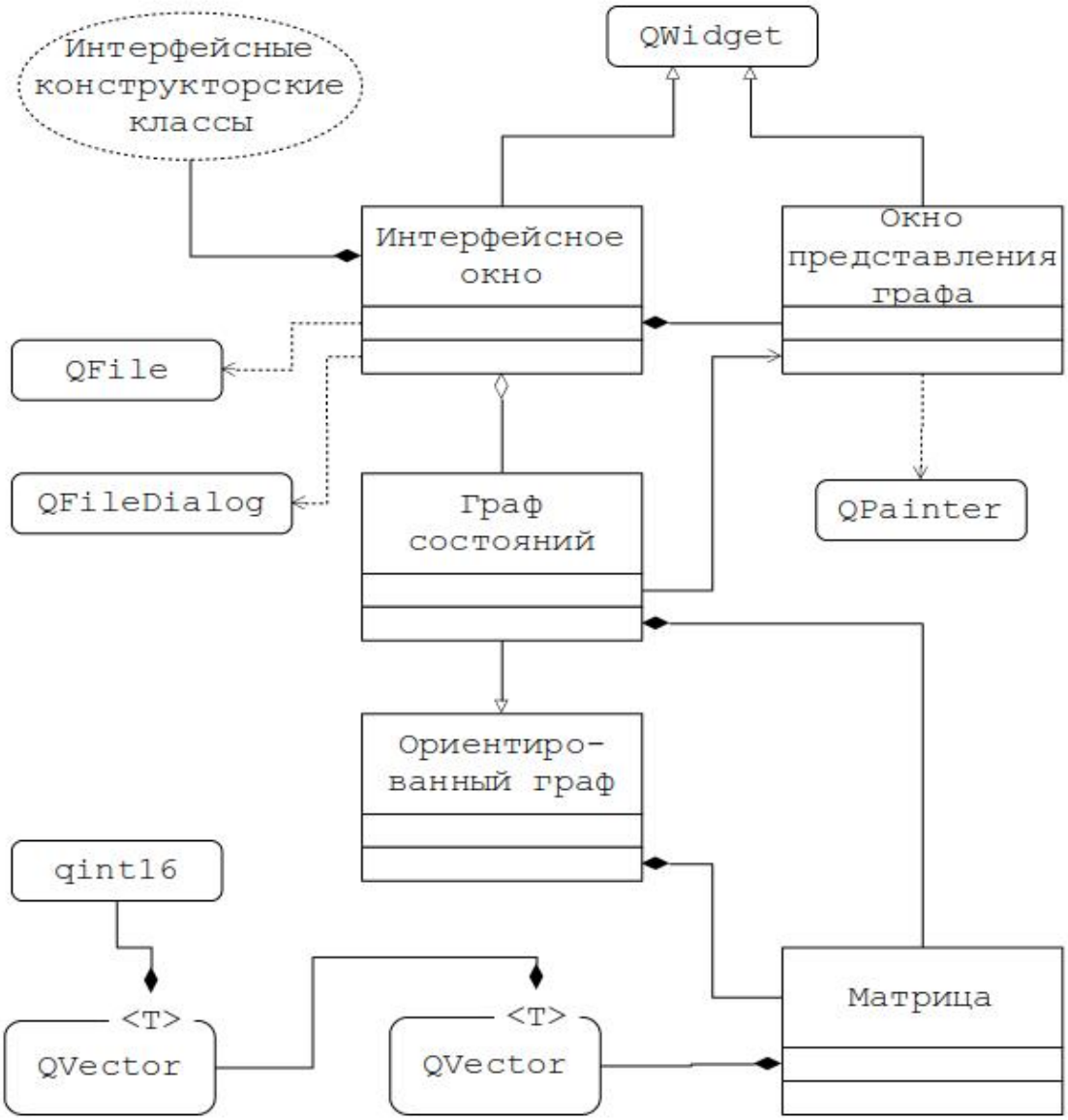


Рис.1. Диаграмма классов работы №7

Разработать GUI приложение, выполняющее функцию визуализации графа состояний.

Граф состояний - это ориентированный граф, одна из вершин которого в каждый момент времени считается активной. Каждой дуге приписано некоторое событие, при возникновении которого происходит смена активной вершины.

Граф состояний описывается матрицей, число строк которой равно числу вершин, а число столбцов - числу событий. Элементом i-ой строки и j-го столбца является номер строки (т.е. соответствующая ей вершина графа), которая становится активной при возникновении j-го события, если при этом вершина i была активна.

На рис.1 представлен макет диаграммы классов приложения, который требуется реализовать в приложении.

Основной функцией объекта класса "Интерфейсное окно" является выбор файла, который содержит данные о графе состояний. При чтении файла необходимо проверить корректность данных и в случае обнаружения ошибки необходимо сформировать соответствующее сообщение пользователю.

Номер активной вершины также задается в интерфейсе.

При корректности данных создается объект класса "Граф состояний", устанавливаются (если необходимо) связи между новым объектом и существующими, после чего граф отображается в соответствующем окне (объект класса "Окно представления графа").

Активная вершина помечается цветом. При смене значения номера активной вершины должны происходить изменения в отображении.

В интерфейсе должна быть предусмотрена возможность инициирования любого из возможных событий. При их возникновении должен происходить переход в новую активную вершину, согласно графу, смена значения в интерфейсном окне и его перерисовка.

При выборе в интерфейсе другого графа (другого файла) старый должен заменяться на новый, номер активной вершины принимать исходное (корректное) значение и граф перерисовываться.

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

Спецификация классов

Класс TInterface

Метод/атрибут	Описание
Атрибут TStateGraph * g;	область видимости - public. Переменная, которая отвечает за состояние графа
Атрибут TCanvas * canvas;	объект класса TCanvas, область видимости private
Метод TInterface(QWidget *parent = nullptr)	Область видимости public. Конструктор класса
Метод ~TInterface()	Область видимости public. Деструктор класса
Метод void on_pushButton_file_clicked();	Формальных параметров нет, область видимости private. Метод позволяет загрузить файл
Метод void on_pushButton_view_clicked();	Формальных параметров нет, область видимости private. Метод обрабатывает и выводит на экран то, что находится в файле

Таблица 1. Класс Tinterface

Класс TGraph

Метод/атрибут	Описание
Атрибут int curr_str;	область видимости — private. Хранит размер матрицы
Атрибут TMatrix matrix;	объект класса TMatrix, область видимости private
Методы TGraph(int, Tmatrix); ~TGraph();	область видимости — public. Конструктор и деструктор класса
Метод int create_m(unsigned int,unsigned int);	область видимости — public. Метод создает матрицу
Метод void delete_m();	Область видимости — public. Метод удаляет матрицу
Методы unsigned int get_m_lines(); unsigned int get_m_columns();	Область видимости — public. Методы возвращают количество строк и столбцов матрицы

Таблица 2. Класс TGraph

Класс TMatrix

Метод/атрибут	Описание
Атрибут unsigned int lines; unsigned int columns;	область видимости — private. Количество строк и столбцов
Методы Tmatrix(); ~TMatrix();	Формальных параметров нет, область

	видимости public. Конструктор и деструктор класса
Метод QVector < QVector<qint16> > get_data();	Тип формальных параметров — QVector<QVector<qint16>>, область видимости public. Метод получает матрицу
Метод void set_data(unsigned int,unsigned int,qint16);	Тип формальных параметров-unsigned int,unsigned int,qint16, область видимости public. Метод записывает в матрицу значение в соответствии со строкой и столбцом

Таблица 3. Класс TMatrix

Класс TCanvas

Метод/атрибут	Описание
Атрибут TGraph *g;	объект класса TGraph, область видимости private
Методы: TCanvas(TGraph *, QWidget*parent = 0); *~TCanvas*();	область видимости — public. Конструктор и деструктор класса

Методы:
void paintEvent(QPaintEvent *);
void closeEvent(QCloseEvent*);
	область видимости — public.
	1 метод рисует граф, 2 закрывает окно при нажатии на определенную кнопку

Метод void *keyPressEvent* (QKeyEvent *event);	область видимости — protected. Метод обрабатывает нажатые клавиши

Таблица 4. Класс Tcanvas

Класс Stategraph

Метод/атрибут	Описание
Атрибут qint16 active_node;	Атрибут хранит текущий узел графа, который был выбран пользователем
Методы: StateGraph(); `~StateGraph();`	область видимости — public. Конструктор и деструктор класса

Методы:
void paintEvent(QPaintEvent *);
void closeEvent(QCloseEvent*);
	область видимости — public.
	1 метод рисует граф, 2 закрывает окно при нажатии на определенную кнопку

Методы:
void reset_active_node();
`qint16 get_active_node();`
	область видимости — public.
	Методы сбрасывают и устанавливают текущий узел, выбранный пользователем

Таблица 4. Класс Stategraph

Диаграмма классов

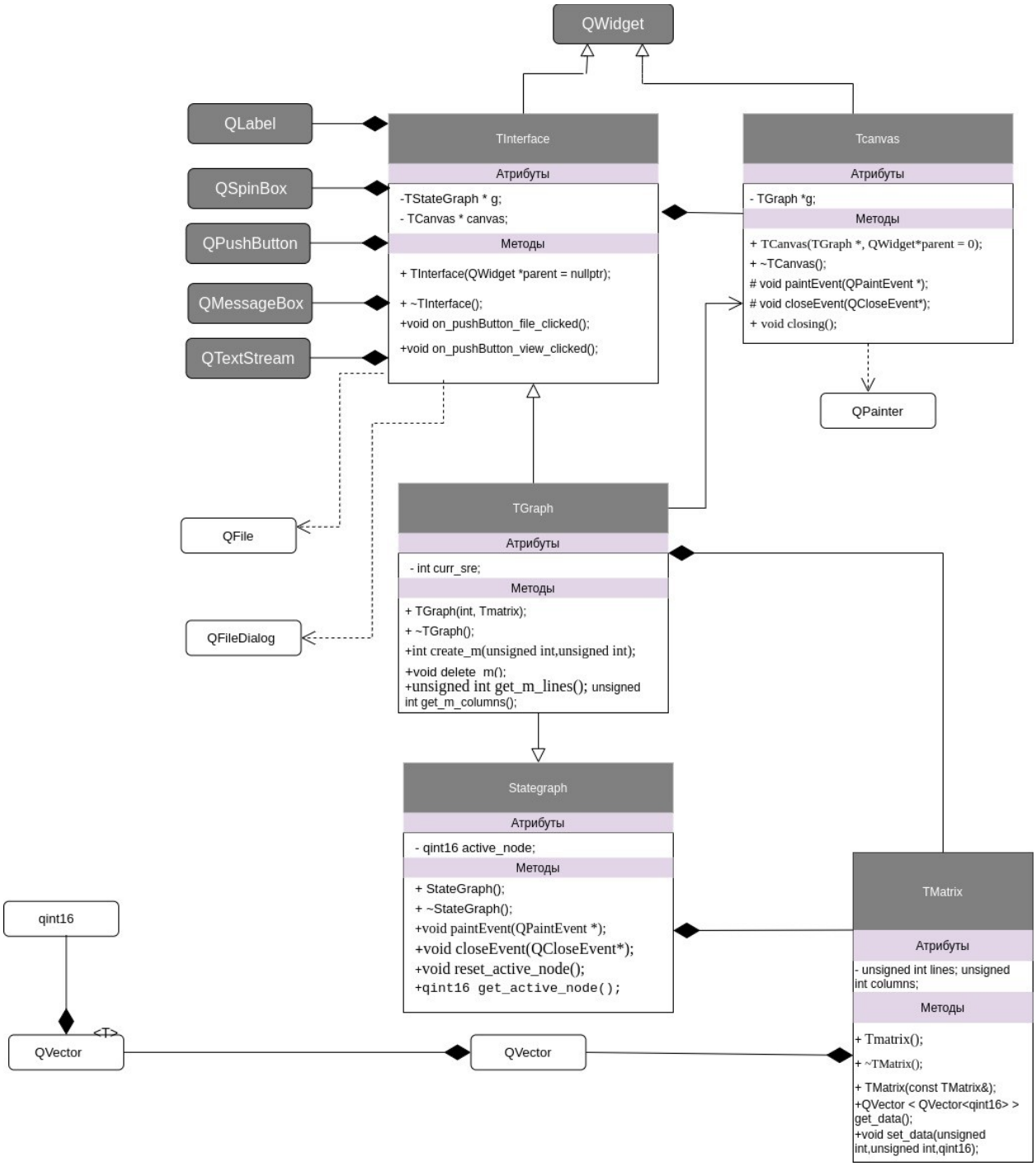


Рис.2. Реализация диаграммы классов клиентской части

Символ	Значение
+	public - открытый доступ
-	private - только из операций того же класса
#	protected - только из операций этого же класса и классов, создаваемых на его основе

Таблица 6. Обозначение аттрибутов и методов класса

Описание контрольного примера с исходными и ожидаемыми (расчетными) данными

Пример 1:

Исходные данные:

Матрица смежности:

```
5 5
4 0 1 2 3
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
```

Ожидаемые данные:

Должен быть построен граф с 5 вершинами, с соответствующими направлениями стрелок

Пример 2:

Исходные данные:

Матрица смежности:

```
5 3
4 0 1
0 1 2
1 2 3
2 3 4
3 4 0
```

Ожидаемые данные:

Должен быть построен граф с 5 вершинами, с соответствующими направлениями стрелок

Пример 3:

Исходные данные:

Матрица смежности:

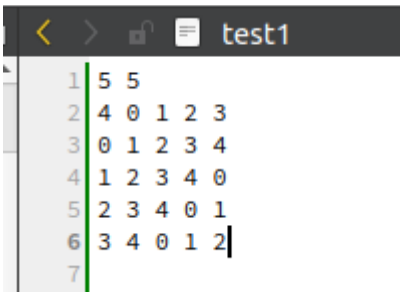
```
12 9
4 0 1 7 6 10 3 0 0
 0 1 2 5 8 1 7 3 1
 1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 11
3 4 0 5 6 7 8 9 10
4 5 6 7 8 9 10 11 0
5 6 7 8 9 10 11 0 1
6 7 8 9 10 11 0 1 2
7 8 9 10 11 0 1 2 3
8 9 10 11 0 1 2 3 4
9 10 11 1 2 3 4 5 6
10 11 1 0 2 3 4 5 6
```

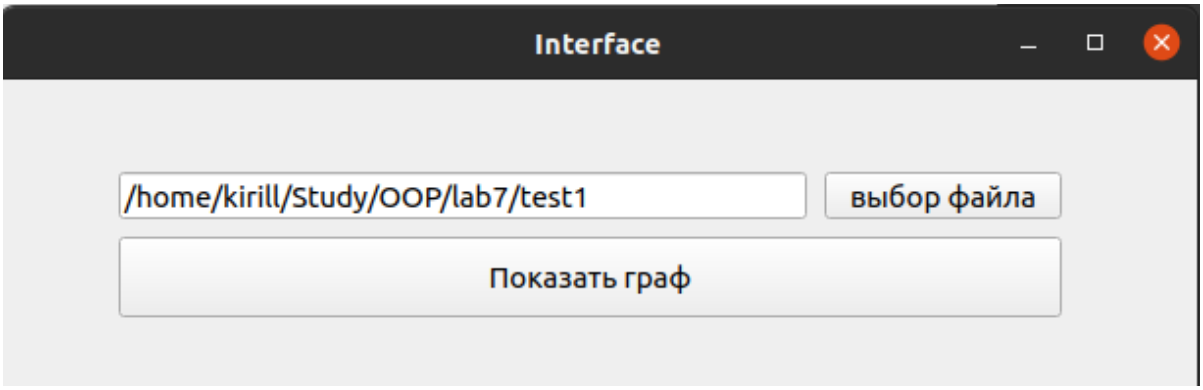
Ожидаемые данные:

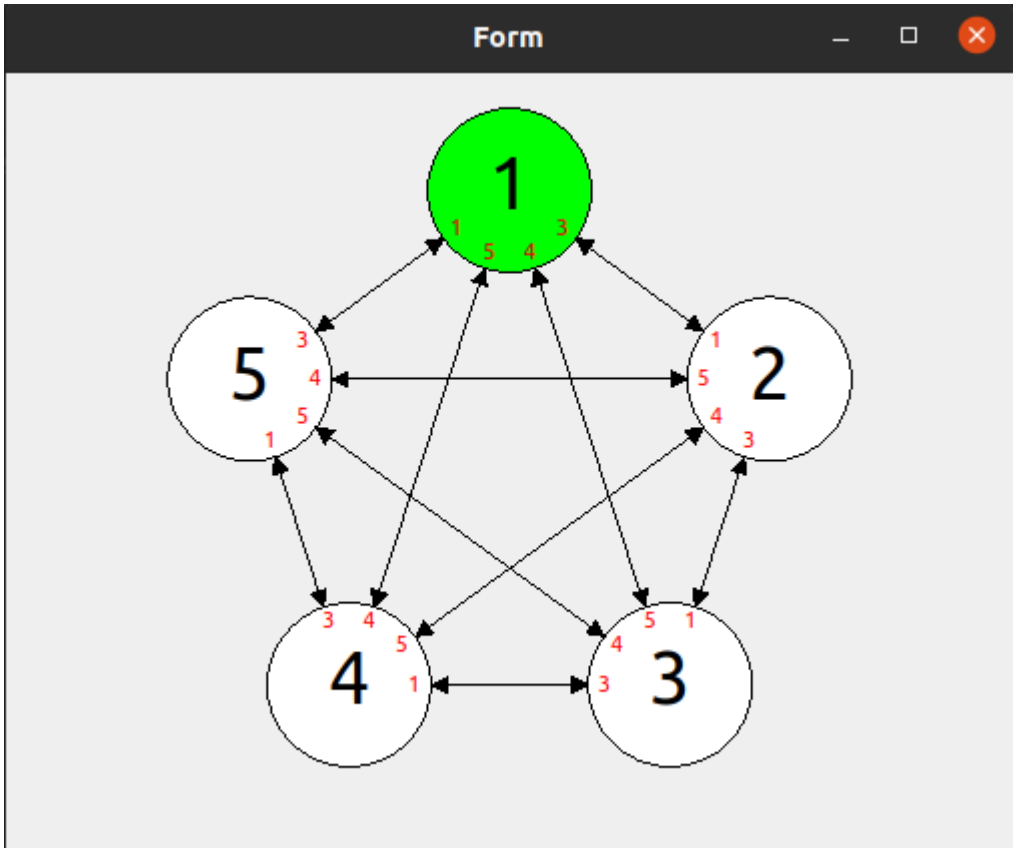
Должен быть построен граф с 12 вершинами, с соответствующими направлениями стрелок

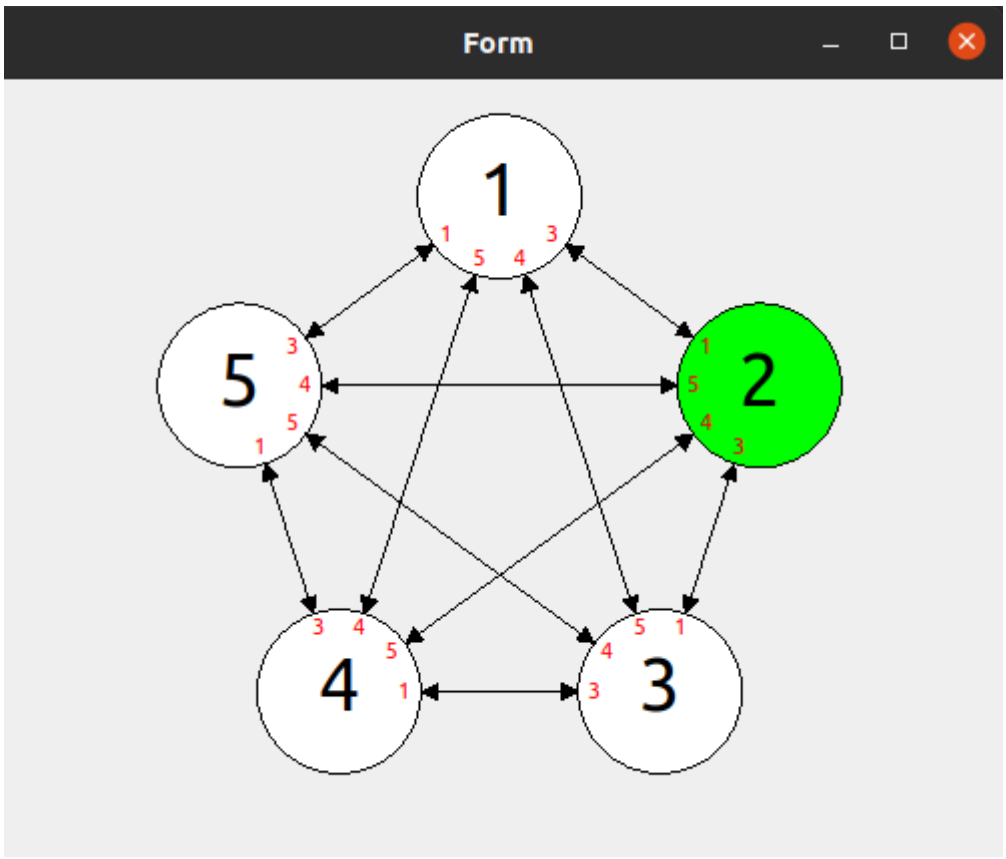
Скриншоты программы на контрольных примерах

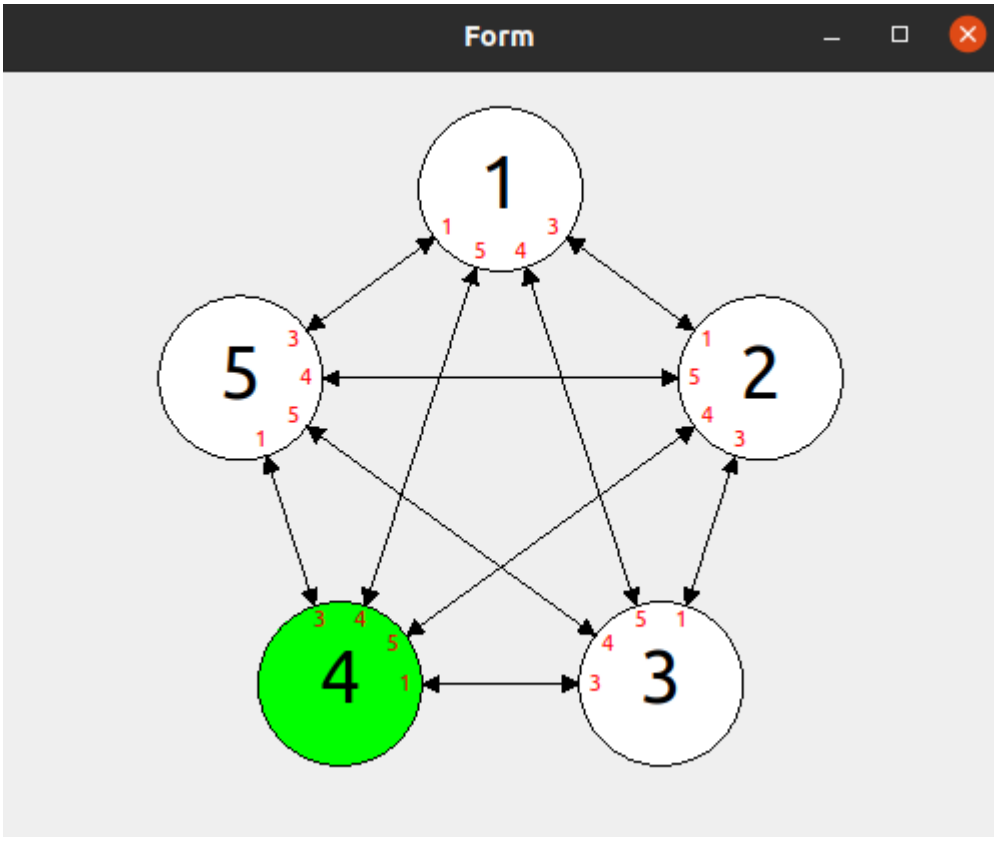
Пример 1:







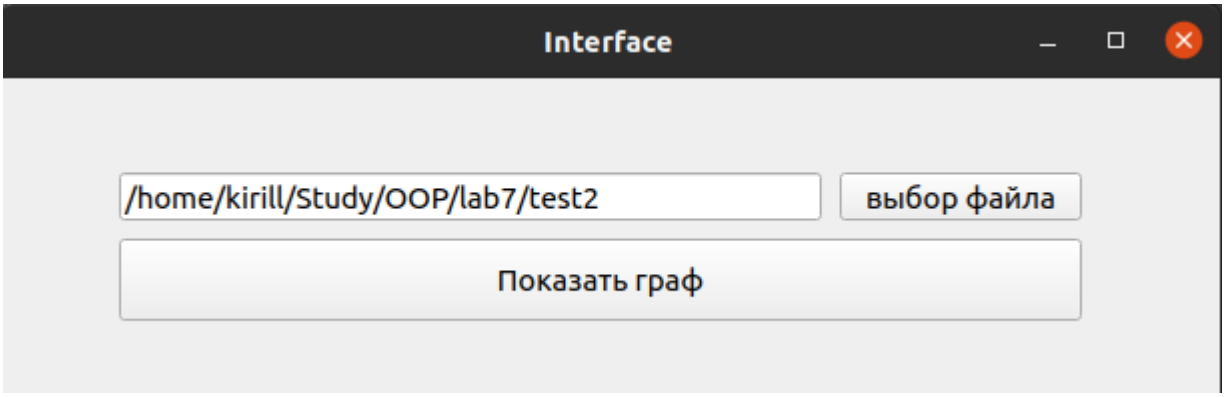

Form
1
5
2
4
3

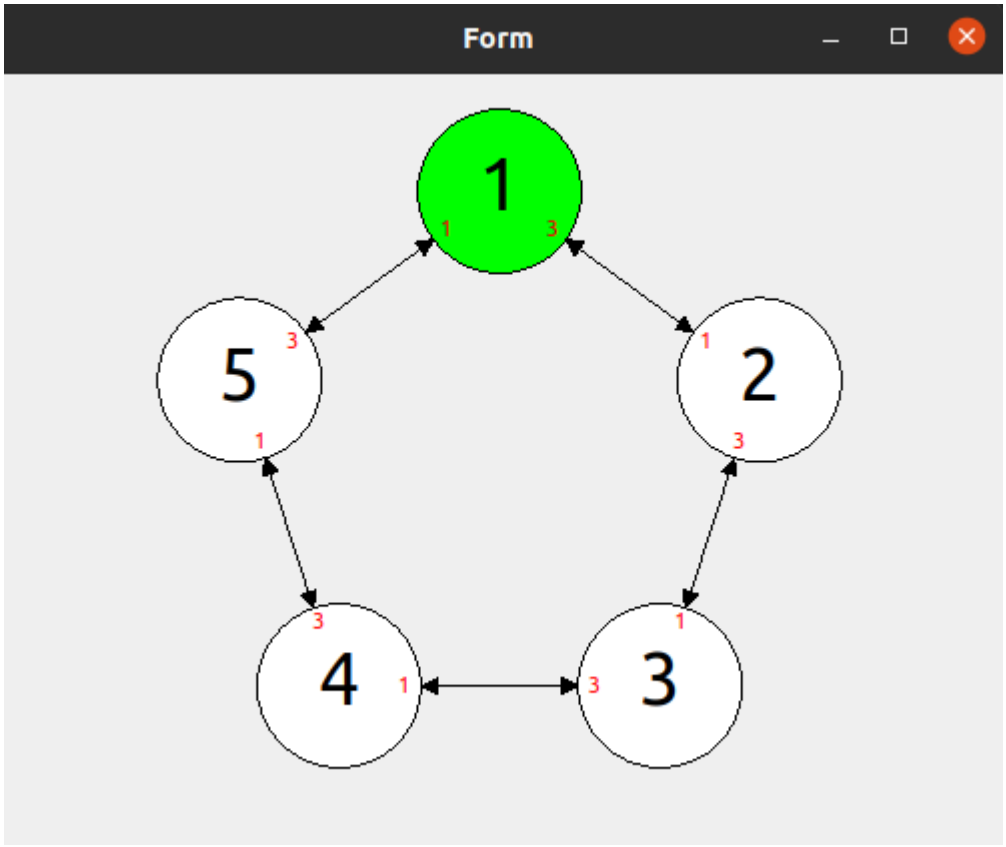

Form
1
5
2
4
3

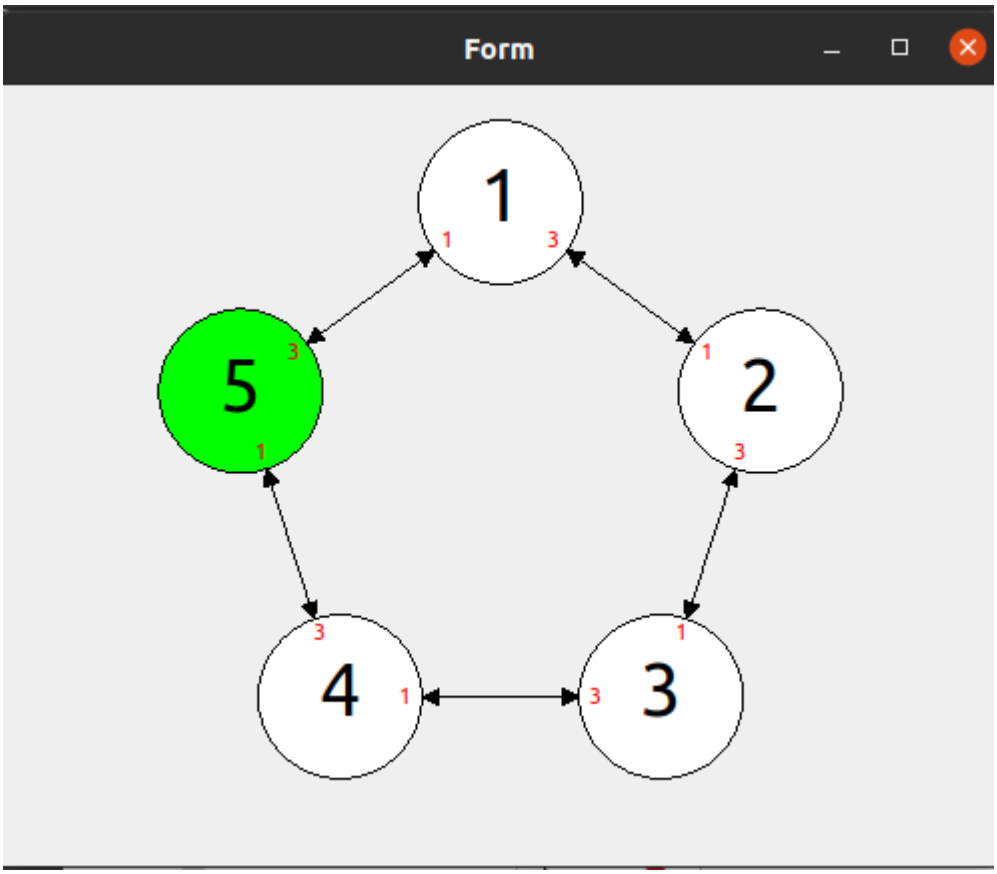
Пример 2:

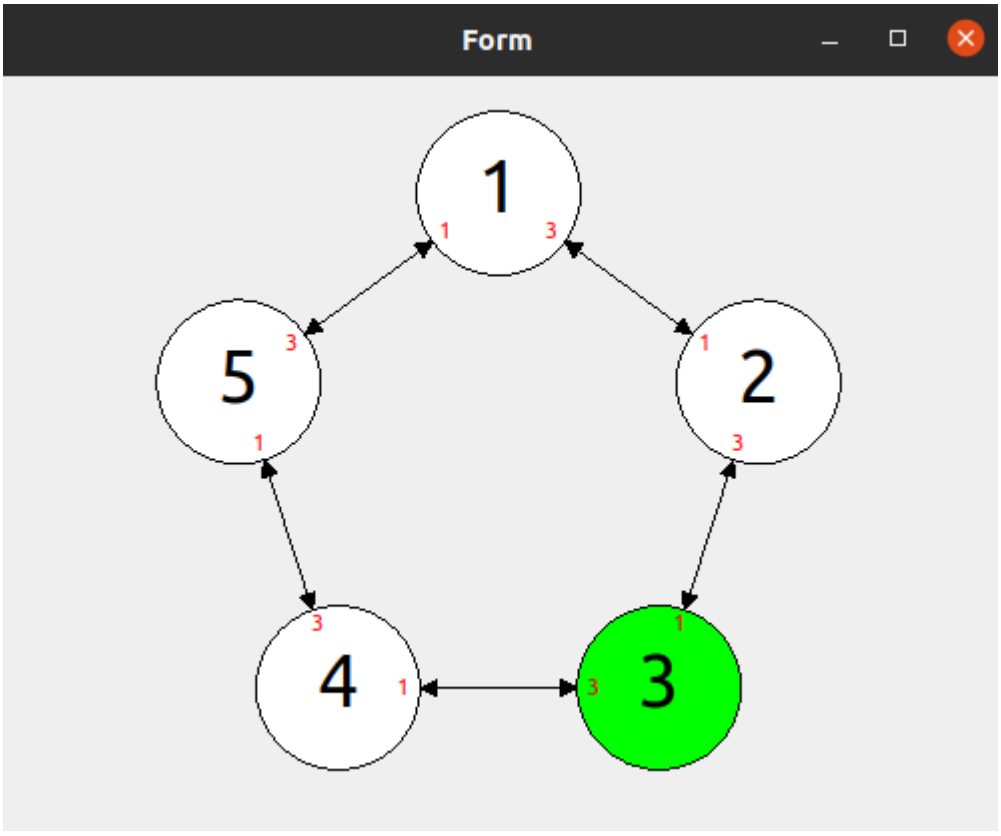
test2

```
5 3
4 0 1
0 1 2
1 2 3
2 3 4
3 4 0
```





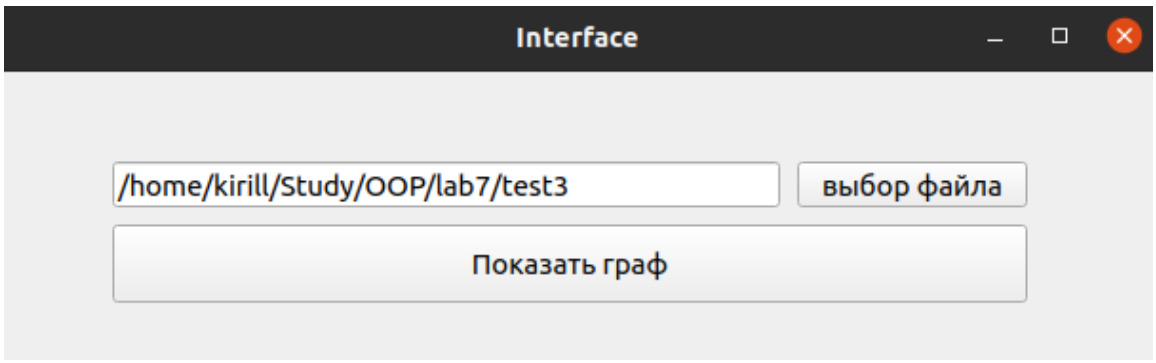

Form
1
2
3
4
5

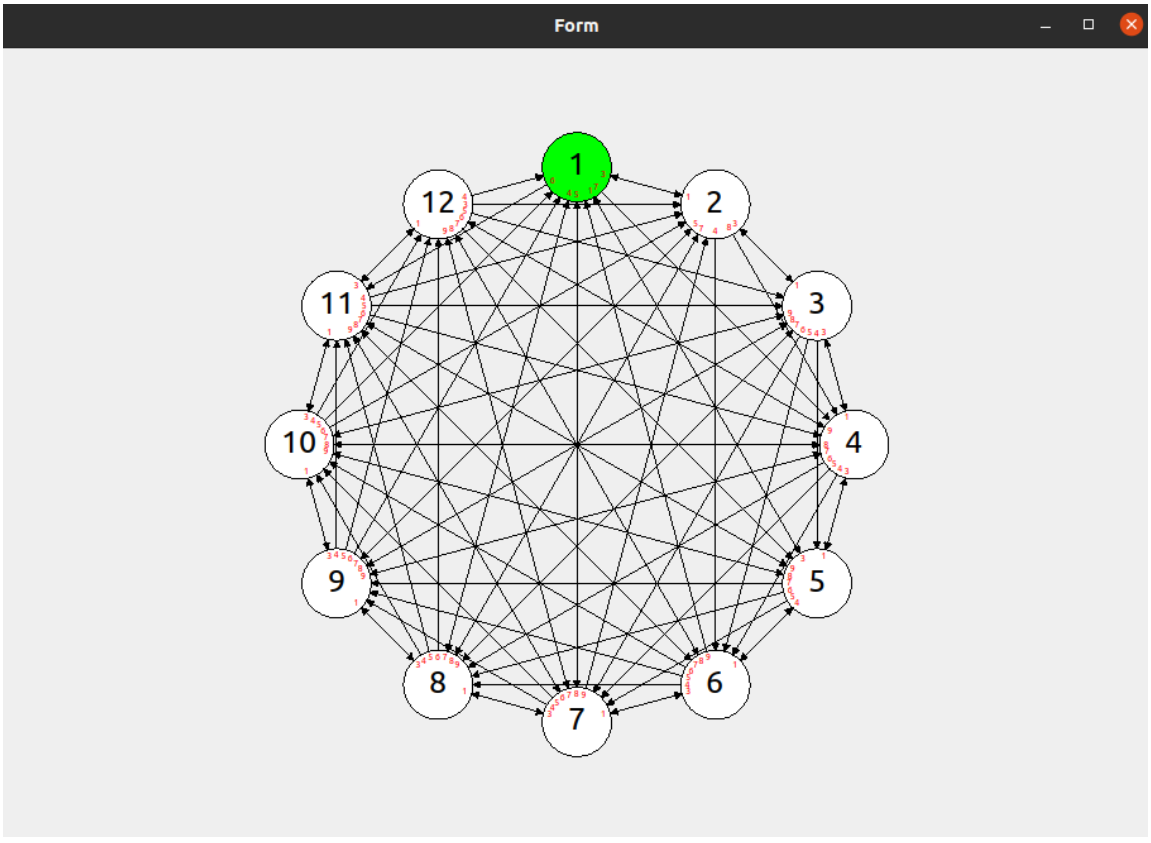

Form
1
2
3
4
5

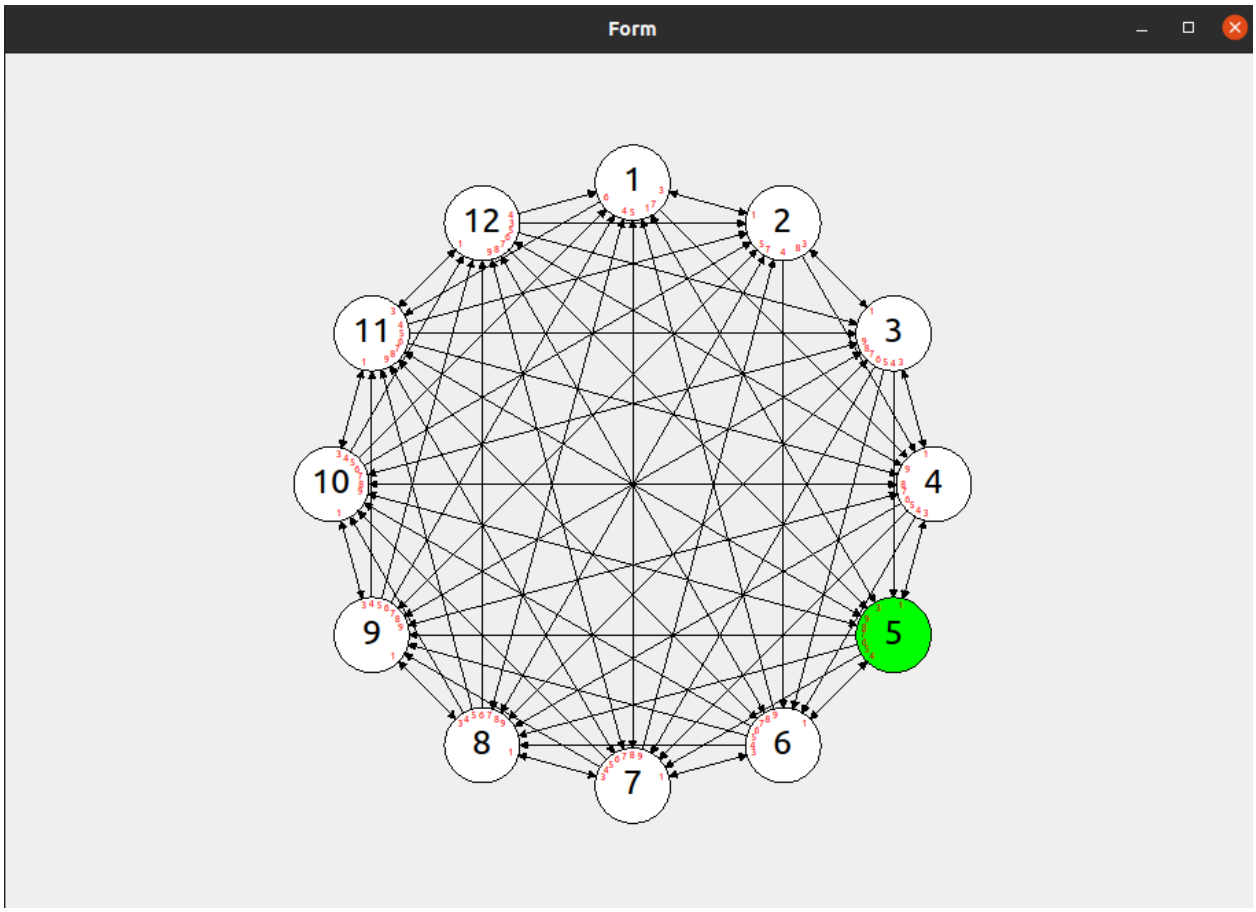
Пример 3:

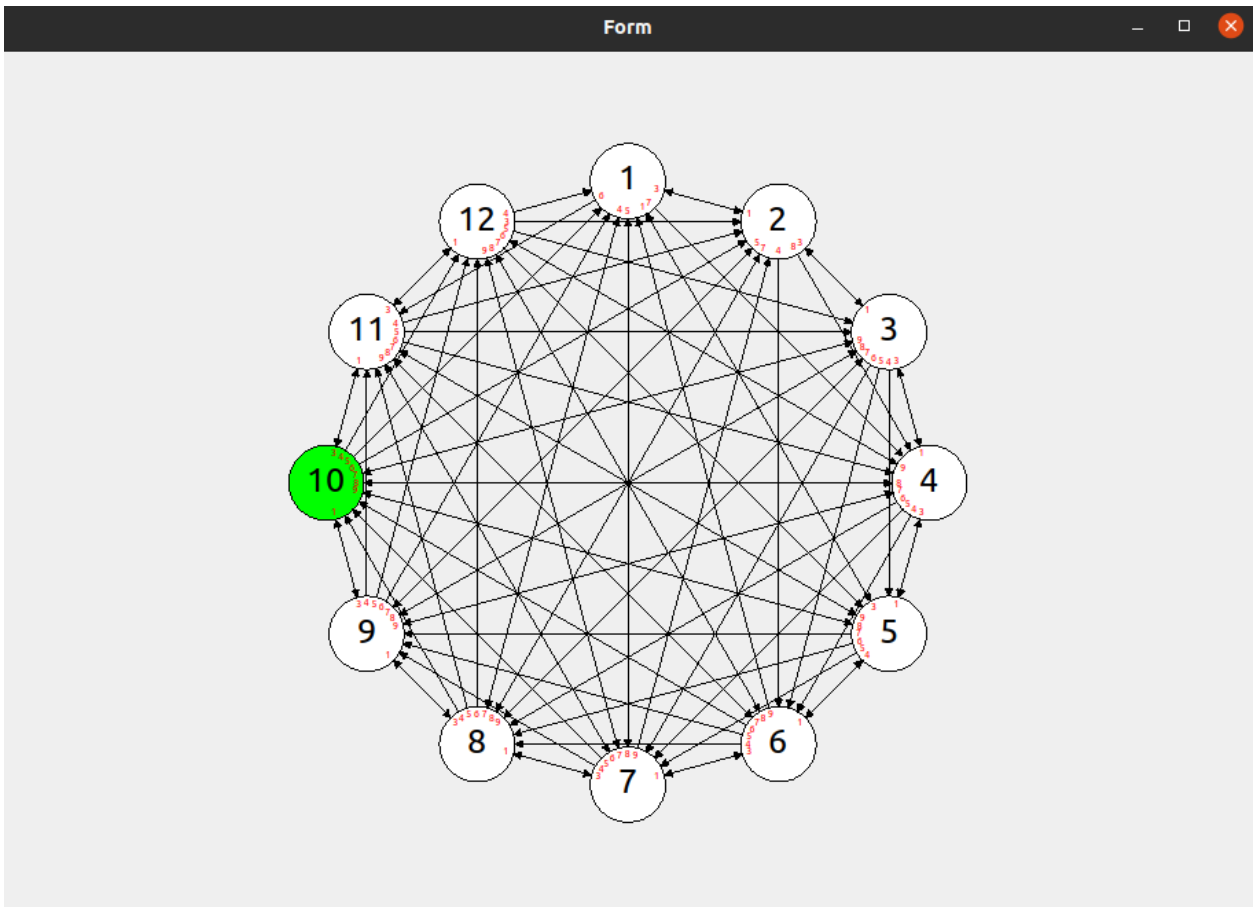
test3

```
12 9
4 0 1 7 6 10 3 0 0
0 1 2 5 8 1 7 3 1
1 2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 11
3 4 0 5 6 7 8 9 10
4 5 6 7 8 9 10 11 0
5 6 7 8 9 10 11 0 1
6 7 8 9 10 11 0 1 2
7 8 9 10 11 0 1 2 3
8 9 10 11 0 1 2 3 4
9 10 11 1 2 3 4 5 6
10 11 1 0 2 3 4 5 6
```






Form
1
2
3
4
5
6
7
8
9
10
11
12


Form
1
2
3
4
5
6
7
8
9
10
11
12

Вывод

В ходе данной лабораторной работы было создано GUI приложение, выполняющее функцию визуализации ориентированного графа, задаваемого матрицей смежности, представленной в виде файла. При корректности данных создается объект класса "Граф состояний", устанавливаются (если необходимо) связи между новым объектом и существующими, после чего граф отображается в соответствующем окне (объект класса "Окно представления графа"). Активная вершина помечается цветом. При смене значения номера активной вершины происходят изменения в отображении. На рис.1 представлен макет диаграммы классов приложения, который требуется реализовать в приложении.

Помимо этого, была создана диаграмма классов(рис.2) а также произведена отладка работы программы. Разработаны контрольные примеры с исходными и ожидаемыми данными, которые затем были протестированы в созданном GUI приложении. Все результаты совпали.