Алгоритмы для работы матрицой смежности.
© GitHub NiceOneFox

Оглавление

Введение. Общая постановка задачи

Вариант 8

В работе показан пример работы с графами посредством матрицы смежности, методов данного класса и функций, которые использует данные методы.

1. Реализовать представление ориентированного графа в виде матрицы смежности.
2. Реализовать представление ориентированного графа в виде списков смежных вершин.
3. Определить исходящую и входящую степени каждой вершины.
4. Определить вершины с максимальной степенью.
5. Удалить кратные ребра и циклы в мультиграфе.
6. Предусмотреть обработку и инициализацию исключительных ситуаций, связанных, например, с проверкой значения полей перед инициализацией и присваиванием.
7. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines (http://geosoft.no/development/cppstyle.html).

Требования

Нефункциональные

Требования к продукту

1. Программа должна работать на OS Windows
2. Программа не вызывает ошибок и предупреждений Visual Studio 2017

Организационные требования

1. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines
2. Срок выполнения работы не должен превышать 2 месяцев
3. Интерфейс должен быть прост и понятен пользователю
    - Интерфейс должен быть на русском языке

Функциональные

1. Требования к файлу:
    - Путь к файлу должен быть указан верно
    - Файл не должен быть пустым
    - В файле первое число указывает на размер матрицы
    - В файле размещена готовая матрица смежности либо пары чисел, обозначающих смежные вершины
    - В файле отсутствуют посторонние символы или информация
2. Требования к вводу информацию с клавиатуры
    - Пользователь вводит число 1 или 2, обозначающие способ работы с входными данными.
    - Пользователь вводит названия файла с входными данными
3. Требования к функциям программы
    - Построение матрицы смежности на основе входных данных
    - Создания списка смежных вершин
    - Поиск входящей и исходящей степени каждой вершины матрицы
    - Поиск вершины с максимальной степенью
    - Удаление кратных рёбер в мультиграфе
    - Удаление рёбер цикла в мультиграфе
4. Требования к выходным данным
    - Построение таблиц
    - Оформление заголовков колонок и столбцов

Спецификации

№ спец.	Входные данные	Результат
1	Неверно указан путь к файлу или файл отсутствует	Сообщение: Название файла «Файл не открыт»
1.1	Файл пустой	Сообщение: Название файла «Файл пустой»
2	Пользователь ввёл число отличное от 1 или 2, обозначающие способ работы с входными данными	Сообщение: "Ошибка! Введите число 1 или 2"
3	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Вывод в выходной файл исходной матрицы смежности
3.1	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Вывод в выходной файл списка смежных вершин
3.2	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Вывод в выходной файл исходящей и входящей степени каждой вершины
3.3	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Вывод в выходной файл Вершин с максимальной степенью
3.4	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Удаление кратных рёбер
3.5	Указан допустимый способ работы с входными данными и название файла, содержащего размер матрицы и соответствующие представление матрицы	Удаление рёбер цикла в мультиграфе

Тест план с результатами тестов

№ теста	№ спеп.	Входные данные	Ожидаемый результат	Подтверждение
1	1	cin >> fileIn; fileIn = inpfile.txt	Сообщение: inpfile.txt «Файл не найден»	+
2	1	cin >> fileIn; fileIn = inputFileText.txt	Сообщение: inputFileText.txt «Файл не найден»	+
3	1.1	*пустой файл* input.txt	Сообщение: input.txt «Файл пустой»	+
4	1.1	*пустой файл* input2.txt	Сообщение: input.txt «Файл пустой»	+
5	2	Cin>>method; Method = 6	Сообщение: "Ошибка! Введите число 1 или 2"	+
6	3	5 1 0 1 1 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 0	=Adjacency Matrix= No| 0 1 2 3 4 ========= 0 | 0 0 1 1 0 | 1 | 1 0 1 0 1 | 2 | 0 0 0 0 1 | 3 | 0 0 0 0 1 | 4 | 0 0 0 0 0 | ==========	+
7	3	5 2 4 4 4 3 0 2 4 0 1 2 3 4 1 2 3 4 4 2 4 4 0 0 2 2 1 1 1	=Adjacency Matrix= No| 0 1 2 3 4 ========= 0 | 0 1 1 0 0 | 1 | 0 0 0 0 0 | 2 | 0 1 0 1 1 | 3 | 1 0 0 0 0 | 4 | 1 1 0 0 0 | ==========	+

8	3.1	5 1 0 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 0	=AdjacencyList= from to | 0 | 2 3 | 1 | 0 2 4 | 2 | 3 4 | 3 | 4 ==========	+
9	3.1	5 2 4 4 3 3 0 2 4 0 1 2 3 4 1 2 3 4 0 2 4 4 0 0 2 2 1 1 1	=AdjacencyList= from to | 0 | 1 2 | 2 | 1 3 4 | 3 | 0 | 4 | 0 1 3 ==========	+
10	3.2	5 1 0 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 0	=PowerVertex= No |0 1 2 3 4 In |1 0 2 2 3 Out|2 3 2 1 0 ==========	+
11	3.2	5 2 4 4 3 3 0 2 4 0 1 2 3 4 1 2 3 4 0 2 4 4 0 0 2 2 1 1 1	=PowerVertex= No |0 1 2 3 4 In |2 3 1 2 1 Out|2 0 3 1 3 ==========	+
12	3.3	5 1 0 1 1 0 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 1 0 1 0	=MaxVertexPower= Power|Vertex | 3| 2 | ==========	+

13	3.3	5 2 4 4 3 3 0 2 4 0 1 2 3 4 1 2 3 4 0 2 4 4 0 0 2 2 1 1 1	=MaxVertexPower= Power|Vertex | 4| 0 2 4 | =========	+
14	3.4	5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	=Adjacency Matrix= No| 0 1 2 3 4 ========== 0 | 0 1 1 1 1 | 1 | 0 0 1 1 1 | 2 | 0 0 0 1 1 | 3 | 0 0 0 0 1 | 4 | 0 0 0 0 0 | ==========	+
15	3.4	5 1 1 4 3 3 0 2 4 0 1 2 3 4 1 2 3 4 0 2 4 4 0 0 2 2 1 1 1	=Adjacency Matrix= No| 0 1 2 3 4 ========== 0 | 0 1 1 0 0 | 1 | 0 0 0 0 0 | 2 | 0 1 0 1 1 | 3 | 1 0 0 0 0 | 4 | 1 1 0 1 0 | ==========	+
16	3.5	5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	=Adjacency Matrix= No| 0 1 2 3 4 ========== 0 | 0 1 1 1 1 | 1 | 1 0 1 1 1 | 2 | 1 1 0 1 1 | 3 | 1 1 1 0 1 | 4 | 1 1 1 1 0 | ==========	+

Описание алгоритмов

Алгоритм поиска входящей и исходящей степени вершин

```
INOUTPOWERVERTEX( graph )
For i <-1 to vertexNumber_ do
  (i)[inPowerVertex]graph[push_back]InPower
  (i)[outPowerVertex]graph[push_back]Outpower
Return InPower, OutPower;

INPOWERVERTEX( NoVertex )
Count <- 0
For i <- 1 to vertexNumber_ do
  If ( [i][NoVertex]graph = 1 )
    count <- count + 1
return count;

OUTPOWERVERTEX( NoVertex )
Count <- 0
For j <- 1 to vertexNumber_ do
  If ( [NoVertex][j]graph = 1)
    count <- count + 1
return count;
```

Алгоритм поиска максимальной степени вершины

```
VERTEXMAXPOWER(graph)
For i <- 1 to n do
  Max <- (i)[inPowerVertex]graph + (i)[outPowerVertex]graph
  If ( max = prevMax)
    (i)[push_back]NoMaxVertex
  If (max > prevMax)
    [clear]NoMaxVertex
    prevMax <- (i)[inPowerVertex]graph + (i)[outPowerVertex]graph
return max, NoMaxVertex;
```

Алгоритм удаления удаления рёбер цикла в мультиграфе

```
DELETECIRCLEVERTEXES()
j <- 0
for i <- 1 to vertexNumber_ do
  if ([i][j]graph = true)
    [i][j]graph <- false
  j <- j + 1
```

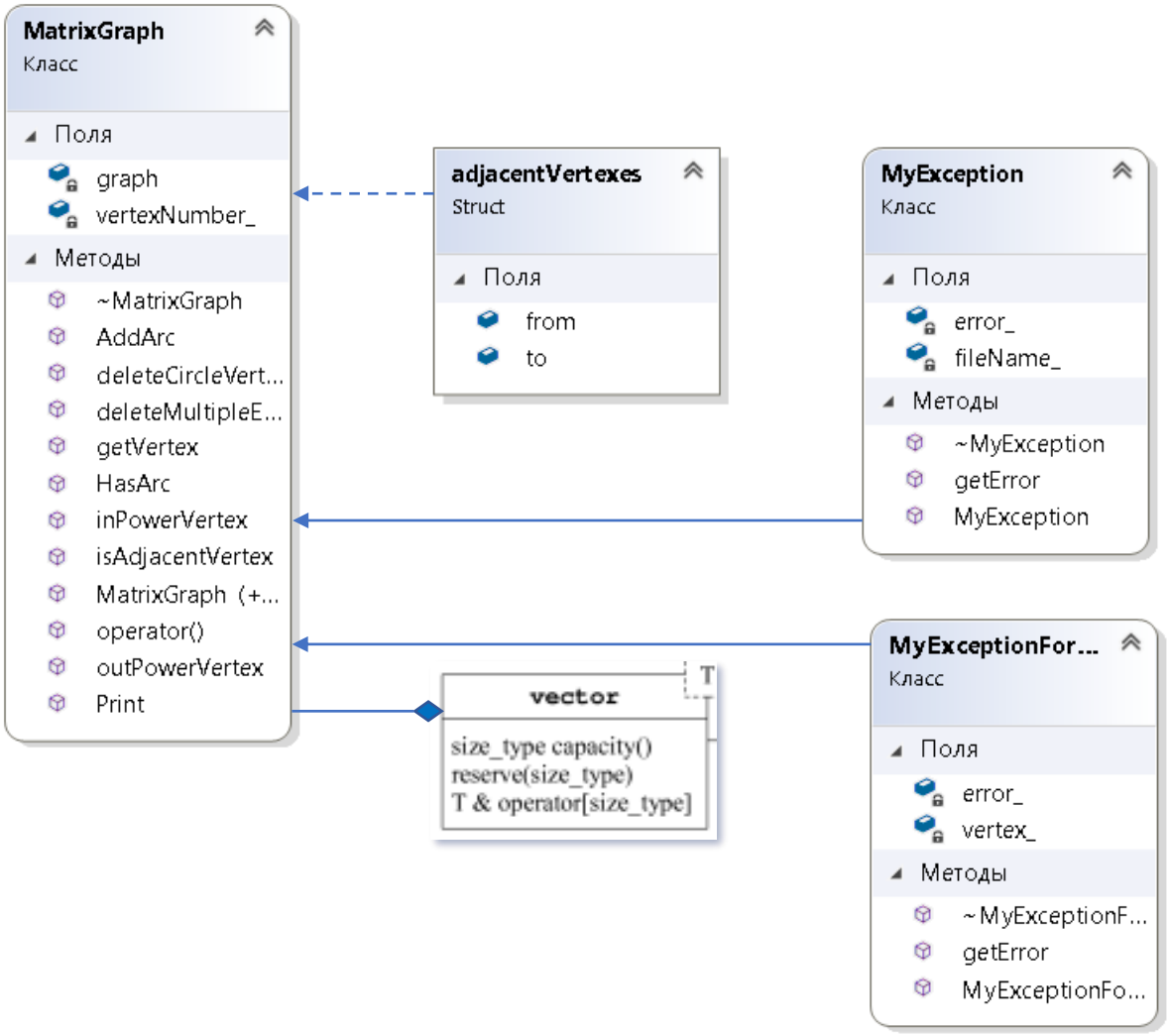
Алгоритм удаления кратных рёбер в мультиграфе

```
DELETEMULTIPLEEDGE()
For i <- vertexNumber_ -1 downto 0
  If [i][j]graph = [j][i]graph = 1
    [j][i]graph <- 0;
```

Алгоритм построения списка смежных вершин

```
LISTADJACENTVERTEXES(graph, adjVert)
count <- 0
n <- [getVertex]graph
isAdjacent <- false
for i <- 1 to n do
  isAdjacent <- false
  for i <- 1 to n do
    if ( (i, j)[isAdjacentVertexes]graph )
      isAdjacent <- true;
      [from][count]adjVert <- i
      (j)[push_back][to][count]adjVert
   If (isAdjacent = true)
     [from][count]adjVert <- i
     count <- count + 1
```

Диаграмма классов



В программе используются классы из стандартной библиотеки: iostream, fstream, string, iomanip.

Классы MyException и MyExceptionForVertex имеют тип отношения с классом MatrixGraph “спецификация и её выполнение”.

Структура AdjacentVertexes имеет поле to, использующие класс vector.

Заключение

На основе задания был разработан класс MatrixGraph в которым хранится матрица смежности и методы работы с ней. Так же разработаны функции использующие методы класса и перегрузку одного из операторов. Были созданы два собственные класса исключения, которые помогают избежать ошибок программы и упростить их поиск.

По заданию была создана матрица смежности, определена входящая и исходящая степень каждой вершины, построен список смежности, найдены вершины с максимальной степенью, удалены циклы и кратные рёбра в мультиграфе. Результат выведен в виде таблицы в выходной текстовый файл.

Разработанные алгоритмы были описанный на псевдокоде. Описаны требования, спецификации и тест план к работе. В ходе работы были изучены алгоритмы для работы с графами с помощью матрицы смежности.

Список литературы

1. Кормен Т. и др. Алгоритмы: построение и анализ: М. [и др.]: Вильямс,2011.

2. Седжвик Р., Моргунов А.А. Алгоритмы на C++. Анализ, структуры данных, сортировка, поиск, алгоритмы на графах: М.: Вильямс, 2011.

3. Лекции и и упражнения по курсу "Алгоритмы и структуры данных": https://www.intuit.ru/studies/courses/3496/738/info

Приложение 1

MatrixGraph.h

```
#ifndef MATRIX_GRAPH_H
#define MATRIX_GRAPH_H

class MatrixGraph
{
public:
	MatrixGraph() {}

	MatrixGraph(int n);

	MatrixGraph(const MatrixGraph &other);

	~MatrixGraph();

	int getVertex() const {	return vertexNumber_; }

  void AddArc(int from, int to);

  bool HasArc(int from, int to) const;

	void Print();

	bool isAdjacentVertex(int from, int to);

	int  inPowerVertex (int NoVertex);

	int  outPowerVertex(int NoVertex);

	void deleteCircleVertexes();

	void deleteMultipleEdge();

	bool& operator()(const int i, const int j) { return graph[i][j];	}

private:
	int vertexNumber_;
	bool **graph;
};

struct adjacentVertexes
{
	int from;
	vector <int> to;
};

void listAdjacentVertexes(MatrixGraph & graph, adjacentVertexes *adjVert);

void vertexMaxPower(MatrixGraph & graph);
```

```
void outputVertexMaxPower(MatrixGraph graph, const int max, std::vector<int>
NoMaxVertex);

void inputMatrixGraph  (std::vector<MatrixGraph> & arr, const string & fileIn);
void inputGraphAsMatrix(std::vector<MatrixGraph> & arr, const string & fileIn);

void inOutPowerVertex(MatrixGraph graph);
void outputInOutPowerVertex(MatrixGraph graph, std::vector<int> InPower,
std::vector<int> OutPower);

void outputMatrixGraph(MatrixGraph graph);
#endif
```

Pch.h

```
#ifndef PCH_H
#define PCH_H

#include <iostream>
#include <fstream>

#include <string>
#include <vector>

#include <iomanip>

using namespace std;


#endif
```

MyException.h

```
#ifndef MY_EXCEPTION
#define MY_EXCEPTION
#include "pch.h"
class MyException
{
public:
	MyException(string error, string fileName) : error_(error), fileName_(fileName) {}


	~MyException() {}

	string getErrorFile()
	{
		cout << endl << fileName_ << " " << error_ << endl;
		return "";
	}
private:
	string error_;
	string fileName_;
}
#endif
```

MyEceptionForVertex.h

```
#ifndef MY_EXCEPTION_FOR_VERTEX
#define MY_EXCEPTION_FOR_VERTEX
#include "pch.h"
class MyExceptionForVertex
{
public:
	MyExceptionForVertex(int vertex, string error) : vertex_(vertex), error_(error) {}
	~MyExceptionForVertex() {}

	string getError()
	{
		cout << endl << "Вершина с номер " << vertex_ << error_ << endl;
		return "";
	}

private:
	int vertex_;
	string error_;
};

#endif
```

MatrixGraph.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyExceptionForVertex.h"

MatrixGraph::MatrixGraph(int n)
{
	graph = new bool*[vertexNumber_ = n];
	for (int i = 0; i < n; i++) {
		bool *row = graph[i] = new bool[n];
		for (int j = 0; j < n; j++) {
			row[j] = false;
		}
	}
}

MatrixGraph::MatrixGraph(const MatrixGraph &other)
{
	vertexNumber_ = other.vertexNumber_;
	int n = other.getVertex();

	graph = new bool*[n];
	for (int i = 0; i < n ; i++)
		graph[i] = new bool[n];

	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			graph[i][j] = other.graph[i][j];
```

```
                }
        }
}

MatrixGraph::~MatrixGraph()
{
        for (int i = 0; i < vertexNumber_; i++) {
                delete graph[i];
        }
        delete graph;
}

void MatrixGraph::AddArc(int from, int to)
{
        if (from < 0 || from >= vertexNumber_)
          throw MyExceptionForVertex(from, " выходит за размеры матрицы");
        if (to < 0 || to >= vertexNumber_)
                throw MyExceptionForVertex(to, " выходит за размер матрицы");

        if (graph[from][to] == true)
                return;

        graph[from][to] = true;
}

bool MatrixGraph::HasArc(int from, int to) const
{
        if (from < 0 || from >= vertexNumber_ || to < 0 || to >= vertexNumber_)
                return false;
        return graph[from][to];
}

void MatrixGraph::Print()
{
        int n = getVertex();
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        cout << (HasArc(i, j) ? '1' : '0') << " ";
                }
                cout << endl;
        }
}

bool MatrixGraph::isAdjacentVertex(int from, int to)
{
        if (graph[from][to] == true)
                return true;

        return false;
}

int MatrixGraph::inPowerVertex(int NoVertex)
```

```
{
	int count = 0;
	for (int i = 0; i < vertexNumber_; i++) {
		if (graph[i][NoVertex] == 1)
			count++;
	}

	return count;
}

int MatrixGraph::outPowerVertex(int NoVertex)
{
	int count = 0;
	for (int j = 0; j < vertexNumber_; j++) {
		if (graph[NoVertex][j] == 1)
			count++;
	}

	return count;
}

void MatrixGraph::deleteCircleVertexes()
{
	int n = vertexNumber_;
	int j = 0;
	for (int i = 0; i < n; i++) {
		if (graph[i][j] == true)
			graph[i][j] = false;
		j++;
	}
}

void MatrixGraph::deleteMultipleEdge()
{
	for (int i = vertexNumber_-1; i >= 0; i--) {
		for (int j = 0; j < vertexNumber_; j++) {
			if (graph[i][j] == graph[j][i] == 1)
				graph[i][j] = 0;
	  }
	}
}
```

inputMatrixGraph.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyException.h"

void inputMatrixGraph(std::vector<MatrixGraph> & arr, const string & fileIn)
{
	ifstream fin;
	fin.open(fileIn, ios::binary);
	if (!fin.is_open())
```

```
		throw MyException("Файл не открыт", fileIn); // возможно добавить
переменную с именем файла  throw
	else {
		if (fin.peek() == EOF)
			throw MyException("Файл пустой", fileIn);
		int from;
		int to;
		int n;
		if (fin >> n){ // считываем размер матрицы
			arr.push_back(MatrixGraph(n));
		  while (!fin.eof()) {
				fin >> from;
				fin >> to;
				arr.back().AddArc(from, to);
			}
		}
	}
	fin.close();
}

void inputGraphAsMatrix(std::vector<MatrixGraph> & arr, const string & fileIn)
{
	ifstream fin;
	fin.open(fileIn, ios::binary);
	if (!fin.is_open())
		throw MyException("Файл не открыт", fileIn);
		if (fin.peek() == EOF)
			throw MyException("Файл пустой", fileIn);
		int n;
		if (fin >> n) { // считываем размер матрицы
			MatrixGraph M(n);
			for (int i = 0; i < n; i++){
				for (int j = 0; j < n; j++){
					fin >> M(i,j);
				}
			}
		arr.push_back(M);
		}
		fin.close();
	}
}
```

OutputMatrixGraph.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyException.h"
void outputMatrixGraph(MatrixGraph graph)
{
	int n = graph.getVertex();
	ofstream fout("output.txt", ios::out);
	if (!fout.is_open())
		throw MyException("Файл не найден", "output.txt");
```

```
	else {
		fout << "=Adjacency Matrix=" << endl;
		fout << "No| ";
		for (int i = 0; i < n; i++) { // вывод нумерации вершин
			fout << i << " ";
		}
		fout<< endl << "  "; // вывод верхней границы таблицы
		for (int i = 0; i < (n * 2) + 3; i++) {
			fout << "=";
		}
		fout << endl;
		for (int i = 0; i < n; i++){
		fout << i << " | ";
			for (int j = 0; j < n; j++){
				fout << graph(i, j) << " ";
			}
			fout << "|" << endl;
		}
		fout << "  "; // вывод нижней границы таблицы
		for (int i = 0; i < (n*2)+3; i++) {
			fout <<"=";
		}
		fout << endl;
		fout.close();
	}
}
```

listAdjacentVertexes.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
void listAdjacentVertexes(MatrixGraph & graph, adjacentVertexes *adjVert)
{
	int count = 0; // кол-во элементов в списке
	int n = graph.getVertex();
	bool isAdjacent = false;

	for (int i = 0; i < n; i++) {
		isAdjacent = false; // найдена хотя бы одна смежная вершина значит
размер списка увеличился
		for (int j = 0; j < n; j++){
			if (graph.isAdjacentVertex(i, j)) {

				isAdjacent = true;
				adjVert[count].from = i;
				adjVert[count].to.push_back(j);
			}
		}
		if (isAdjacent == true) {
			adjVert[count].from = i;
			count++;
		}
	}
```

```
	ofstream fout("output.txt", ios::app);
	if (!fout.is_open())
		throw "Файл не найден";
	else {
		fout << "=AdjacencyList=" << endl;
		fout << " from" << "   to   " << endl;
		for (int i = 0; i < count; i++) {
			fout << "|" << setw(4) << right << adjVert[i].from << " | ";
			for (vector<int>::iterator it = adjVert[i].to.begin(); it != 
adjVert[i].to.end(); ++it)
				fout << *it << " ";
			//fout << "|";
	fout << endl;
		}
		fout << "=============\n";
		fout.close();
	}
}
```

outputInOutPowerVertex.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyException.h"
void inOutPowerVertex(MatrixGraph graph)
{
	int n = graph.getVertex();
	vector <int> InPower;
	vector <int> OutPower;
	for (int i = 0; i < n; i++) {
		InPower.push_back(graph.inPowerVertex(i));
		OutPower.push_back(graph.outPowerVertex(i));
	}

	outputInOutPowerVertex(graph, InPower, OutPower);
}

void outputInOutPowerVertex(MatrixGraph graph, std::vector<int> InPower, 
std::vector<int> OutPower)
{
	int n = graph.getVertex();
	ofstream fout("output.txt", ios::app);
	if (!fout.is_open())
		throw MyException("Файл не найден", "output.txt");
	else {
		fout << "=PowerVertex=\n" << "No |";
		for (int i = 0; i < n; i++) {
			fout << i << " ";
		}
		fout << endl << "In |";
```

```
		for (vector<int>::iterator it = InPower.begin(); it != InPower.end(); ++it)
			fout << *it << " ";
		fout << endl << "Out|";
		for (vector<int>::iterator it = OutPower.begin(); it != OutPower.end(); ++it)
			fout << *it << " ";

		fout << "\n====";
		for (int i = 0; i < n * 2; i++) {
			fout << "=";
		}
		fout << endl;
		fout.close();
	}
}
```

outputVertexMaxPower.cpp

```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyException.h"

void vertexMaxPower(MatrixGraph & graph)
{
	vector<int> NoMaxVertex;
	int max = -1;
	int prevMax = 0;
	int n = graph.getVertex();
	for (int i = 0; i < n; i++) {
		max = graph.inPowerVertex(i) + graph.outPowerVertex(i);

		if (max == prevMax) {
			NoMaxVertex.push_back(i);
		}

		if (max > prevMax){
			NoMaxVertex.clear(); // если новая макс степень очищаем вектор
			prevMax = graph.inPowerVertex(i) + graph.outPowerVertex(i);
			NoMaxVertex.push_back(i);
		}

	}
	outputVertexMaxPower(graph, max, NoMaxVertex);
}

void outputVertexMaxPower(MatrixGraph graph, const int max, std::vector<int>
NoMaxVertex)
{
	int n = graph.getVertex();
	ofstream fout("output.txt", ios::app);
	if (!fout.is_open())
		throw MyException("файле не найден", "output.txt");
	else {
		fout << "=MaxVertexPower=\n";
```

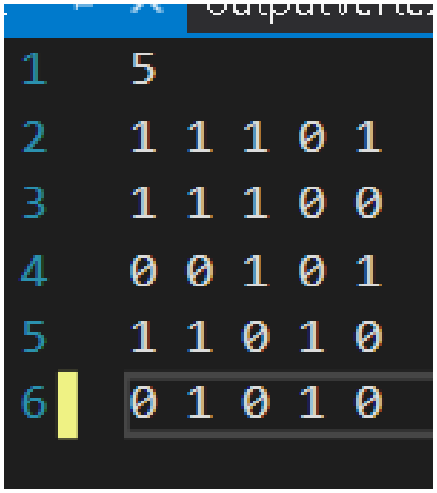
```
                fout << "Power|" << "Vertex\n";
                fout << "|" << setw(4) << max << "| ";
                for (vector<int>::iterator it = NoMaxVertex.begin(); it !=
NoMaxVertex.end(); ++it)
                        fout << *it << " ";
                fout << "|" << "\n====";
                for (int i = 0; i < (n * 2); i++) {
                        fout << "=";
                }
                fout << endl;
                fout.close();
        }
}
```

Main.cpp

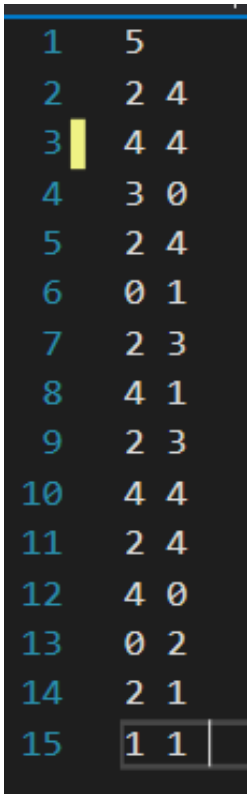
```
#include "pch.h"
#include "MatrixGraph.h"
#include "MyException.h"
#include "MyExceptionForVertex.h"

int main()
{
        try{
                setlocale(LC_ALL, "Rus");

                vector <MatrixGraph> arr;

                cout << "Введите номер способа считывания графа из файла\n";
                cout << "1: В файле указан размер матрицы и готовая матрица
                        смежности\n";
                cout << "2: В файле указан размер матрицы и пары смежных
                        вершин\n";
                int method;
                cin >> method;
                string fileIn;
                cout << "Введите названия файла где укана информация\n";
                cin >> fileIn;
                switch (method){
                case 1:
                        inputGraphAsMatrix(arr, fileIn);
                        break;

                case 2:
                        inputMatrixGraph(arr, fileIn);
                        break;

                default:
                        throw "Ошибка! Введите число 1 или 2";
                        break;
                }

                arr.back().Print();
```

```
			arr.back().deleteCircleVertexes(); // удаление циклов в мультиграфе

			arr.back().deleteMultipleEdge(); // удаления кратных рёбер

			outputMatrixGraph(arr.back());

			adjacentVertexes *adjVert = new adjacentVertexes[arr.back().getVertex()];
			// список смежных вершин
			listAdjacentVertexes(arr.back(), adjVert); // список смежности
			delete[] adjVert;

			inOutPowerVertex(arr.back()); // входящая и исходящая степень
							вершин

			vertexMaxPower(arr.back()); // максимальная степень вершин

		}
		catch (MyException ex) {
			cerr << ex.getErrorFile() << endl;
			system("Pause");
			return -1;
		}
		catch (MyExceptionForVertex ex) {
			cerr << ex.getError() << endl;
			system("Pause");
			return -1;
		}
		catch (const char* error) {
			cout << endl << error << endl;
			return -1;
		}
		return 0;
}
```

Приложение 2



1. Входные данные в файле для первого представления графа



2. Входные данные в файле для второго представления графа

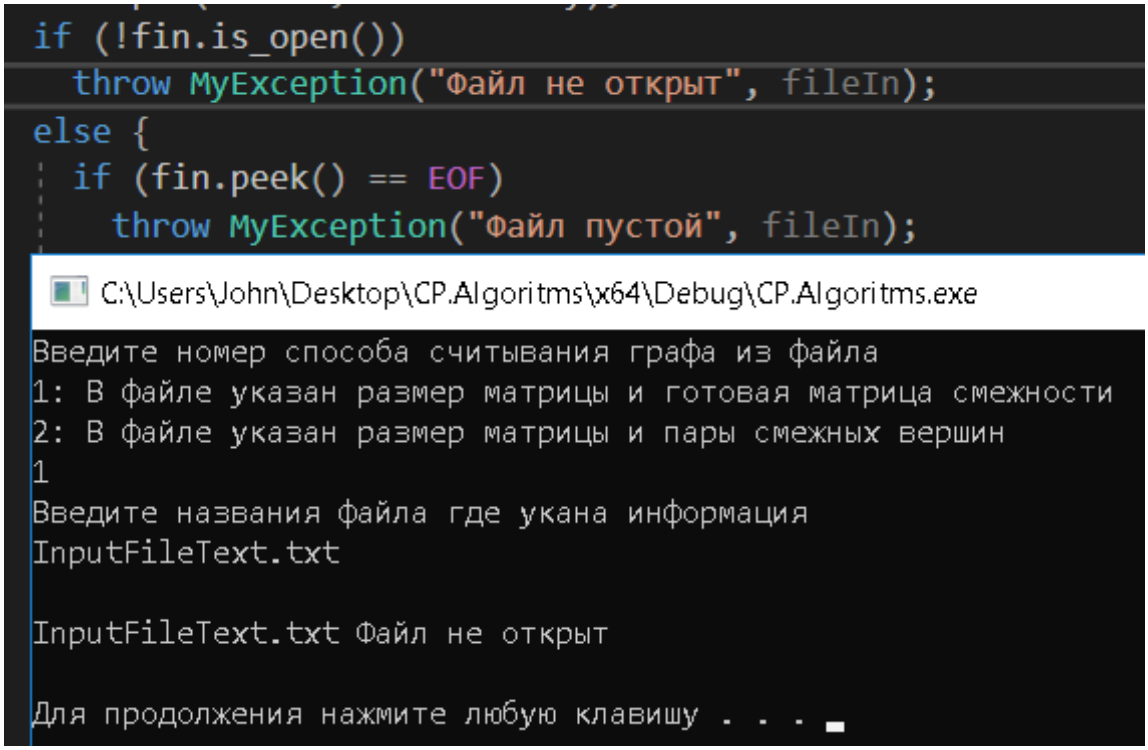
```
=Adjacency Matrix=
No| 0 1 2 3 4
  =============
0 | 0 1 1 0 1 |
1 | 0 0 1 0 0 |
2 | 0 0 0 0 1 |
3 | 1 1 0 0 0 |
4 | 0 1 0 1 0 |
  =============
=AdjacencyList=
 from   to
|   0 | 1 2 4
|   1 | 2
|   2 | 4
|   3 | 0 1
|   4 | 1 3
=============
=PowerVertex=
No |0 1 2 3 4
In |1 3 2 1 2
Out|3 1 1 2 2
==============
=MaxVertexPower=
Power|Vertex
|   4| 0 1 4 |
==============
```

3. Вывод всей информации в выходной файл

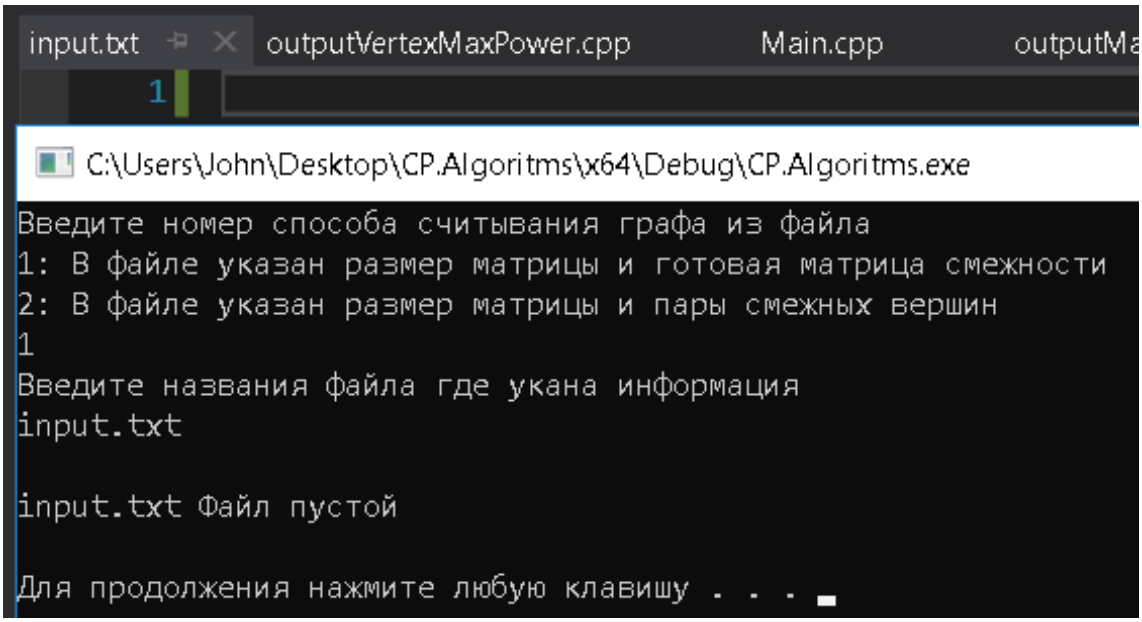
```
cout << "Введите номер способа считывания графа из файла\n";
cout << "1: В файле указан размер матрицы и готовая матрица смежности\n";
cout << "2: В файле указан размер матрицы и пары смежных вершин\n";
int method;
cin >> method;
string fileIn;
cout << "Введите названия файла где укана информация\n";
cin >> fileIn;
```

```
Консоль отладки Microsoft Visual Studio
Введите номер способа считывания графа из файла
1: В файле указан размер матрицы и готовая матрица смежности
2: В файле указан размер матрицы и пары смежных вершин
4
Введите названия файла где укана информация
input.txt

Ошибка! Введите число 1 или 2

C:\Users\John\Desktop\CP.Algoritms\x64\Debug\CP.Algoritms.exe (процесс 4316) завершает работу с кодом -1.
Чтобы закрыть это окно, нажмите любую клавишу…
```

4. Пользователь ввёл недопустимый способ работы с файлом



5. Пользователь ввёл названия файла, который не был найден



6. Пользователь ввёл файл, который оказался пустым