Санкт-Петербургский Государственный Электротехнический Университет
«ЛЭТИ» им. В. И. Ульянова (Ленина)

Кафедра информационных систем

Отчет

По практической работе №7

По дисциплине «Объектно-ориентированное программирование»

Студенты группы 2372 _____ Тубшинов В. Т., Алексеев Г.

Преподаватель _____ Егоров С. С.

г. Санкт-Петербург

2023 г.

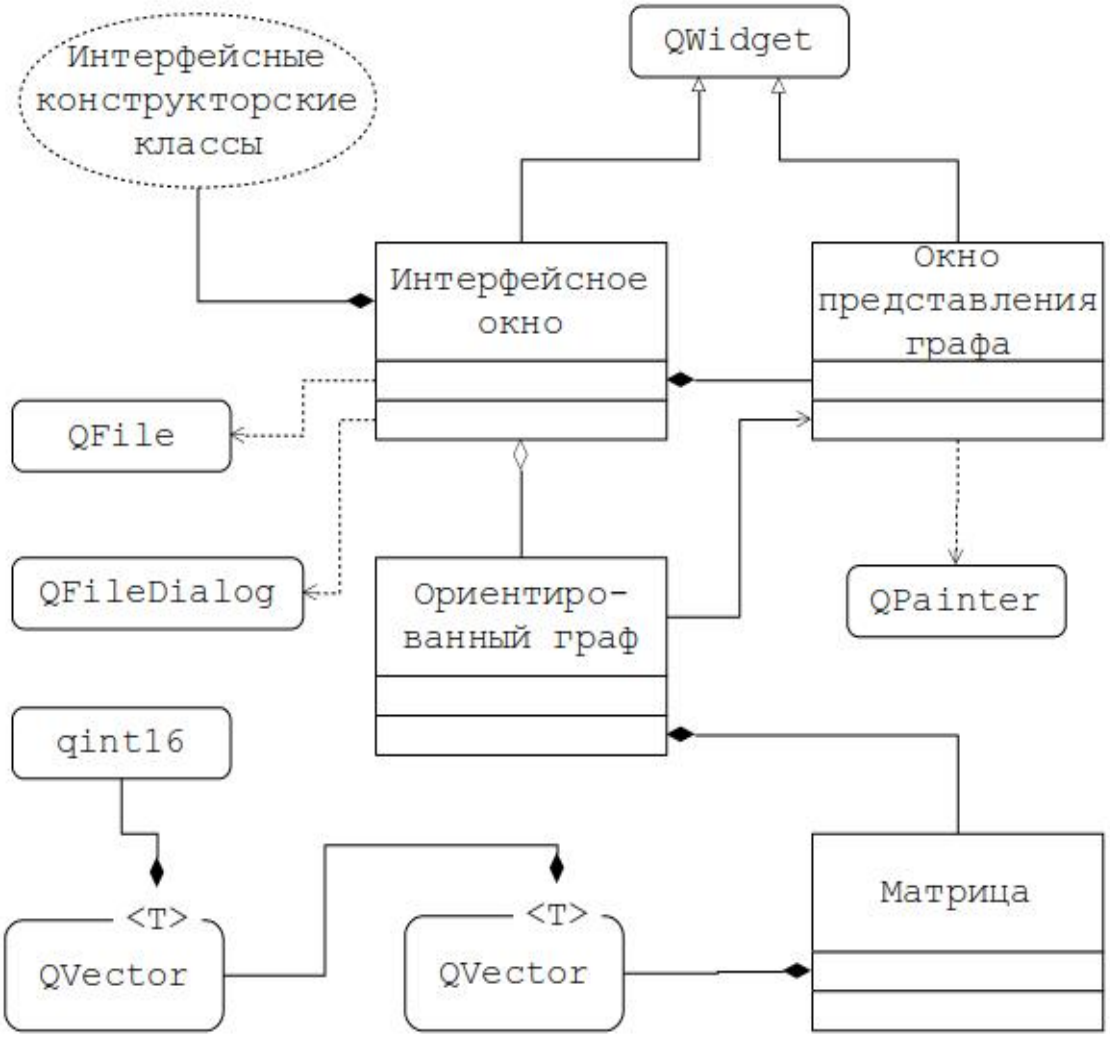


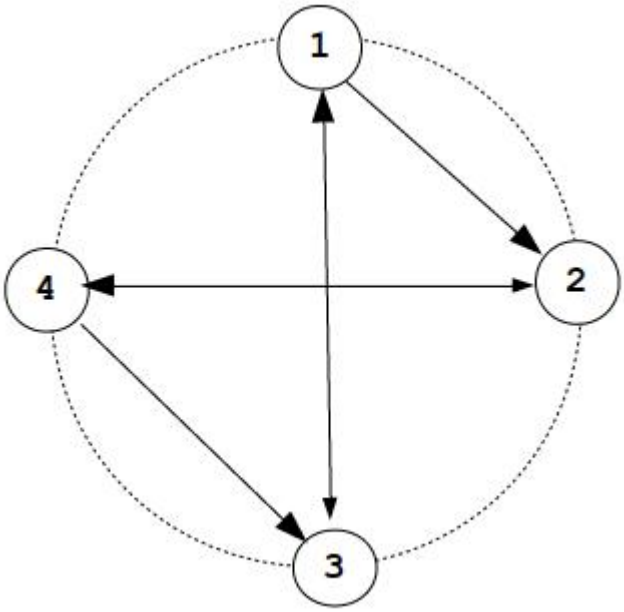
Рис.7. Диаграмма классов работы №7

Разработать GUI приложение, выполняющее функцию визуализации ориентированного графа, задаваемого матрицей смежности, представленной в виде файла, структуру которого требуется разработать. На рис.6 представлен макет диаграммы классов приложения, который требуется реализовать в приложении.

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

При корректности данных создается объект класса "Ориентированный граф", устанавливаются (если необходимо) связи

между новым объектом и существующими, после чего граф отображается в соответствующем окне (объект класса "Окно представления графа"). Пример вида графа с 4 вершинами представлен ниже:



При выборе в интерфейсе другого графа (другого файла) старый должен заменяться на новый и 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.

Спецификации классов:

Класс Graph:

Атрибуты:

private:

vector<Node*> nodes - вектор содержащий вершины графа;

Методы:

public:

Graph() - конструктор класса;

vector<Node*> getNodes() const - возвращает ссылку на вершину;

int size() const - возвращает количество вершин;

bool empty() - возвращает 1 - если вершин нет и 0 - если есть;

void addNode(char) - метод добавления вершины;

void addEdge(Node*, Node*) - метод добавления ребра;

string selectFile() - выбор входного файла;

short InputFileValues(string) - метод считывания данных с файла;

Класс Interface:

Атрибуты:

private:

Q_OBJECT

QPushButton *ChangeBtnFile - кнопка для выбора файла;

Sample *sample;

Методы:

public:

explicit Interface(Sample*, QWidget *parent = nullptr);

~Interface() override - деструктор класса;

protected:

void paintEvent(QPaintEvent*) override - вывод на экран;

public slots:

void changeFile() - Выбор другого файла;

Класс Sample:

Атрибуты:

Graph *graph - граф;

Методы:

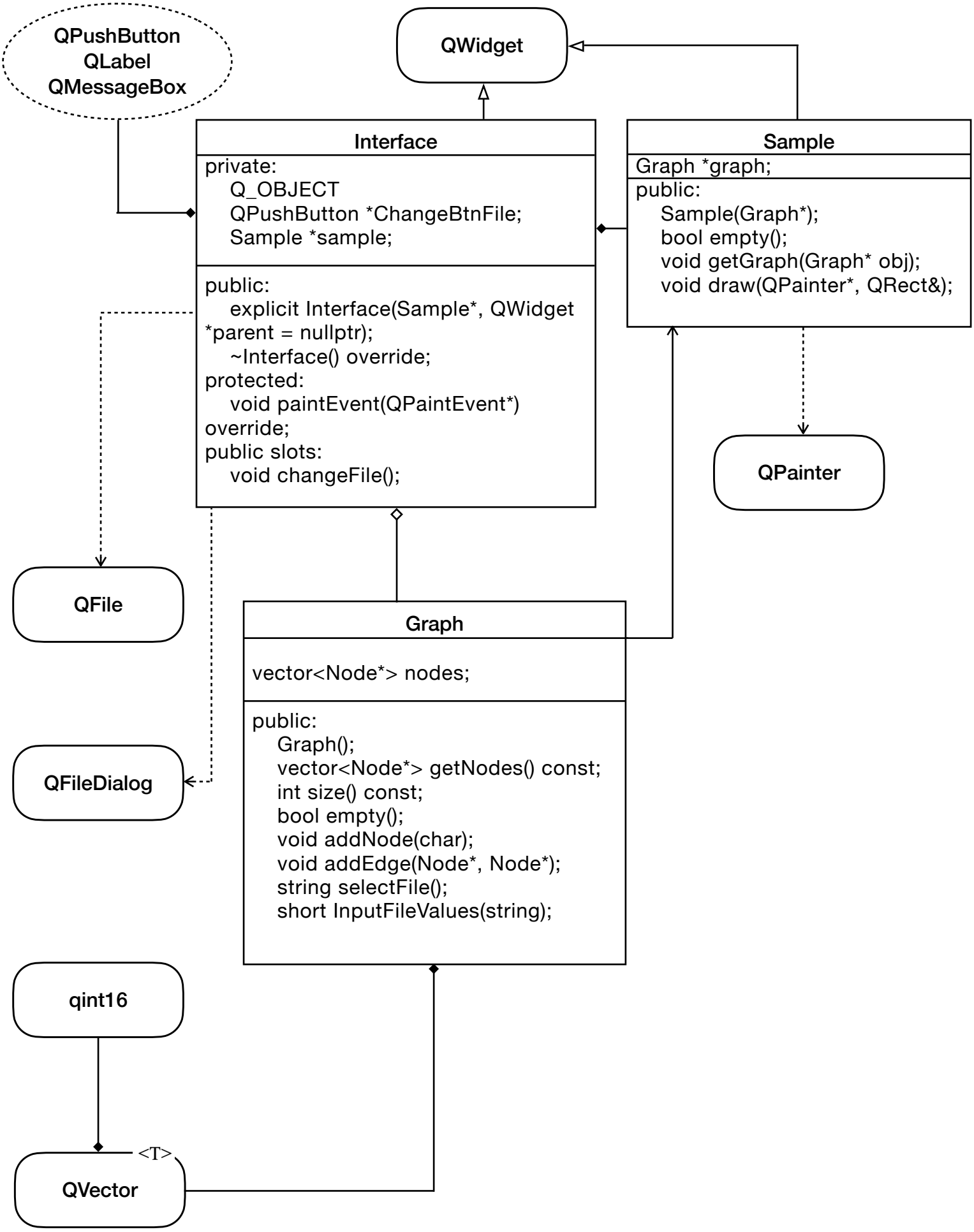
public:

Sample(Graph*) - конструктор класса;

bool empty() - возвращает 1 - если вершин нет и 0 - если есть;

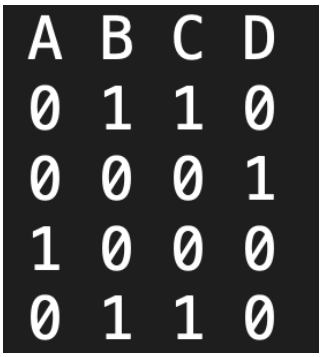
void getGraph(Graph* obj) - функция обращающаяся к атрибуту graph;

void draw(QPainter*, QRect&) - вывод графа на экран;


QPushButton
QLabel
QMessageBox
QWidget
Interface
private:
Q_OBJECT
QPushButton *ChangeBtnFile;
Sample *sample;
public:
explicit Interface(Sample*, QWidget *parent = nullptr);
~Interface() override;
protected:
void paintEvent(QPaintEvent*) override;
public slots:
void changeFile();
Sample
Graph *graph;
public:
Sample(Graph*);
bool empty();
void getGraph(Graph* obj);
void draw(QPainter*, QRect&);
QPainter
QFile
Graph
vector<Node*> nodes;
public:
Graph();
vector<Node*> getNodes() const;
int size() const;
bool empty();
void addNode(char);
void addEdge(Node*, Node*);
string selectFile();
short InputFileValues(string);
QFileDialog
qint16
<T>
QVector

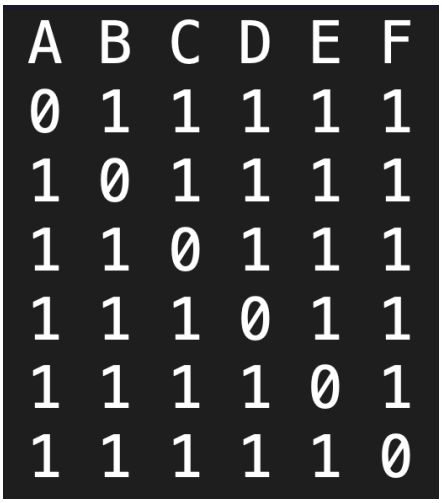
Входные файлы(матрицы смежности):

input1.txt



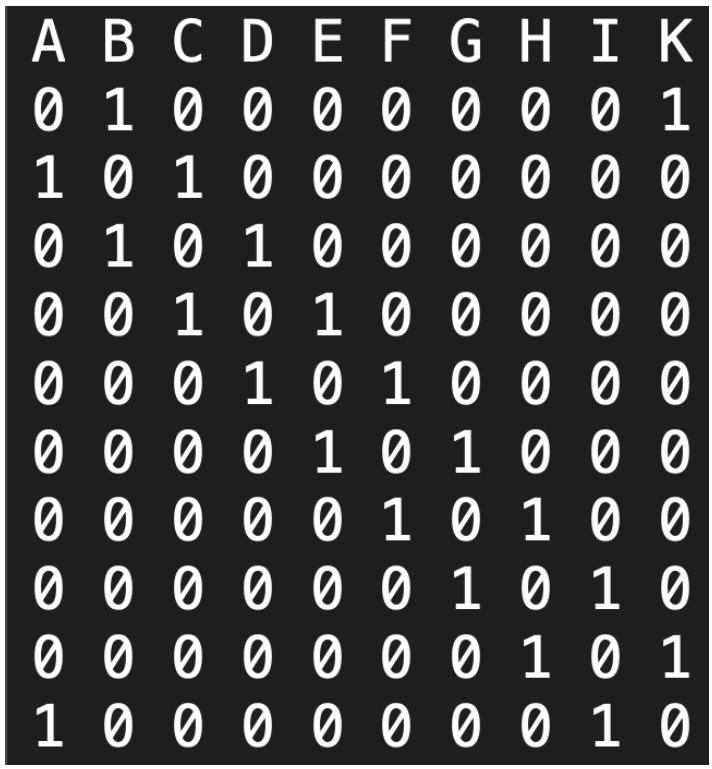
```
A B C D
0 1 1 0
0 0 0 1
1 0 0 0
0 1 1 0
```

input2.txt



```
A B C D E F
0 1 1 1 1 1
1 0 1 1 1 1
1 1 0 1 1 1
1 1 1 0 1 1
1 1 1 1 0 1
1 1 1 1 1 0
```

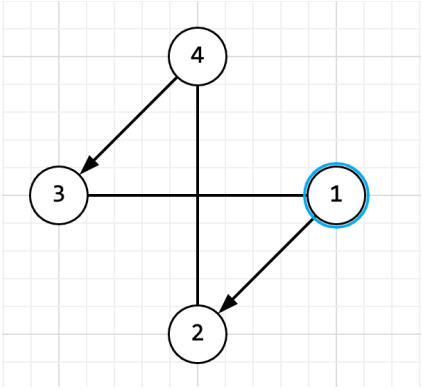
input3.txt



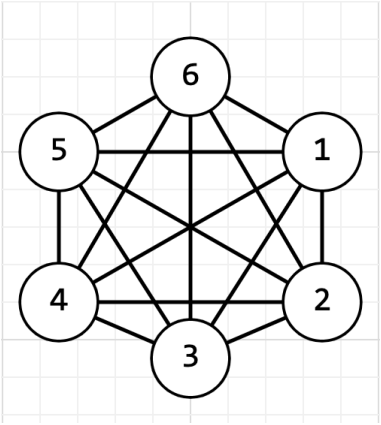
```
A B C D E F G H I K
0 1 0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 1 0 1
1 0 0 0 0 0 0 0 1 0
```

Ожидаемые данные:

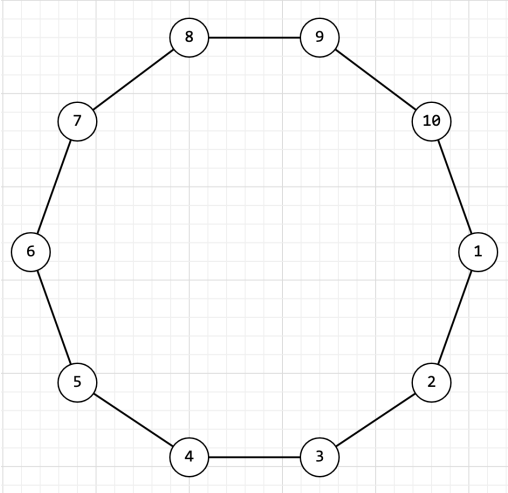
Для входного файла «input1.txt»



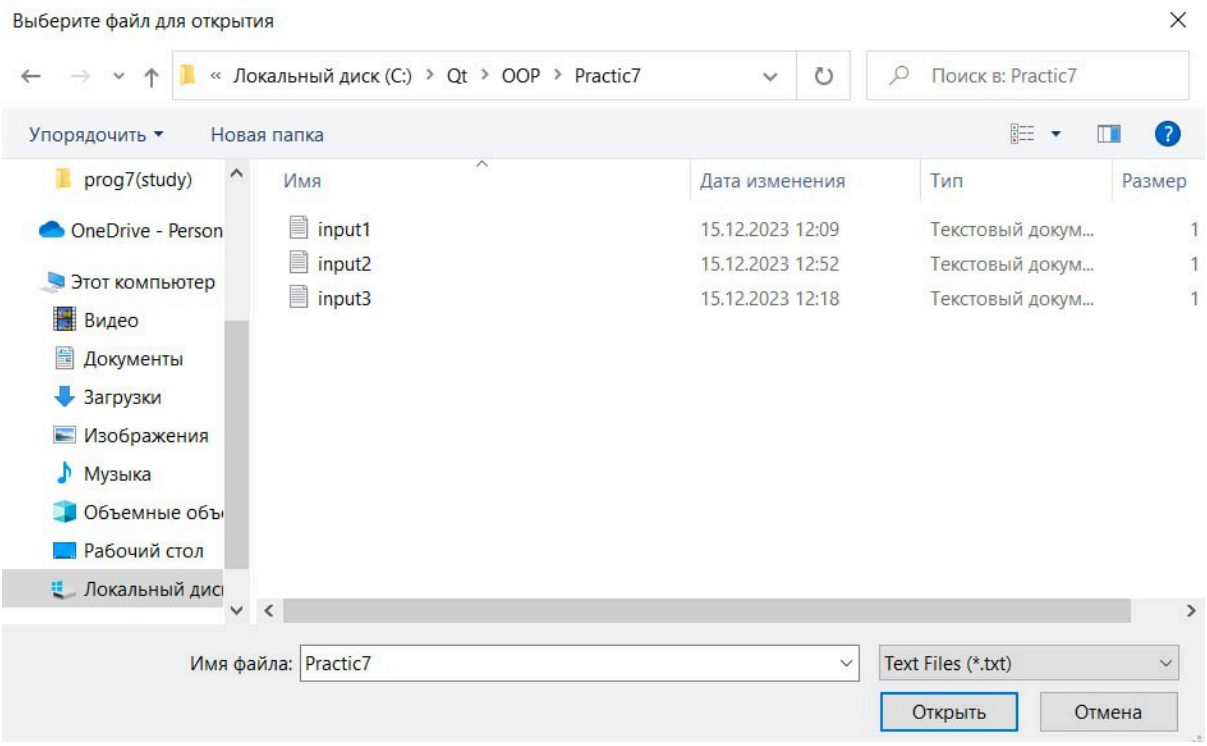
Для входного файла «input2.txt»



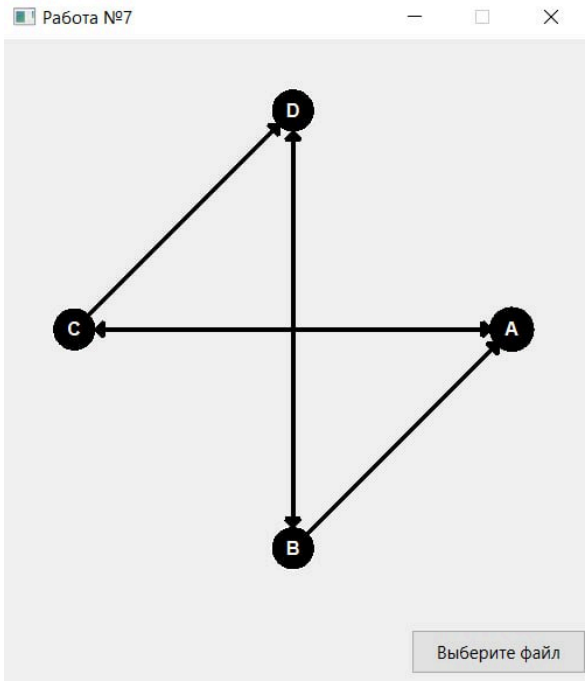
Для входного файла «input3.txt»



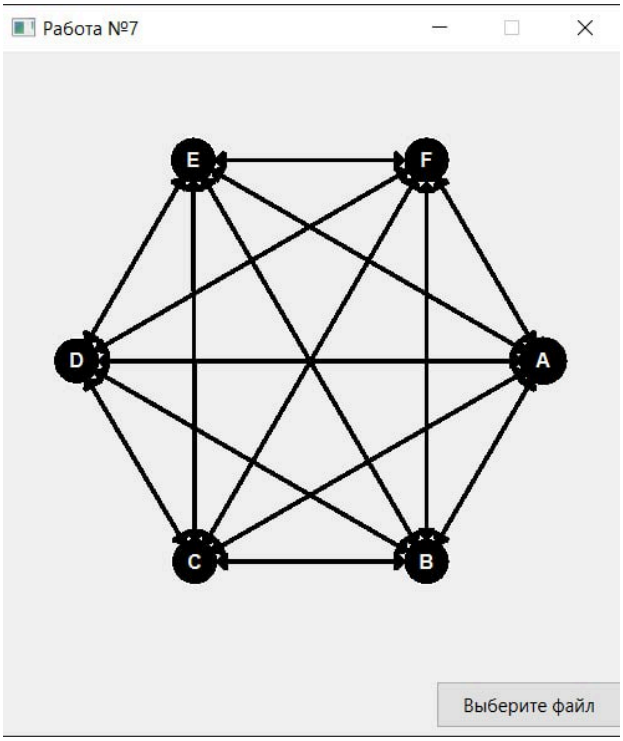
Работа программы:



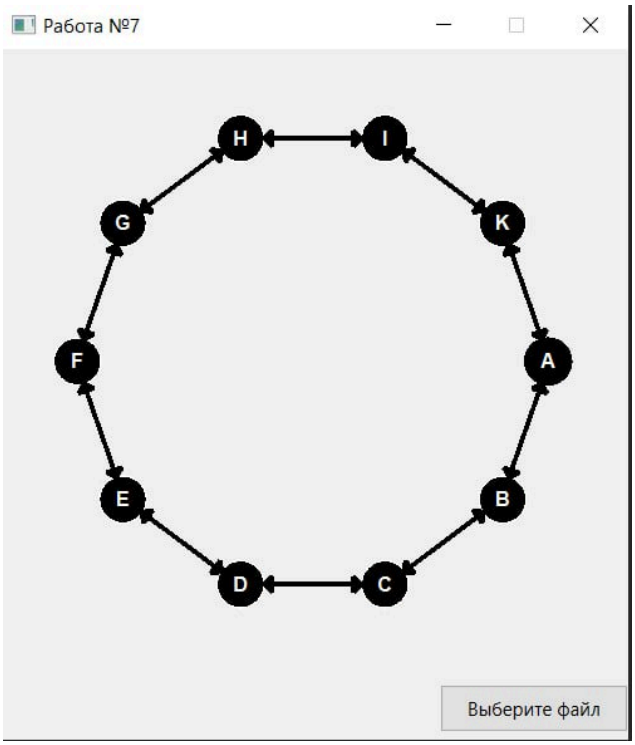
Для входного файла «input1.txt»



Для входного файла «input2.txt»



Для входного файла «input3.txt»



Вывод: В ходе работы было разработано GUI приложение для визуализации ориентированного графа. Пользователь выбирает файл с данными о графе, при чтении файла проверяется корректность данных. Если данные корректны, создаётся объект класса Граф, затем граф передаётся окну представления и визуализируется.