Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра “Вычислительная техника”

Пояснительная записка
К курсовому проектированию
По курсу “Логика и основы алгоритмизации
в инженерных задачах”
на тему “Реализация алгоритма нахождения изолированных вершин в графе”

Выполнил:
студент группы
24ВВВ4
Евсеев К.С

Принял:
Акифьев И.В.

Акифьев 09.06.26г.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ _____

«____»_____20__

ЗАДАНИЕ

на курсовое проектирование по курсу

Логика и основы алгоритмизации в инженерных задачах
Студенту Евсееву К.С Группа 2ЧВВВ4
Тема_проекта Реализация алгоритма нахождения кратчайших путей в графе

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данным заданием курсового проекта
Пояснительная записка должна содержать
1. Постановку задачи
2. Теоретическую часть задания
3. Описание алгоритма поставленной задачи
4. Пример ручного решения задачи и вычислений
5. Описание самой программы
6. Тесты
7. Список литературы
8. Листинг программы
9. Результаты работы программы

Объем работы по курсу

1. Расчетная часть

Ручной расчет работы алгоритма

2. Графическая часть

Схема алгоритма в формате блок-схемы

3. Экспериментальная часть

Тестирование программы
Результаты работы программы на тестовых данных

Срок выполнения проекта по разделам

1 Исследование теоретической части курсового проекта
2 Разработка алгоритмов программы
3 Разработка программы
4 Тестирование и отладка работы программы
5 Оформление пояснительной записки
6
7
8

Дата выдачи задания "10" сентября

Дата защиты проекта "___" _____

Руководитель Антропов И.В.

Задание получил "10" сентября 2025 г.

Студент Евлев Константин Сергеевич

Содержание

Оглавление

Реферат 6

Введение 7

Постановка задачи 6

Теоретическая часть задания 7

Описание алгоритма программы 12

Описание программы 13

Описание алгоритма поиска изолированных вершин. 15

Тестирование 16

Ручной расчёт задачи 18

Заключение 20

Список литературы 21

Приложение А. 22

Листинг программы. 22

Реферат

Отчет 22 стр, 11 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, ОРГРАФ.

Цель исследования – Реализация алгоритма нахождения изолированных вершин в графе.

В работе рассмотрена реализация алгоритма нахождения изолированных вершин в графе. Установлено, что с помощью данного алгоритма можно найти изолированные вершины в графе.

Введение

Алгоритм нахождения изолированных вершин в графе основан на просмотре строк в матрице смежности графа.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2021, язык программирования – C++.

Целью данной курсовой работы является разработка программы на языке C++, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм нахождения изолированных вершин в графе.

Постановка задачи

Требуется разработать программу, которая находит изолированные вершины в графе.

Исходный граф в программе должен задаваться матрицей смежности, причём при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки этих данных на экран должна выводиться матрица смежности орграфа, вид орграфа и все компоненты связности орграфа. Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

Устройства ввода – клавиатура, мышь.

Теоретическая часть задания

Граф - это топологичекая модель, которая состоит из множества вершин и множества соединяющих их рёбер. При этом значение имеет только сам факт, какая вершина с какой соединена.

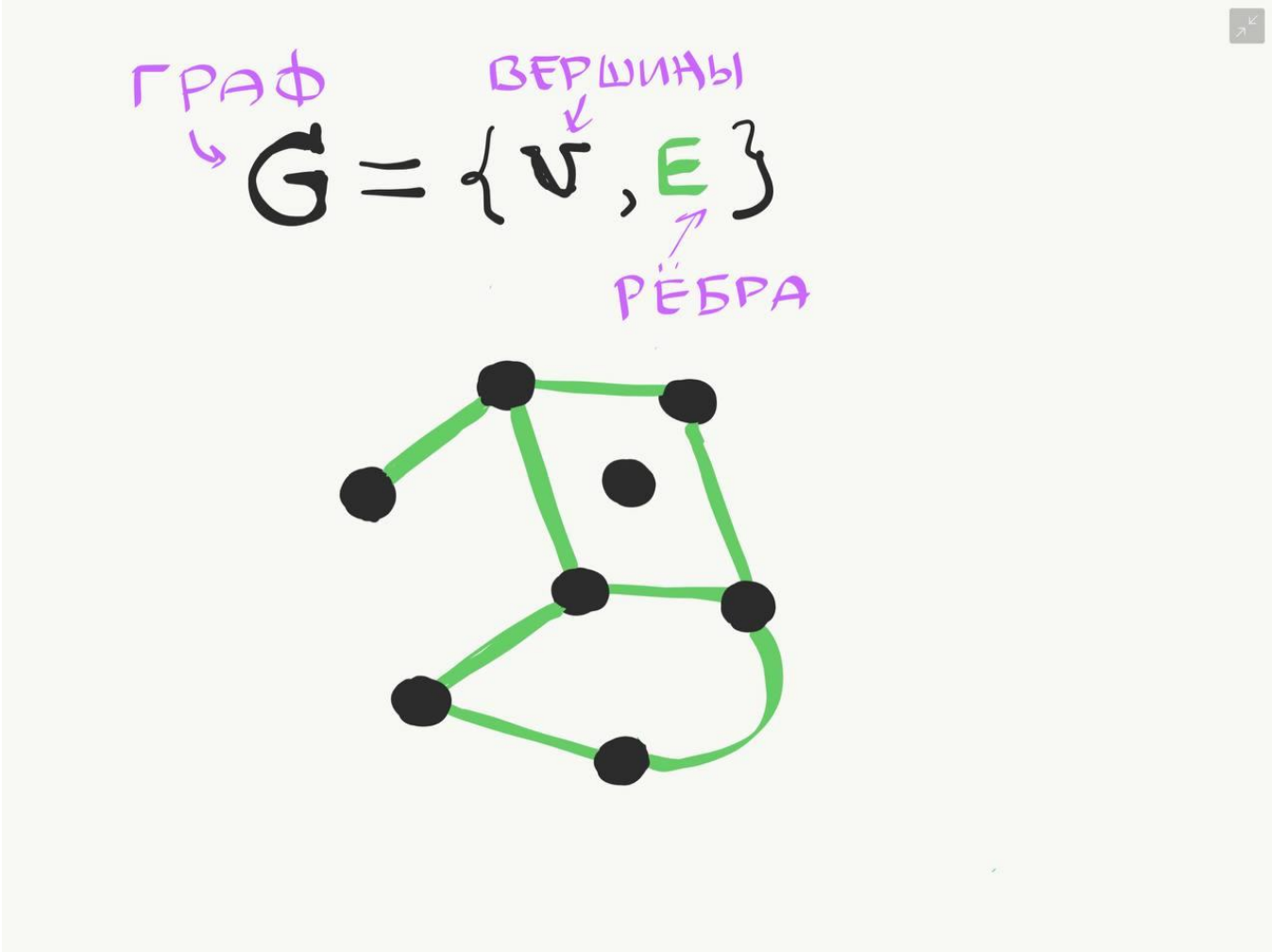

Рисунок 1

Очень многие задачи могут быть решены используя богатую библиотеку алгоритмов теории графов. Для этого достаточно лишь принять объекты за вершины, а связь между ними - за рёбра, после чего весь арсенал алгоритмов теории графов к вашим услугам: нахождение маршрута от одного объекта к другому, поиск связанных компонент, вычисление кратчайших путей, поиск сети максимального потока и многое другое.

Вершина - точка в графе, отдельный объект, для топологической модели графа не имеет значения координата вершины, её расположение, цвет, вкус, размер; однако при решении некоторых задачах вершины могут раскрашиваться в разные

цвета или сохранять числовые значения.

Ребро - неупорядоченная пара двух вершин, которые связаны друг с другом. Эти вершины называются концевыми точками или концами ребра. При этом важен сам факт наличия связи, каким именно образом осуществляется эта связь и по какой дороге - не имеет значения; однако рёбра может быть присвоен “вес”, что позволит говорить о “нагруженном графе” и решать задачи оптимизации.

Инцидентность - вершина и ребро называются инцидентными, если вершина является для этого ребра концевой. Обратите внимание, что термин “инцидентность” применим только к вершине и ребру.

Смежность вершин - две вершины называются смежными, если они инцидентны одному ребру.

Смежность рёбер - два ребра называются смежными, если они инцедентны одной вершине.

Говоря проще - две вершины смежные, если они соединены ребром, два ребра смежные - если они соединены вершиной.

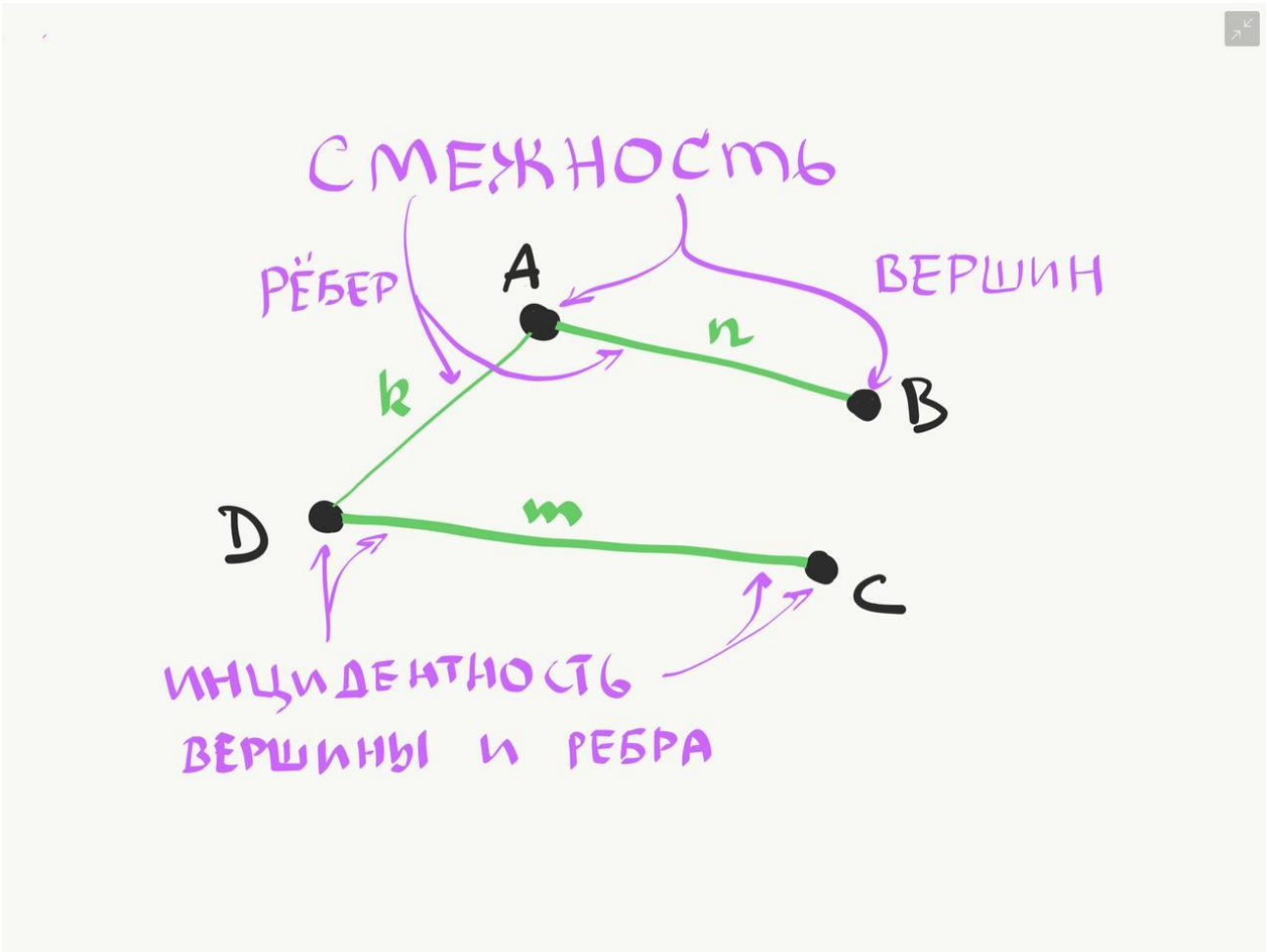

Рисунок 2

Петля - ребро, инцидентное одной вершине. Ребро, которое замыкается на одной вершине.

Псевдограф - граф с петлями. С такими графами не очень удобно работать, потому что переходя по петле мы остаёмся в той же самой вершине, поэтому у него есть своё название.

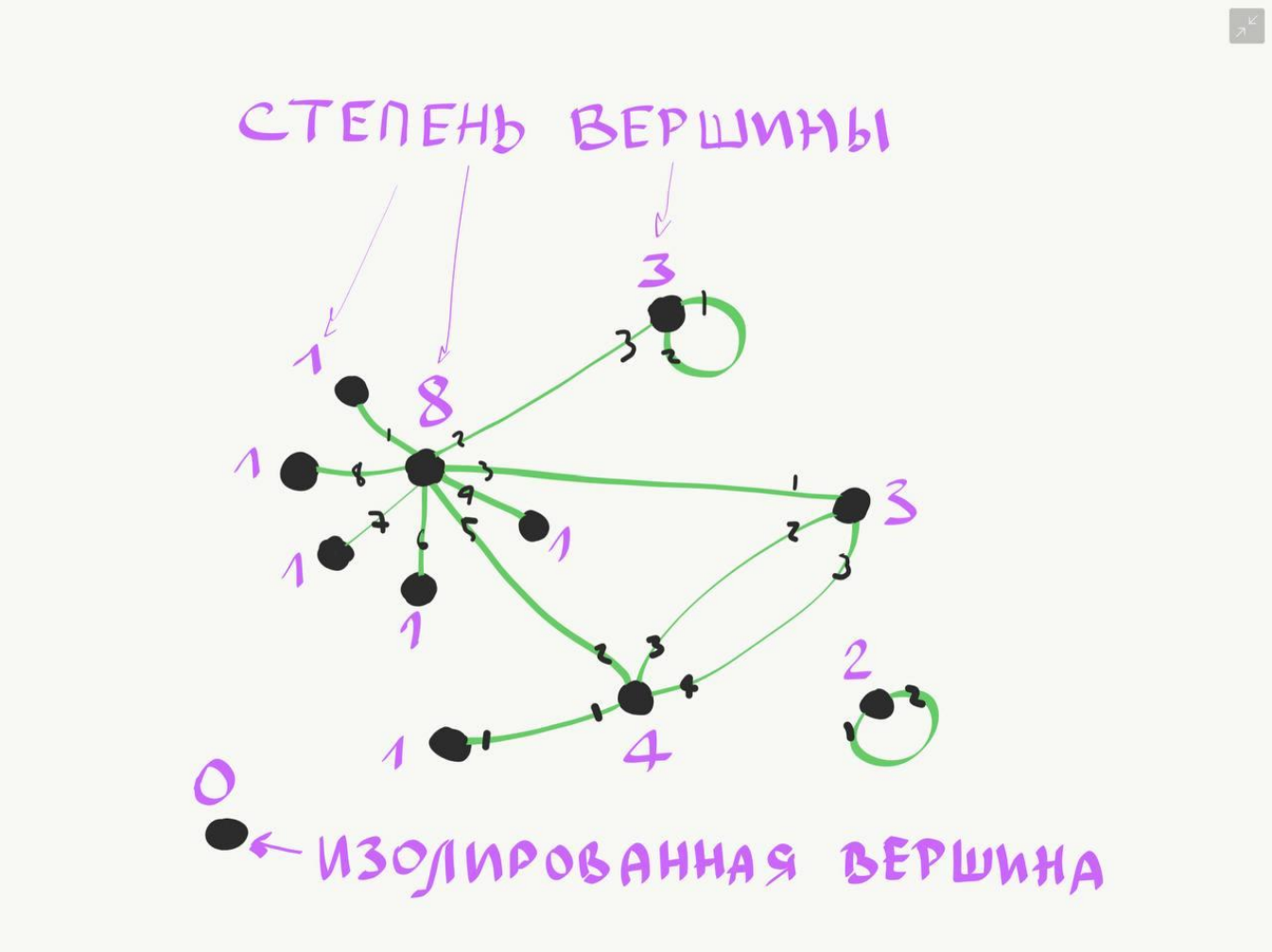

Рисунок 3

Подграф. Если в исходном графе выделить несколько вершин и несколько рёбер (между выбранными вершинами), то мы получим подграф исходного графа.

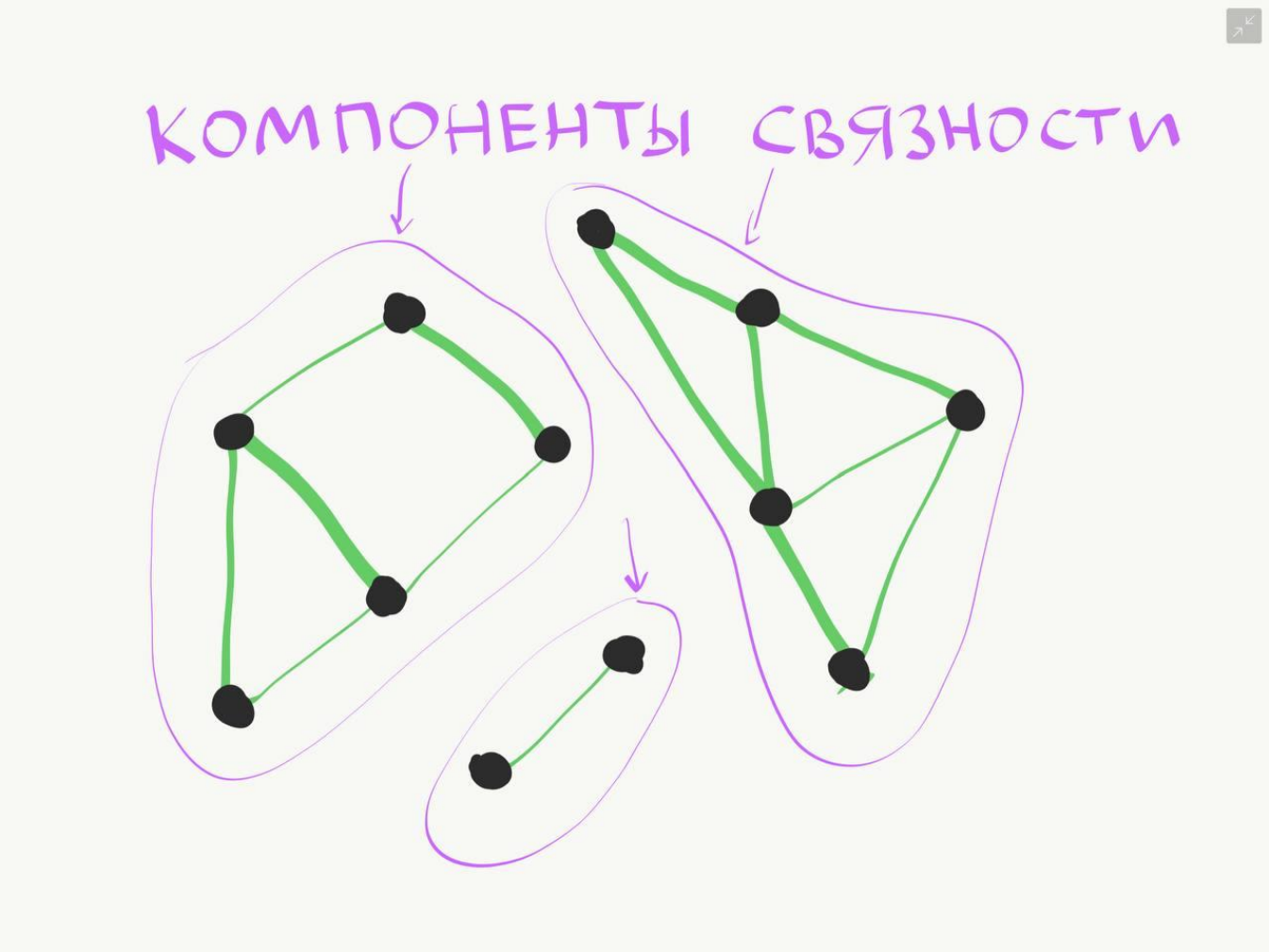

Рисунок 4

Компонента сильной связности - максимальное множество вершин орграфа, между любыми двумя вершинами которого существует путь по дугам.

Компонента слабой связности - максимальное множество вершин орграфа, между любыми двумя вершинами которого существует путь по дугам без учёта направления (по дугам можно двигаться в любом направлении).

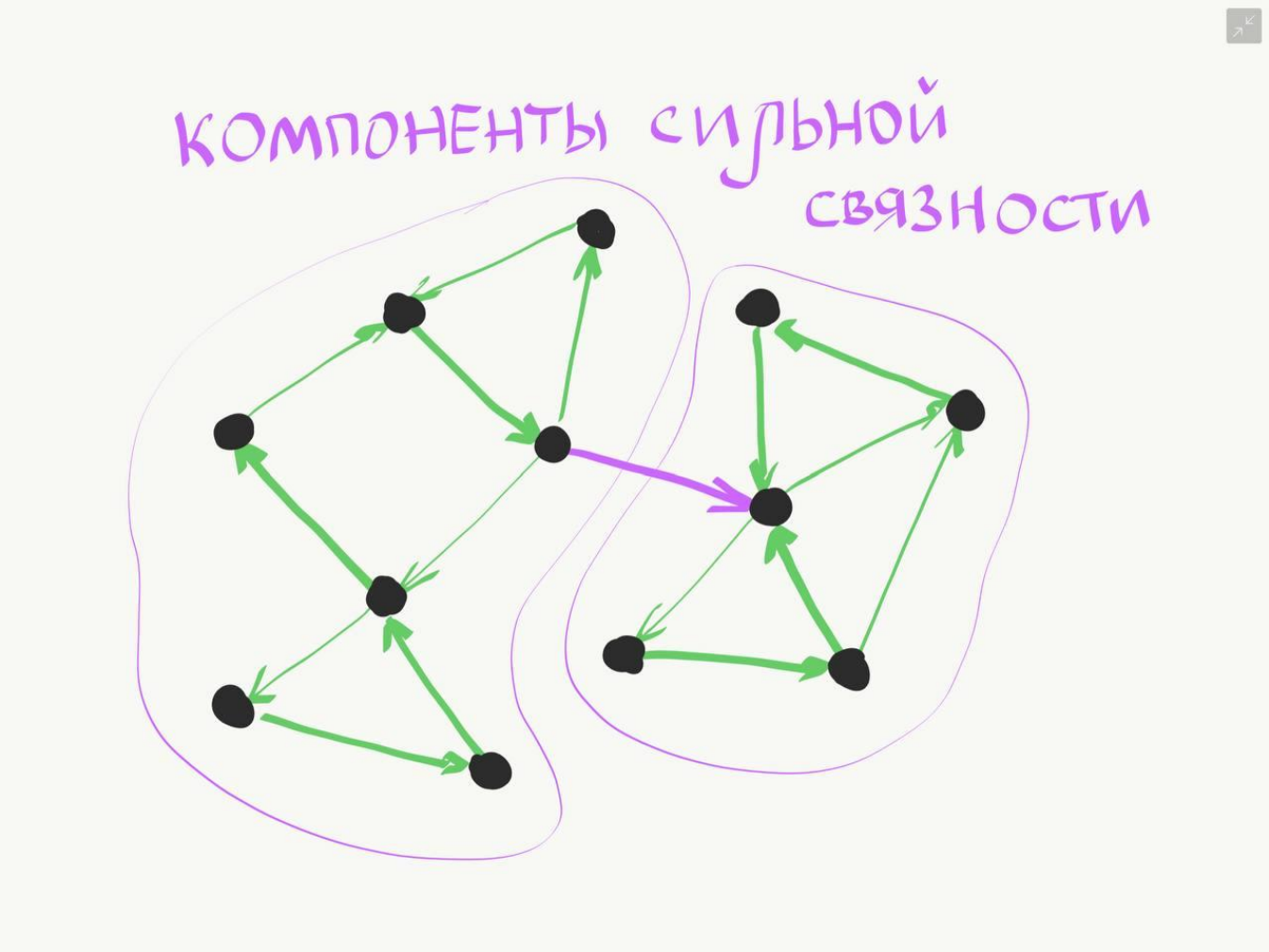

Рисунок 5

Мост - ребро, при удалении которого, количество связанных компонент графа увеличивается.

Описание алгоритма программы

Для программной реализации алгоритма понадобится: двойной указатель int** a – для создания динамической матрицы, которая хранит ориентированный граф, параметр int n, с помощью которого мы задаём размеры графа, int sum для подсчёта количества изолированных вершин.

Создаётся динамическая матрица и случайно заполняется 1 и 0. С помощью функции addisolated(int** a, int n) мы добавляем случайное количество изолированных вершин в наш граф, после выводим его не экран. Далее мы просматриваем каждую вершину в графе, если она изолирована, то выводим ее на экран и увеличиваем значение sum на единицу. После этого, мы выводим количество изолированных вершин.

Ниже представлен псевдокод части программы, которая отвечает за проверку изолированных вершин.

1. для i = 0, пока i < n, i++;
    1. для j = 0, пока j < n, j ++;
        1. если a[i][j] == 1 или a[j][i] == 1, то выходим из цикла;
        2. если j == n – 1, то
            1. sum++;
            2. выводим вершину i на экран

Полный код программы можно увидеть в Приложении А.

Описание программы

Для написания данной программы использован язык программирования C++. Язык программирования C++ - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Проект был создан в виде консольного приложения Win32 (Visual C++).

Работа программы начинается с запроса размера графа. В графе допускается наличие петель. Петля отображается единицей на главной диагонали матрицы смежности и означает наличие ребра, соединяющего вершину саму с собой. После того, как пользователь ввел размеры графа на экран выводится матрица смежности графа. Далее программа добавляет изолированные вершины в граф и выводит его на экран.

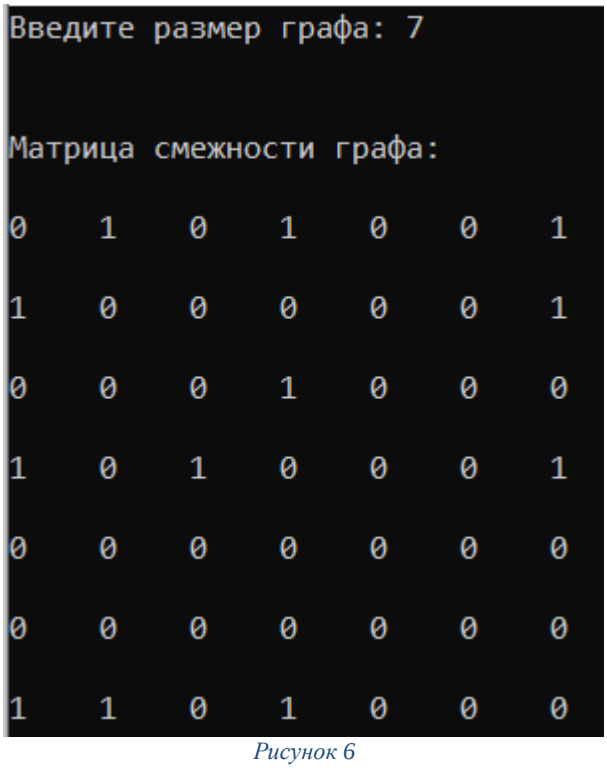

Рисунок 6

Далее программа ищет изолированные вершины и выводит их на экран, так же выводится на экран информация о их количестве.

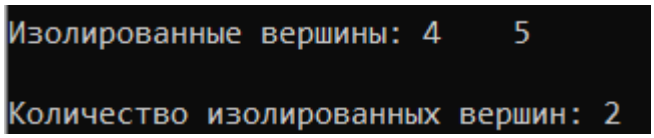

Рисунок 7

Результаты работы программы записываются в файл.

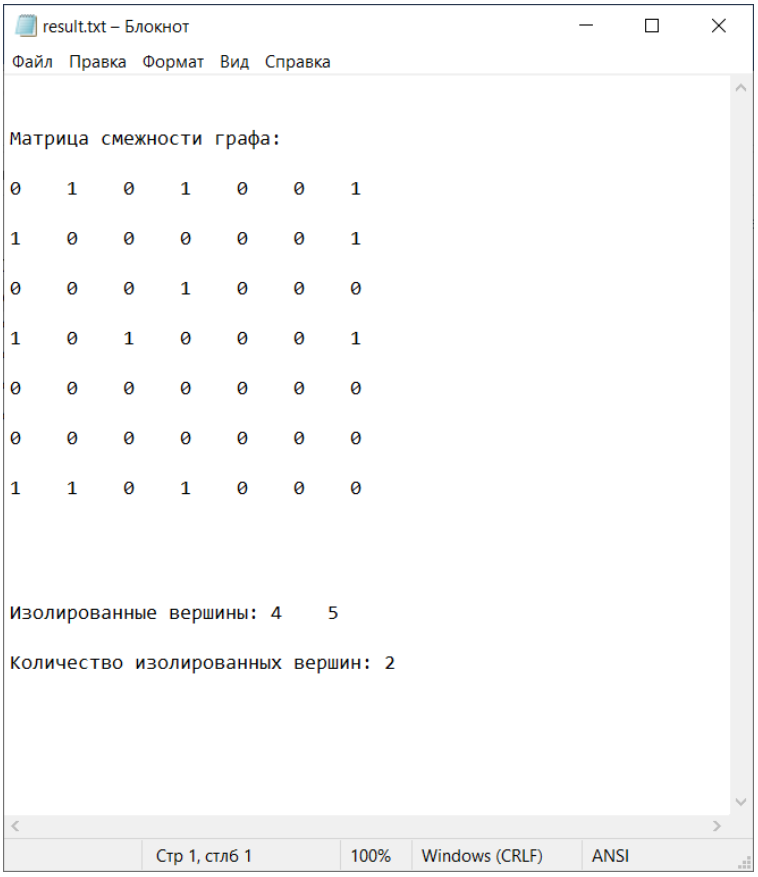

Рисунок 8

Описание алгоритма поиска изолированных вершин.

Ниже представлена часть программы, которая отвечает за поиск изолированных вершин.

```
for (int i = 0; i < n; i++)
{
	for (int j = 0; j < n; j++)
	{
		if (a[i][j] == 1 || a[j][i] == 1)
			break;
		if (j == n - 1)
		{
			sum++;
			cout << i << setw(5);
		}
	}
}
```

Цикл «for (int i = 0; i < n; i++)» и «for (int j = 0; j < n; j++)» отвечают за проход по матрице смежности.

Условие «if (a[i][j] == 1 || a[j][i] == 1) break;» проверяет есть ли у вершины i и j общее ребро, если ребро найдено, то мы переходим к следующей итерации цикла.

Условие «if (j == n - 1)» выполняется в том случае, когда мы уже проверили все вершины с вершиной i на наличие ребра и не нашли его, вершина i является изолированной. При выполнении этого условия мы прибавляем к переменной sum единицу, в переменной sum хранится информация о количестве изолированных вершин, и выводим вершину i на экран.

Тестирование

Среда разработки MicrosoftVisualStudio 2021 предоставляет все средства, необходимые при разработке и отладке многомодульной программы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

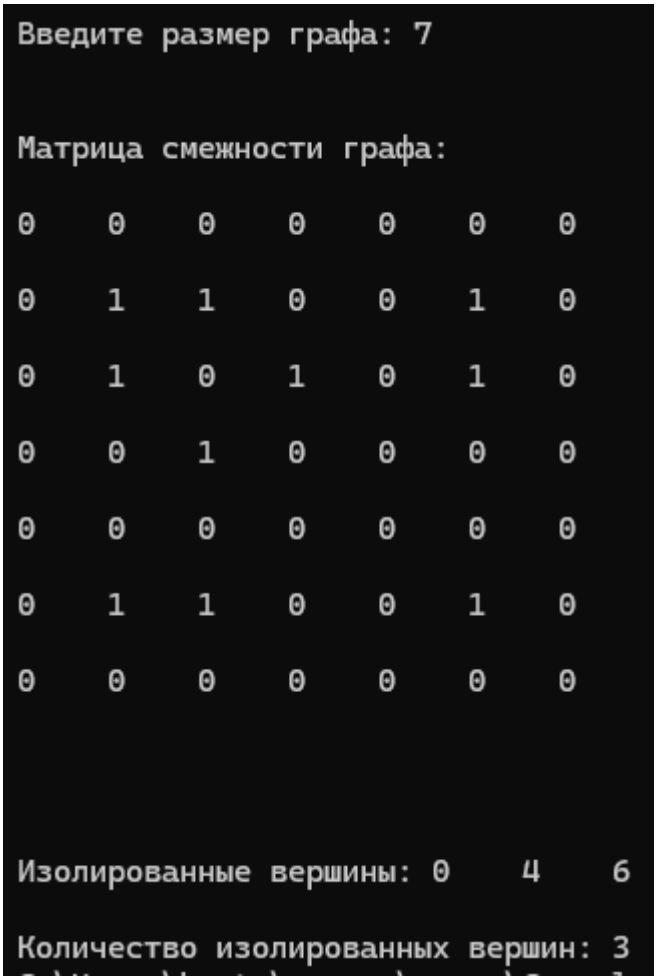

Рисунок 9

На рисунке 6 продемонстрирован результат тестирования при размерах графа 7x7.

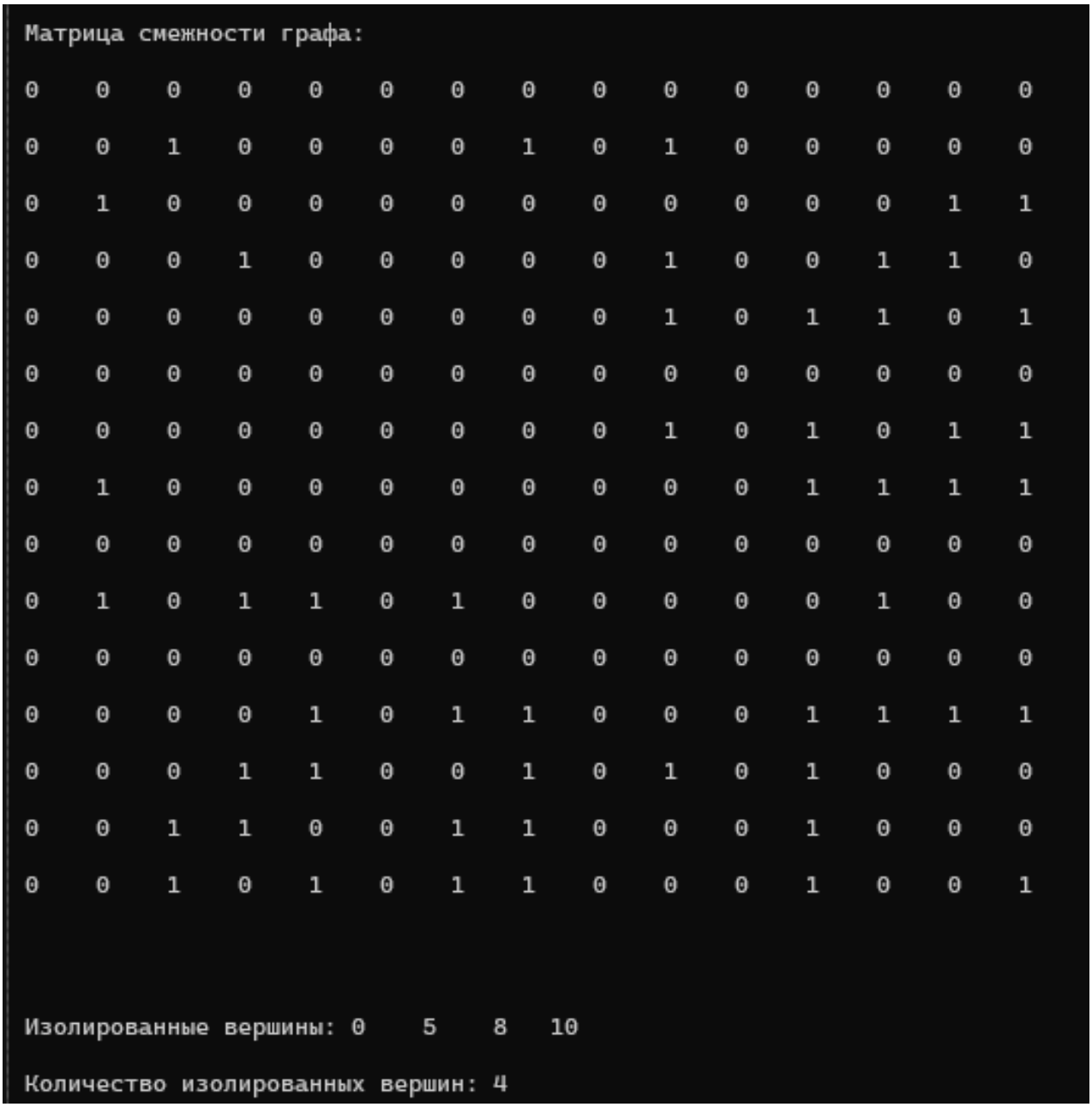

Рисунок 10

На рисунке 7 продемонстрирован результат тестирования при размерах графа 15x15.

В результате тестирования было выявлено, что программа успешно создаёт граф и находит изолированные вершины.

Ручной расчёт задачи

Проведем проверку программы посредством ручных вычислений на примере графа размером 7×7 (рисунок 9).

Проверка осуществляется последовательным просмотром строк и столбцов матрицы смежности. Если для вершины обнаруживается хотя бы одно ребро (значение 1 в соответствующей строке или столбце), то вершина считается неизолированной и выполняется переход к проверке следующей вершины.

Матрица смежности исследуемого графа:

0 0 0 0 0 0 0

0 1 1 0 0 1 0

0 1 0 1 0 1 0

0 0 1 0 0 0 0

0 0 0 0 0 0 0

0 1 1 0 0 1 0

0 0 0 0 0 0 0

Проверка вершины 0:

В строке и столбце отсутствуют единицы, следовательно вершина является изолированной.

Проверка вершины 1:

В строке имеются единицы. Кроме того, на главной диагонали находится значение a[1][1] = 1, что соответствует петле. Вершина не является изолированной.

Проверка вершины 2:

В строке имеются единицы, вершина связана с другими вершинами графа. Вершина не является изолированной.

Проверка вершины 3:

В строке имеется единица, соответствующая связи с вершиной 2. Вершина не является изолированной.

Проверка вершины 4:

В строке и столбце отсутствуют единицы, следовательно вершина является изолированной.

Проверка вершины 5:

В строке имеются единицы. На главной диагонали находится значение a[5][5] = 1, что соответствует петле. Вершина не является изолированной.

Проверка вершины 6:

В строке и столбце отсутствуют единицы, следовательно вершина является изолированной.

Следовательно, изолированными являются вершины 0, 4 и 6.

Для наглядности изобразили данный граф (рисунок 11).

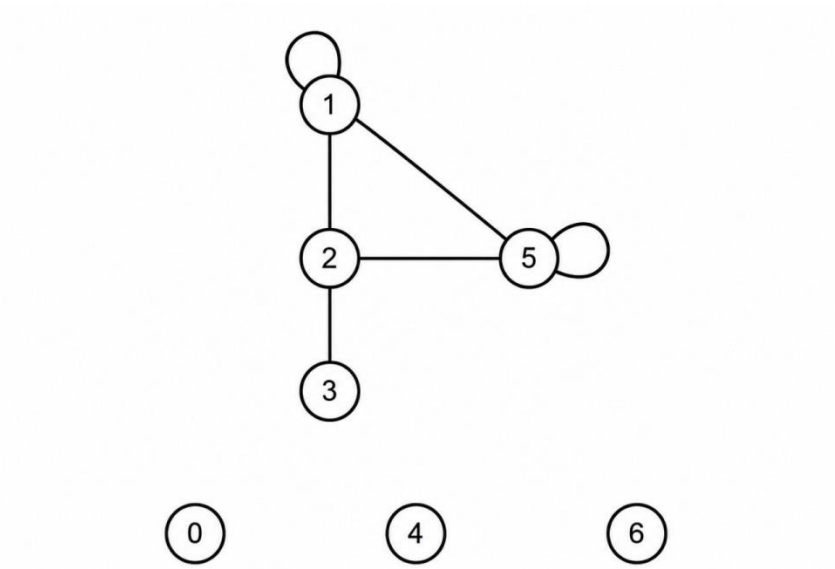

Рисунок 11

Сравнив результат работы программы и ручной расчёт, можно сделать вывод, что программа работает корректно.

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм поиска изолированных вершин в графе в MicrosoftVisualStudio 2021.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежности, основанных на теории орграфов. Приобретены навыки по осуществлению алгоритма поиска изолированных вершин в графа. Углублены знания языка программирования C++.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Потому что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

Программа имеет небольшой, но достаточный для использования функционал возможностей.

Список литературы

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ - М.: МЦНМО, 2001. - 960 с.
2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978
3. Герберт Шилдт «Полный справочник по C++» - Вильямс, 2006
4. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977. 208 с.
5. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. 2009 г.
6. 3. Оре О. Графы и их применение: Пер. с англ. 1965. 176 с.

Приложение А.
Листинг программы.

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <stack>


using namespace std;

void addisolated(int** a, int n)
{
	int mod, del = n;
	while (del)
	{
		mod = rand() % n;
		for (int i = 0; i < n; i++)
			a[i][mod] = a[mod][i] = 0;
		del /= 2;
	}
}

int main()
{
	int** a, n, sum = 0;

	setlocale(LC_ALL, "RUS");
	srand(time(NULL));

	char check = '\0';
	cout << "Загрузить граф из файла? y/n\n";
	cin >> check;

	if (check == 'y')
	{
		stack <int> graph;
		int buffer = 0;
		ifstream fout("result.txt");
		if (!fout.is_open())
		{
			_cputws(L"Не удалось открыть файл.");
			return -1;
		}

		while (!fout.eof())
		{
			fout >> buffer;
			graph.push(buffer);
		}

		fout.close();
		n = (int)sqrt(graph.size());

		a = (int**)malloc(n * sizeof(int*));
		for (int i = 0; i < n; i++)
			a[i] = (int*)malloc(n * sizeof(int));

		for (int i = n - 1; i > -1; i--)
		{
			for (int j = n - 1; j > -1; j--)
			{
				a[i][j] = graph.top();
				graph.pop();
			}
		}

	}
	else
```

```
	{
		cout << "Введите размер графа: ";
		cin >> n;

		a = (int**)malloc(n * sizeof(int*));
		for (int i = 0; i < n; i++)
			a[i] = (int*)malloc(n * sizeof(int));

		for (int i = 0; i < n; i++)
		{
			for (int j = i; j < n; j++)
			{
				int value = rand() % 2;
				a[i][j] = value;
				a[j][i] = value;
			}
		}
		addisolated(a, n);

	}

	ofstream fin("result.txt");

	cout << "\n\nМатрица смежности графа:";
	for (int i = 0; i < n; i++)
	{
		cout << "\n\n";
		if (i != 0)
			fin << "\n\n";

		for (int j = 0; j < n; j++)
		{
			cout << a[i][j] << setw(5);
			fin << a[i][j] << setw(5);
		}
	}

	cout << "\n\n\n\n\nИзолированные вершины: ";

	sum = 0;

	for (int i = 0; i < n; i++)
	{
		bool isolated = true;

		for (int j = 0; j < n; j++)
		{
			if (i != j && a[i][j] != 0)
			{
				isolated = false;
				break;
			}
		}

		if (isolated)
		{
			cout << i << setw(5);
			sum++;
		}
	}

	cout << "\n\nКоличество изолированных вершин: " << sum;
}
```