|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Основные алгоритмы работы с графами.»** | |
|  | |
| Выполнил студент группы ИКБО-04-21 | Исаев В.В. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# Цель работы

Получение практических навыков по выполнению операций над структурой данных граф.

# Постановка задачи

1. Разработать класс «Граф», обеспечивающий хранение и работу со структурой данных «граф», в соответствии с вариантом индивидуального задания. Реализовать метод ввода графа с клавиатуры, наполнение графа осуществлять с помощью метода добавления одного ребра. Реализовать метод вывода графа и методы, выполняющие задачи, определенные вариантом индивидуального задания.
2. Разработать программу, демонстрирующую работу всех методов класса.
3. Произвести тестирование программы на графе, предложенном в таблице 1.2.
4. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №8. Условия заданий. В таблице 1.1 представлены условия для задания 1.

Таблица 1.1. Условия для задания 1

|  |  |  |
| --- | --- | --- |
| 88 | Список смежных вершин | Составить программу нахождения кратчайших путей методом «Флойда».  Используя результат алгоритма вывести путь между вводимыми парами вершин. |

В таблице 1.2 представлен граф, на котором нужно произвести тестирование.

Таблица 1.2 Граф для тестирования

|  |  |
| --- | --- |
| 8 |  |

# Решение

**Граф** — [математическая абстракция](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B0%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F) реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств — множества самих объектов, называемого множеством [вершин](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D0%B0_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)), и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин.

Графы находят широкое применение в современной науке и технике. Они используются и в [естественных науках](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8) и в [социальных науках](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BD%D0%B0%D1%83%D0%BA%D0%B8), но наибольших масштабов применение графов получило в информатике и сетевых технологиях.

В качестве простейшего примера из жизни можно привести схему перелётов определённой авиакомпании, которая моделируется графом, где вершинами графа являются города, а рёбрами — рейсы, соединяющие пары городов. Дерево каталогов в компьютере также является графом: диски, папки и файлы являются вершинами, а рёбра показывают вложенность файлов и папок в папки и диски.

Простой граф G(V,E) есть совокупность двух множеств – непустого множества V и множества E неупорядоченных пар различных элементов множества V. Множество V называется множеством вершин, множество E называется множеством рёбер.

1. Разработать класс «Граф», обеспечивающий хранение и работу со структурой данных «граф», в соответствии с вариантом индивидуального задания. Реализовать метод ввода графа с клавиатуры, наполнение графа осуществлять с помощью метода добавления одного ребра. Реализовать метод вывода графа и методы, выполняющие задачи, определенные вариантом индивидуального задания.

Для графа был разработан класс «Graph». Класс содержит массивы весов, вершин, путей, сумм (для индивидуального задания), следуюущих элементов и методов, необходимых для выполнения задания. Код класса представлен ниже, полный код представлен в пункте 6 данного отчета.

|  |
| --- |
| class Graph  {  int head[Vmax] = {0};  int next\_el[Emax] = {0};  int terminal[Emax] = {0};  int weight[Emax] = {9999};  int summ[Emax][Emax] = { 0,0 };  int next[Emax][Emax] = { 9999,9999 };  int heads[Emax][Emax] = {9999,9999};  public:  void Add(int v, int u, int w);  void print();  void Floyd();  void Help();  int Printnew(int i,int j);  void BeautPrint();  bool IsGraph = false;  bool IsFloyd = false;  }; |

1. Разработать программу, демонстрирующую работу всех методов класса.

Программа была разработана. Функционал будет представлен в пункте 4 данного отчета, полный код в пункте 6 данного отчета.

1. Произвести тестирование программы на графе, предложенном в таблице 1.2.

Тестирование программы будет представлено в пункте 4 данного отчета.

1. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

# Тестирование

Тестирование приложения.

Для тестирования приложения были разработаны тестовые случаи, представленные в таблице 2.

Таблица 2. Тестовые случаи для приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | № задания | Вводимые данные | Ожидаемый результат |
| 1 | 1 | Согласно таблице 1.2 | Граф создан |
| 2 | 2 | Отсутствуют (До ввода графа) | Ошибка об отсутствии графа |
| 3 | 2 | Отсутствуют (После ввода графа) | Вывод списка смежности графа |
| 4 | 3 | Отсутствуют (До ввода графа) | Ошибка об отсутствии графа |

Таблица 2. Тестовые случаи для приложения (Продолжение)

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | 3 | Отсутствуют (После ввода графа) | Выполнение алгоритма Флойда и вывод новых кратчайших весов |
| 6 | 4 | Отсутствуют (До ввода графа) | Ошибка об отсутствии графа |
| 7 | 4 | Отсутствуют (После ввода графа, до вызова алгоритма Флойда) | Уведомление об отсуствии данных алгоритма Флойда |
| 8 | 4 | 4 7 (После ввода графа, после выполнения лгоритма Флойда) | 2 |
| 9 | 4 | 8 9 (После ввода графа, после выполнения лгоритма Флойда) | Ошибка об отсутствии пути |
| 10 | 5 | Отсутствуют | Закрытие приложения |

Тест 1

Ввод вершин графа представлен на рисунке 1.

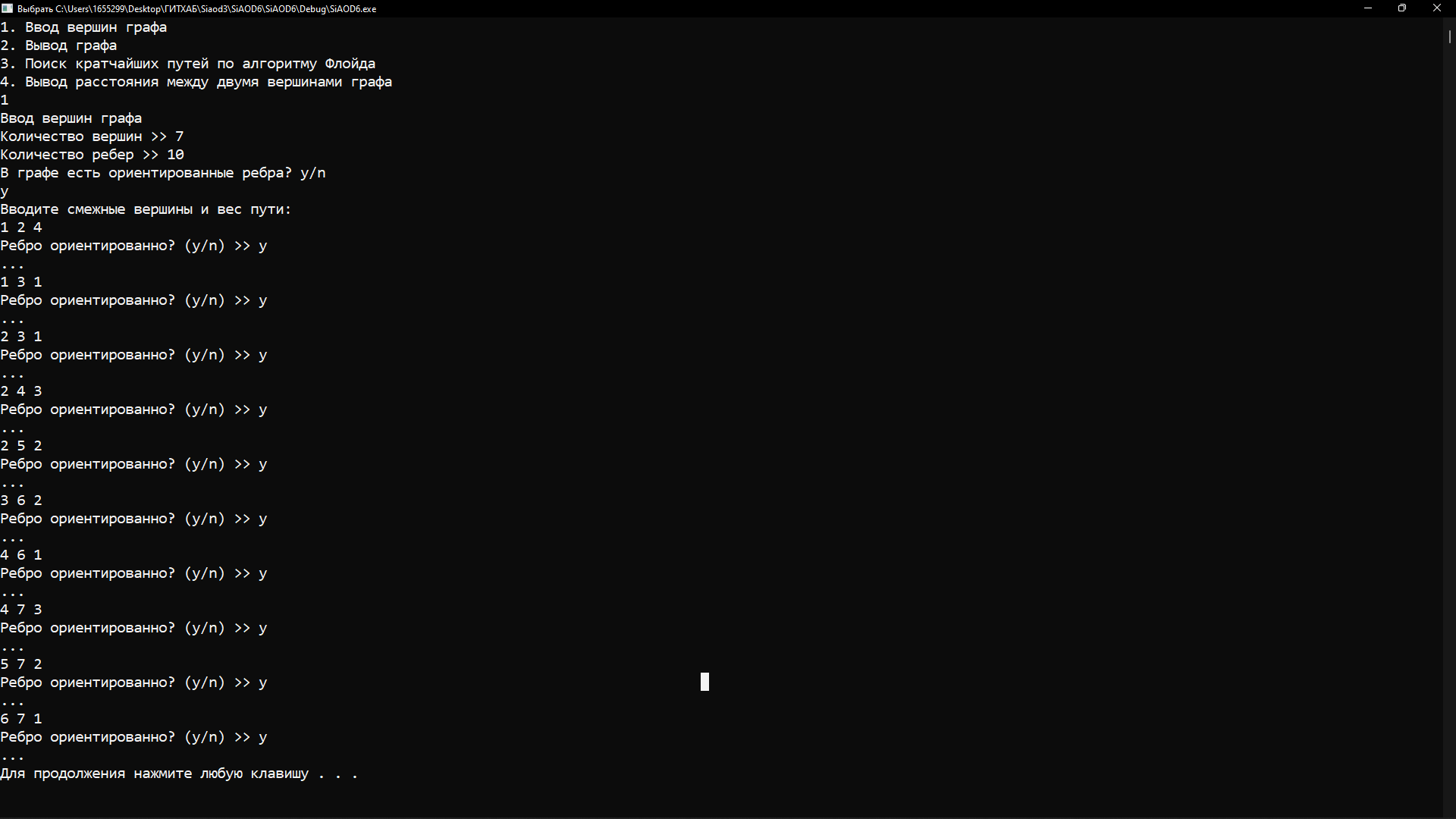


Рис. 1 ввод вершин графа

Тест 2

Подтверждение отсутсвия графа для теста 2 представлено на рисунке 2.

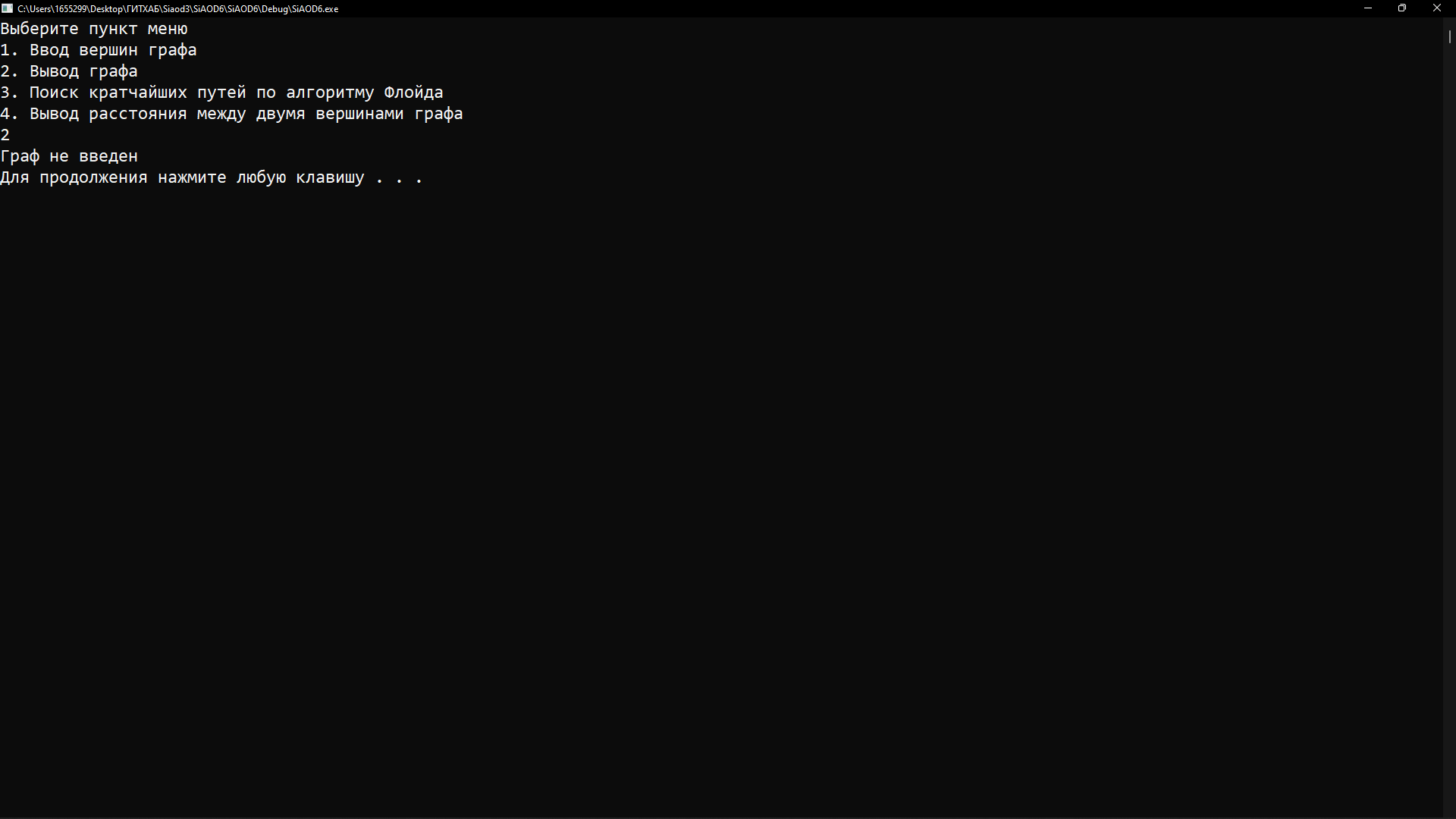


Рис. 2 подтверждение отсутсвия графа для теста 2

Тест 3

Вывод списка смежности графа представлен на рисунке 3.

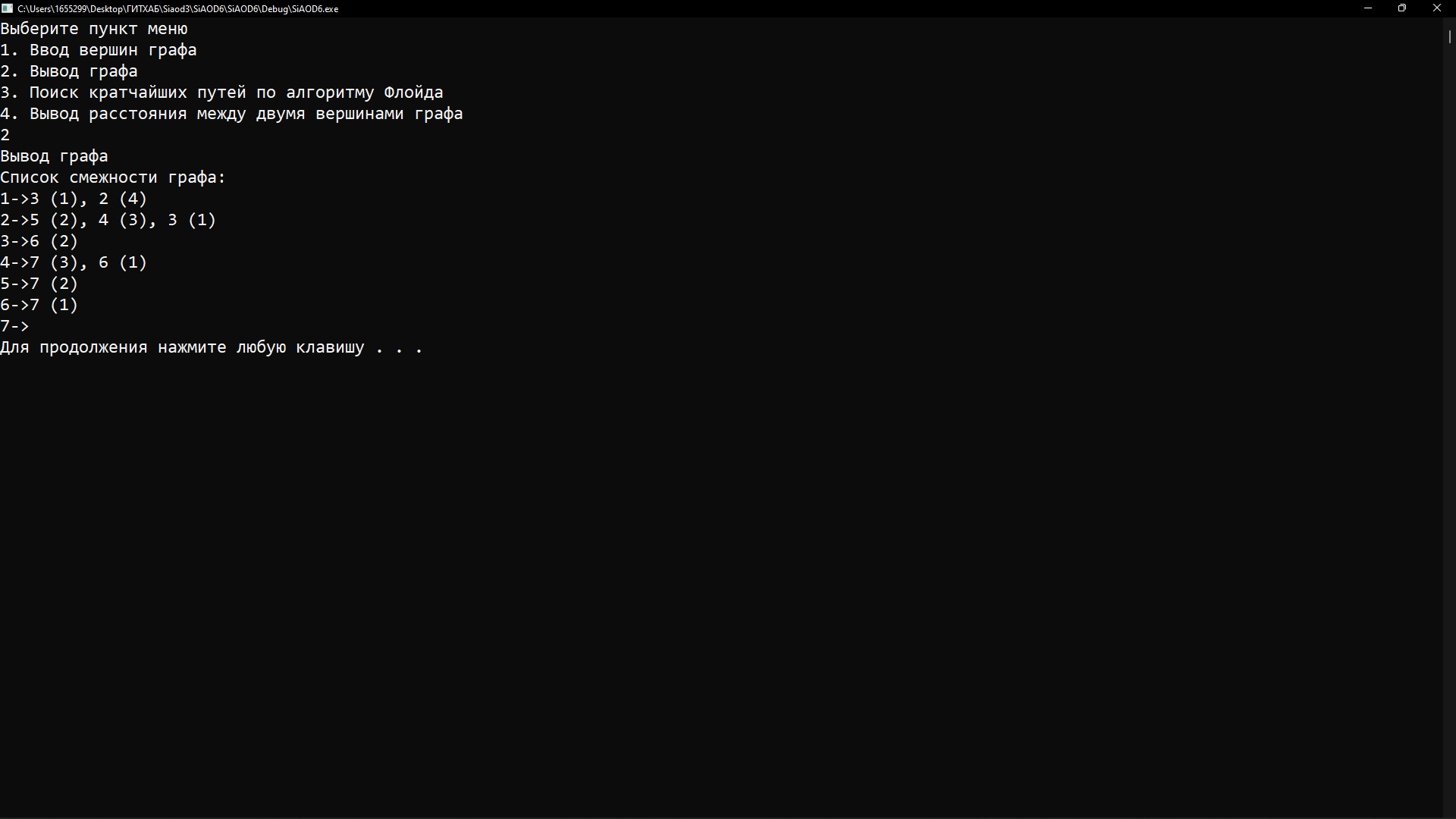


Рис. 3 вывод списка смежности графа

Тест 4

Подтверждение отсутсвия графа для теста 4 представлено на рисунке 4.

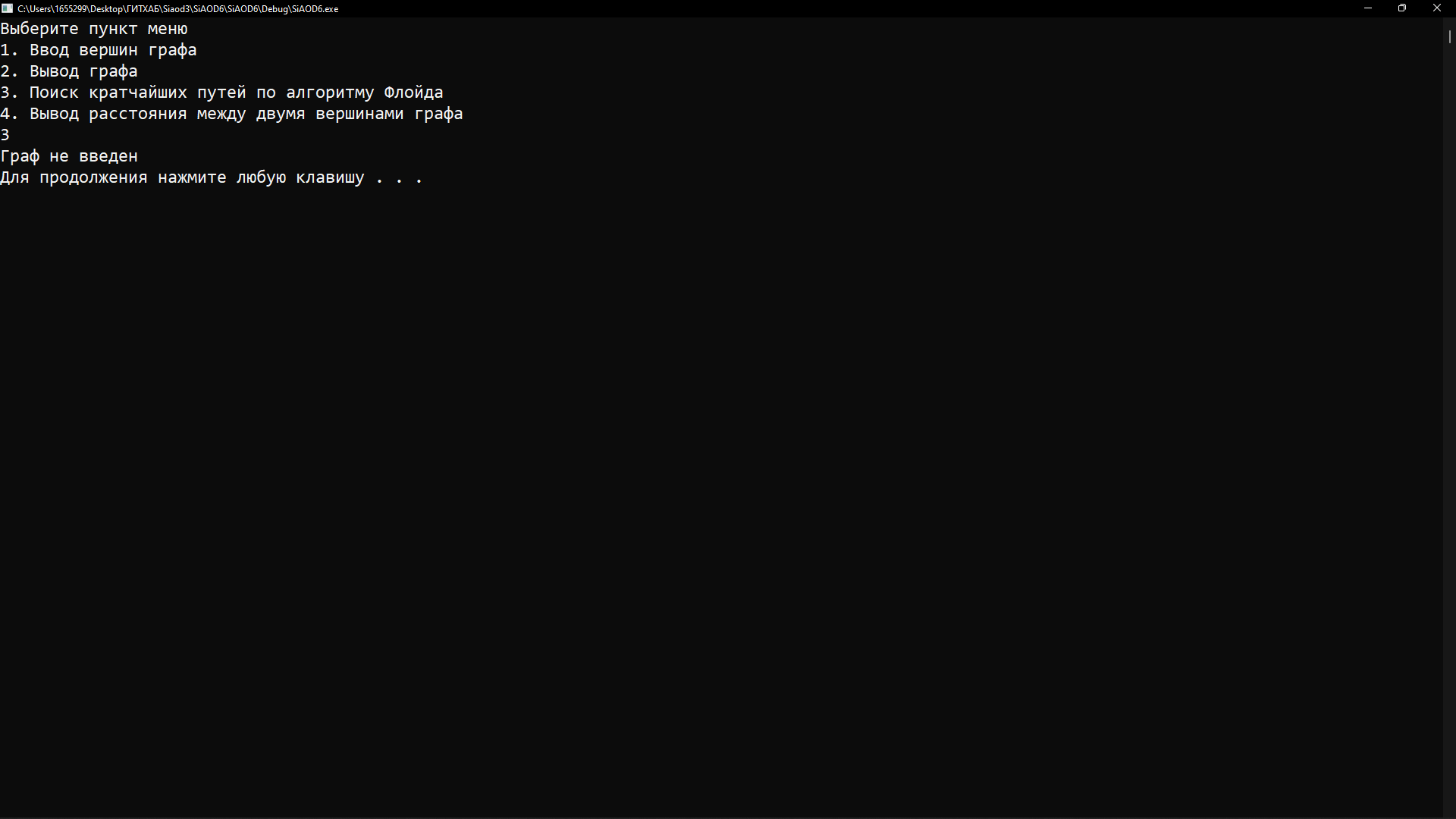


Рис. 4 подтверждение отсутсвия графа для теста 4

Тест 5

Вывод результата работы алгоритма Флойда представлен на рисунке 5.

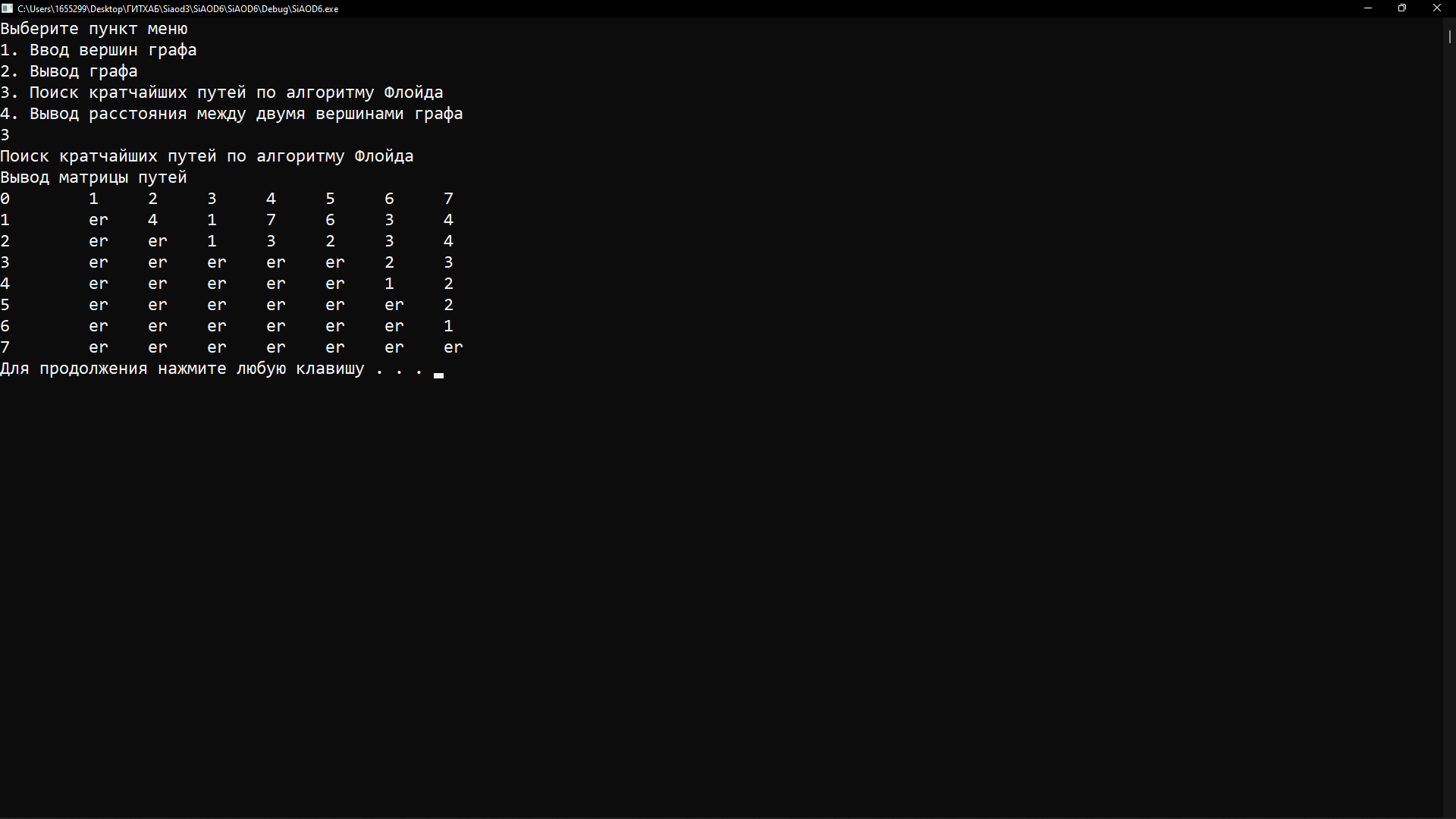


Рис. 5 вывод результата работы алгоритма Флойда

Тест 6

Подтверждение отсутсвия графа для теста 6 представлено на рисунке 6.

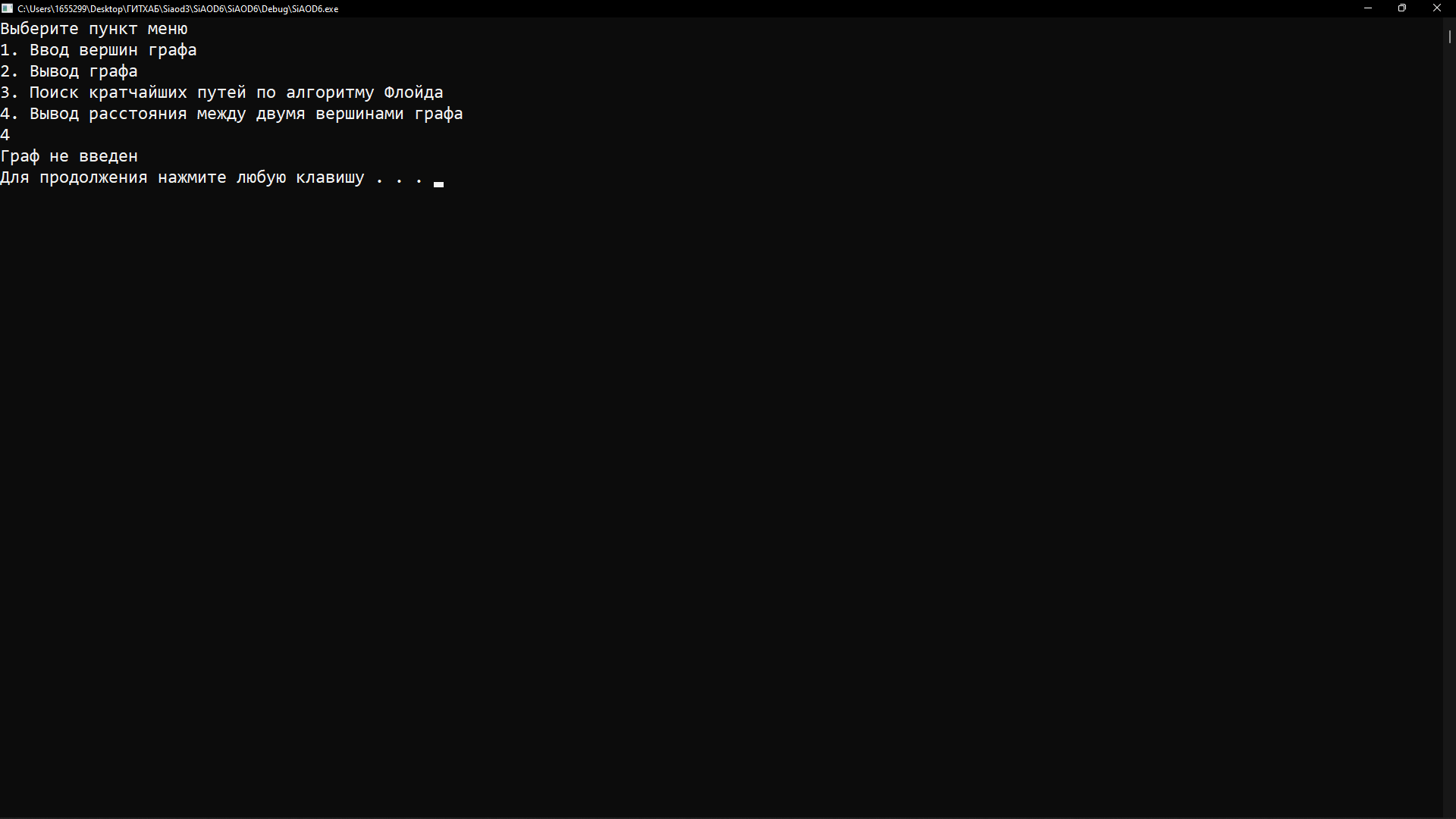


Рис. 6 подтверждение отсутсвия графа для теста 6

Тест 7

Вывод уведомления об отсутствии результатов работы алгоритма Флойда представлен на рисунке 7.



Рис. 7 вывод уведомления об отсутствии результатов работы алгоритма Флойда

Тест 8

Вывод кратчайшего пути по результатам работы алгоритма Флойда и маршрута представлен на рисунке 8.

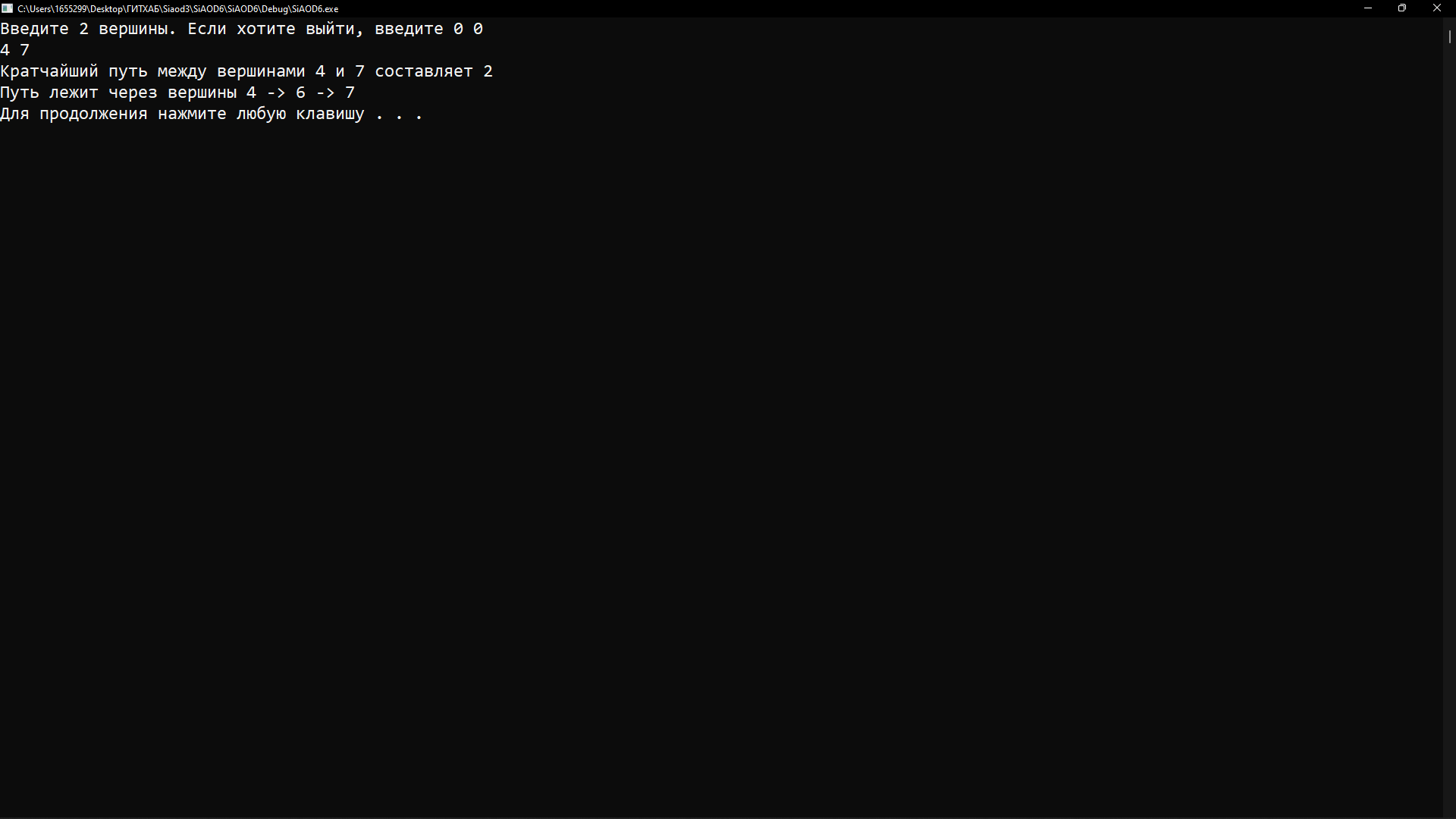


Рис. 8 вывод кратчайшего пути по результатам работы алгоритма Флойда и маршрута

Тест 9

Вывод ошибки об отсутствии пути представлен на рисунке 9.

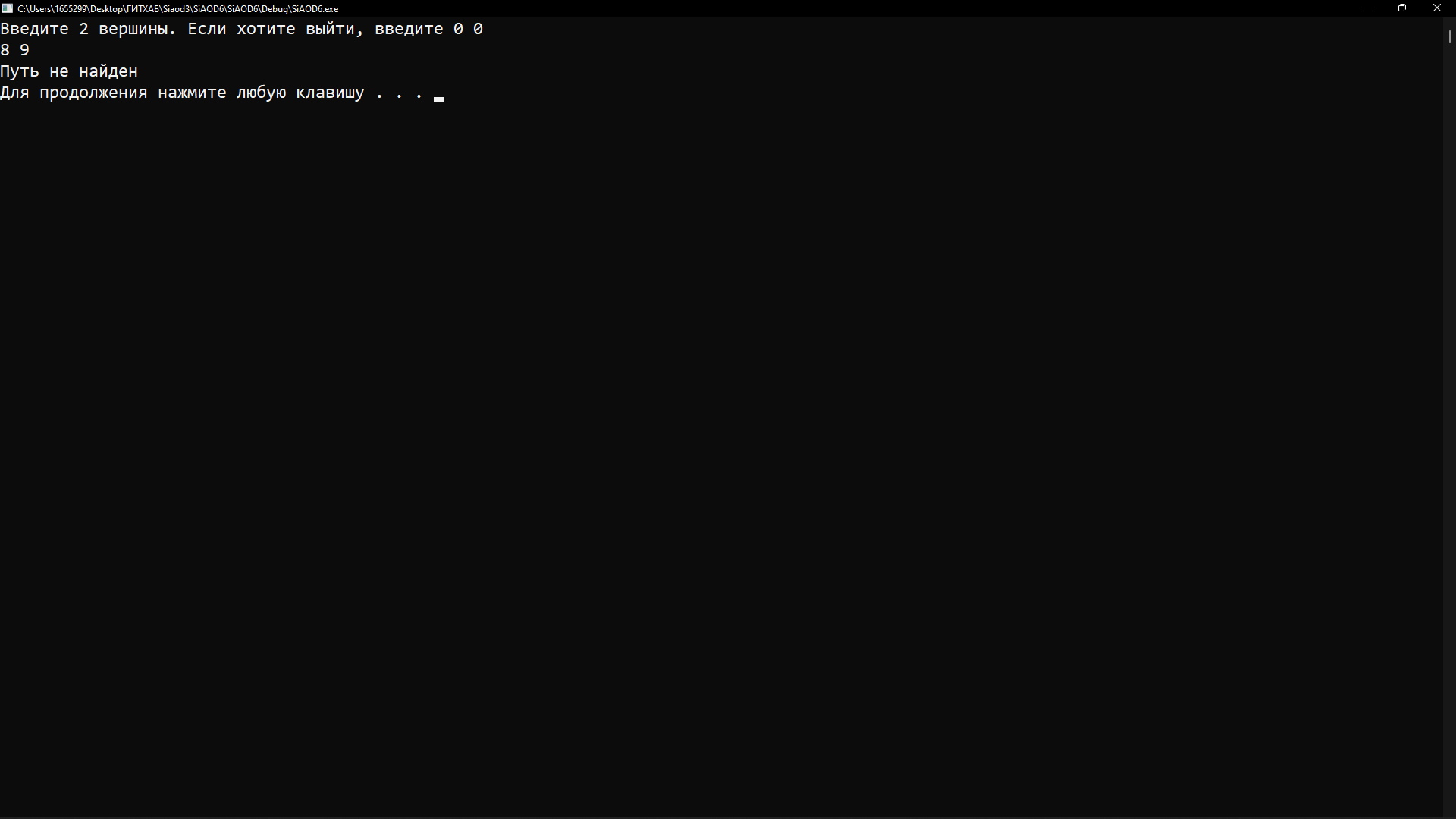


Рис. 9 вывод ошибки об отсутствии пути

Тест 10

Подтверждение закрытия программы представлено на рисунке 10.

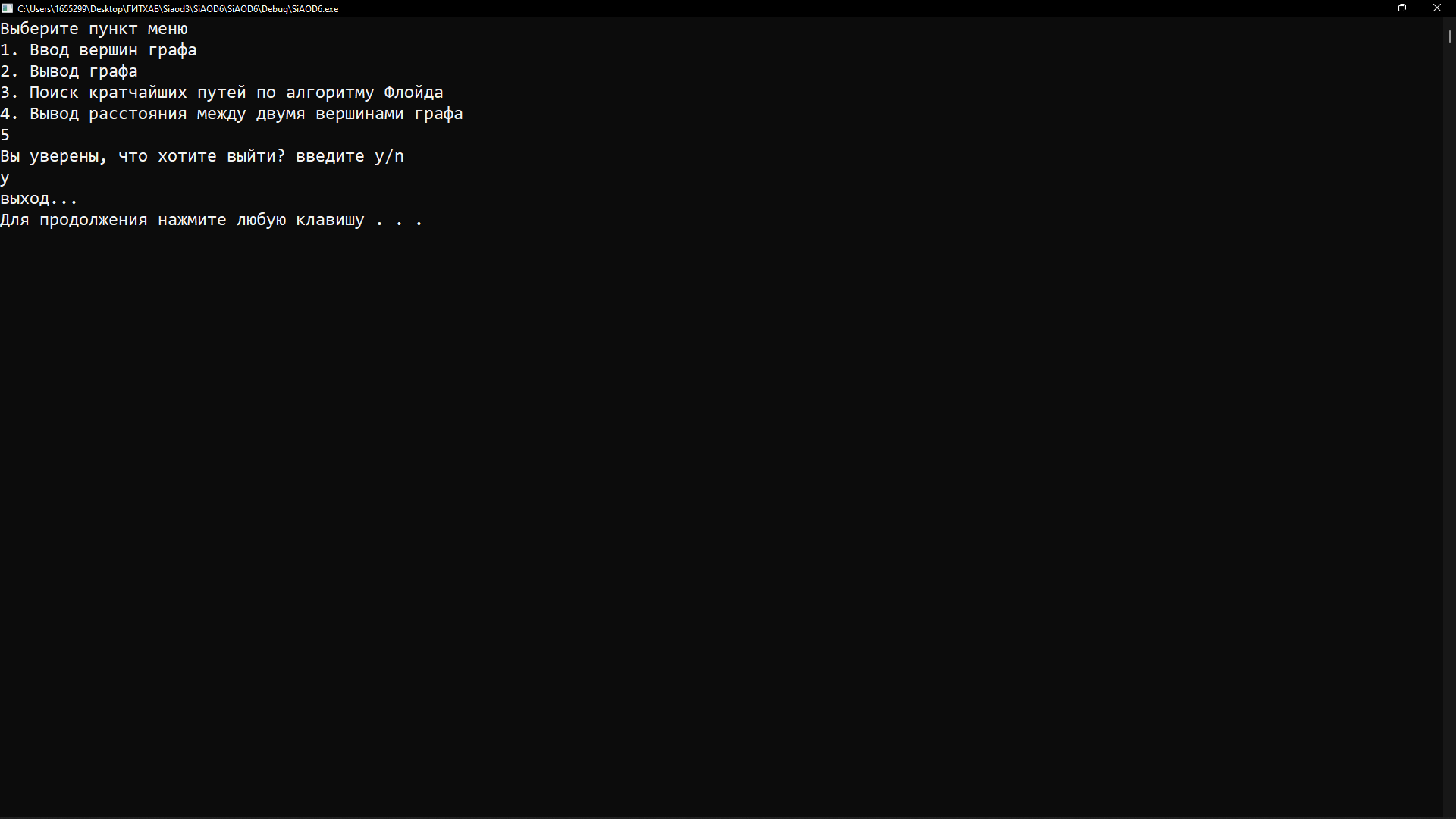


Рис. 10 подтверждение закрытия программы

Таким образом, после проведения тестирования можно подтвердить, что программа работает правильно.

# Вывод

В ходе выполнения практической работы были выполнены все задания, а также:

1. Изучен принцип работы графов;
2. Повторены навыки ООП;
3. Изучен принцип работы алгоритма Флойда.

# Исходный код программы

Source.cpp

|  |
| --- |
| #include <iostream>  #include <iomanip>  using namespace std;  const int Vmax = 100, Emax = Vmax \* 2;  int n, m, i, j, k, v, u, w;  char r;  class Graph  {  int head[Vmax] = {0};  int next\_el[Emax] = {0};  int terminal[Emax] = {0};  int weight[Emax] = {9999};  int summ[Emax][Emax] = { 0,0 };  int next[Emax][Emax] = { 9999,9999 };  int heads[Emax][Emax] = {9999,9999};  public:  void Add(int v, int u, int w);  void print();  void Floyd();  void Help();  int Printnew(int i,int j);  void BeautPrint();  bool IsGraph = false;  bool IsFloyd = false;  };  void Graph::Add(int v, int u, int w) //добавление ребра  {  k = k + 1;  terminal[k] = u;  next\_el[k] = head[v];  head[v] = k;  weight[k] = w;  }  void Graph::print()  {  cout << "Список смежности графа:";  for (i = 0; i<n + 1; i++)  {  j = head[i];  if (i)  {  cout << i << "->";  }  while (j>0)  {  if (!next\_el[j])  {  cout << terminal[j] << " (" << weight[j] << ")";  }  else  {  cout << terminal[j] << " (" << weight[j] << ")" << ", ";  }  j = next\_el[j];  }  cout << endl;  }  }  void Graph::Floyd()  {  for (int i = 0; i < n + 1; i++)  {  for (int j = 0; j < n + 1; j++)  {  next[i][j] = heads[i][j];  }  }  for (int i = 0; i < n + 1;i++)  {  for (int u=0; u < n + 1; u++)  {  for (int v = 0; v < n + 1; v++)  {  if (summ[u][i] + summ[i][v] < summ[u][v])  {  summ[u][v] = summ[u][i] + summ[i][v];  next[u][v] = next[u][i];  }  }  }  }  }  void Graph::Help()  {  for (int i = 0; i < Emax; i++)  {  for (int j = 0; j < Emax; j++)  {  summ[i][j] = 9999;  }  }  int z = 1;  while (z < n + 1)  {  j = head[z];  while (j > 0)  {  summ[z][terminal[j]] = weight[j];  heads[z][terminal[j]] = terminal[j];  j = next\_el[j];  }  z++;  }  }  int Graph::Printnew(int i, int j)  {  int c = i;  if (summ[i][j] < 9999)  {  cout << "Кратчайший путь между вершинами "<<i<<" и "<<j<<" составляет "<<summ[i][j]<<endl;  cout << "Путь лежит через вершины ";  while (c != j&&c!=0)  {  cout << c << " -> ";  c = next[c][v];  }  cout << j << endl;  return 1;  }  else  {  return 0;  }  }  void Graph::BeautPrint()  {  for (int i = 0; i < n + 1; i++)  {  for (int j = 1; j < n + 1; j++)  {  if (j == 1)  {  cout << i << " ";  }  if (i == 0)  {  cout << j << " ";  }  else  {  if (summ[i][j] == 9999)  {  cout << "er" << " ";  }  else  {  cout << summ[i][j] << " ";  }  }  }  cout << endl;  }  }  //главная функция  void main()  {  setlocale(LC\_ALL, "Rus");  k = 0;  int q;  Graph g;  while (true)  {  system("cls");  cout << "Выберите пункт меню" << endl;  cout << "1. Ввод вершин графа\n2. Вывод графа\n3. Поиск кратчайших путей по алгоритму Флойда\n4. Вывод расстояния между двумя вершинами графа" << endl;  cin >> q;  if (q == 1)  {  cout << "Ввод вершин графа" << endl;  cout << "Количество вершин >> ";  cin >> n;  cout << "Количество ребер >> ";  cin >> m;  cout << "В графе есть ориентированные ребра? y/n" << endl;  char f;  cin >> f;  if (f == 'y' || f == 'Y')  {  cout << "Вводите смежные вершины и вес пути:" << endl;  for (i = 0; i<m; i++)  {  cin >> v >> u >> w;  if (w < 0 || v < 0 || u < 0)  {  cout << "Отрицательные числа запрещены" << endl;  system("pause");  exit(0);  }  cout << "Ребро ориентированно? (y/n) >> ";  cin >> r;  if (r == 'y'||r=='Y')  {  g.Add(v, u, w);  }  else  {  g.Add(v, u, w);  g.Add(u, v, w);  }  cout << "..." << endl;  }  }  else  {  cout << "Вводите смежные вершины и вес пути:" << endl;  for (i = 0; i<m; i++)  {  cin >> v >> u >> w;  if (w < 0 || v < 0 || u < 0)  {  cout << "Отрицательные числа запрещены" << endl;  system("pause");  exit(0);  }  g.Add(v, u, w);  g.Add(u, v, w);  cout << "..." << endl;  }  }  g.Help();  g.IsGraph = true;  system("pause");  }  else if (q == 2)  {  if (g.IsGraph)  {  cout << "Вывод графа" << endl;  g.print();  system("pause");  }  else  {  cout << "Граф не введен" << endl;  system("pause");  }  }  else if (q == 3)  {  if (g.IsGraph)  {  cout << "Поиск кратчайших путей по алгоритму Флойда" << endl;  g.Floyd();  cout << "Вывод матрицы путей" << endl;  g.BeautPrint();  g.IsFloyd = true;  system("pause");  }  else  {  cout << "Граф не введен" << endl;  system("pause");  }  }  else if (q == 4)  {  if (g.IsGraph)  {  if (!g.IsFloyd)  {  cout << "Внимание!!!"<<endl<<"Алгоритм Флойда (поиск кратчайших путей) пока не был пройден."<<endl<<"Вывод кратчайшего пути будет согласно введенным весам."<<endl<<"Вы уверены, что хотите продолжить?y/n" << endl;  char k;  cin >> k;  if (k == 'y' || k == 'Y')  {  while (true)  {  system("cls");  cout << "Введите 2 вершины. Если хотите выйти, введите 0 0" << endl;  int d, f;  cin >> d >> f;  if (d != 0 && f != 0)  {  int h = g.Printnew(d, f);  if (h == 0)  {  cout << "Путь не найден" << endl;  }  }  else  {  break;  }  system("pause");  }  }  else  {  cout << "Вызовите пункт 3 меню, чтобы выполнить алгоритм Флойда" << endl;  system("pause");  }  }  else  {  while (true)  {  system("cls");  cout << "Введите 2 вершины. Если хотите выйти, введите 0 0" << endl;  int d, f;  cin >> d >> f;  if (d != 0 && f != 0)  {  int h = g.Printnew(d, f);  if (h == 0)  {  cout << "Путь не найден" << endl;  }  }  else  {  break;  }  system("pause");  }  }  }  else  {  cout << "Граф не введен" << endl;  system("pause");  }  }  else  {  cout << "Вы уверены, что хотите выйти? введите y/n"<<endl;  char b;  cin >> b;  if (b == 'Y' || b == 'y')  {  cout << "выход..."<<endl;  system("pause");  exit(0);  }  }  }  } |