

Содержание

[Реферат 5](#_Toc121735744)

[Введение. 6](#_Toc121735745)

[1. Теоретическая часть программы 8](#_Toc121735746)

[2. Описание алгоритма программы 10](#_Toc121735747)

[3. Пример ручного расчёта задачи 12](#_Toc121735748)

[4. Тестирование 13](#_Toc121735749)

[Заключение 17](#_Toc121735750)

[Список литературы 18](#_Toc121735751)

[Приложение A. Листинг программы 19](#_Toc121735752)

[Приложение Б. Результаты работы программы 25](#_Toc121735753)

Реферат

Отчет 26 стр, 16 рисунков.

АЛГОРИТМ ПОИСКА ГАМИЛЬТОНОВЫХ ЦИКЛОВ, ГРАФЫ, ОБХОД В ГЛУБИНУ.

Цель исследования – разработка программы, способная находить гамильтоновые пути от одной из вершин графа через все остальные веришны.

В работе рассмотрены правила построения графа из файла, автоматически, правила работы алгоритма, записи результатов в файл. Установлено, что с помощью данного алгоритма можно найти все гамильтоновые пути от одной из вершин графа через все остальные.

Введение.

В 1857 году математик Гамильтон придумал игру «Кругосветное путешествие». Она включала додекаэдр (правильный многогранник), двенадцать граней которого представляли собой равные правильные пятиугольники. В каждой из двадцати вершин тела просверливалась дырка, в которую вставлялся колышек, изображавший город (столицу). Используя верëвку, требовалось найти путь через города, посетив каждый город один раз, и вернуться в исходный город. Додекаэдр на плоскости изображается так, как показано на рисунке.

На языке теории графов задача сводится к нахождению в псевдографе простого цикла, проходящего через каждую вершину. Отсюда любой цикл псевдографа, обладающий таким свойством, называется гамильтоновым циклом.

Простая цепь (цикл) в псевдографе G называется *гамильтоновой (гамильтоновым),* если она (он) проходит через каждую вершину псевдографа G. Псевдограф, у которого есть гамильтонов цикл, называется *гамильтоновым графом.*

Очевидно, вопросы существования гамильтоновых цепей и циклов в псевдографах сводятся к аналогичным вопросам для простых графов. Следует отметить, что до сих пор неизвестны необходимые и достаточные условия существования в произвольном простом графе гамильтонова цикла или гамильтоновой цепи.

Однако имеется класс графов, в которых заведомо существуют гамильтоновы цепи и циклы – это полные графы. Таким образом, полнота графа является простейшим достаточным условием существования гамильтоновых цепей и циклов в простом графе.

Очевидным простейшим необходимым условием существования гамильтоновых цепей и циклов в простом графе является его связность. Более тонким необходимым условием существования гамильтоновых цепей и циклов в простом графе является отсутствие в нëм точек сочленения, т.е. вершин, удаление которых увеличивает число компонент связности.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio 2022, язык программирования – С++.

В ходе выполнения данной курсовой работы были приобретены навыки работы с формами и их элементами в среде MicrosoftVisualStudio 2022, навыки работы с проектами и многомодульными программами.

1. **Постановка задачи.**

Требуется разработать программу для нахождения гамильтоновых циклов.

Необходимо найти наиболее подходящий язык программирования для написания программы генерации перестановок, а также написать оптимальный код для реализации алгоритма. Не менее важной задачей является создание удобного, быстрого интерфейса программы, чтобы пользователь не испытывал проблем при работе.

***Необходимо определить, какой тип данных использовать.*** Он наиболее подходящий для реализации массива элементов заданного множества. Это необходимо для структуризации данных, а также при обработке данных в коде программы.

***Многофункциональность программы.*** Необходимо поделить программу на несколько функций, чтобы увеличить быстродействие программы.

***Устройства ввода – вывода – клавиатура и мышь.*** Необходимо научится анализировать и различать события, возникающие от работы с клавиатурой и мышью. Необходимо однозначно идентифицировать и выполнять те или иные действия в зависимости от действия пользователя. Это необходимо для легкого использования программы.

***Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.*** Необходимо изучить технологию построения меню. Это необходимо для создания интуитивно понятного интерфейса.

1. **Теоретическая часть программы**

Пусть G – псевдограф. Цепь (цикл) в G называется гамильтоновой (гамильтоновым), если она (он) проходит через каждую вершину псевдографа G ровно один раз. Простейшим достаточным условием существования гамильтоновых цепей и циклов в графе является его полнота. Граф G называется полным, если каждая его вершина смежна со всеми остальными вершинами. Необходимым условием существования гамильтоновых цепей и циклов в графе G является связность данного графа.

**Теорема Дирака.** Если в графе G(V, E) c n вершинами (n≥ 3) выполняется условие d(v) ≥n/2 для любого vV, то граф G является гамильтоновым.

**Доказательство.**

От противного. Пусть G— не гамильтонов. Добавим к G минимальное количество новых вершин u1, …, un, соединяя их со всеми вершинами G так, чтобы G’ := G+u1+ … +un был гамильтоновым.

Пусть v, u1, w, …, v— гамильтонов цикл в графе G’, причем vG, u1G’, u1G. Такая пара вершинvиu1в гамильтоновом цикле обязательно найдется, иначе граф G был бы гамильтоновым. Тогда wG, w {u1, …, un}, иначе вершинаu1была бы не нужна. Более того, вершина v несмежна с вершиной w, иначе вершинаu1была бы не нужна.

Далее, если в цикле v, u1, w, …, u’, w’, …, v есть вершина w’, смежная с вершиной w, то вершина v’ несмежна с вершиной v, так как иначе можно было бы построить гамильтонов цикл v, v’, …, w, w’, …, v без вершиныu1, взяв последовательность вершин w, …, v’ в обратном порядке. Отсюда следует, что число вершин графа G’, не смежных с v, не менее числа вершин, смежных с w. Но для любой вершины w графа G d(w) ≥p/2+n по построению, в том числе d(v) ≥p/2+n. Общее число вершин (смежных и не смежных с v) составляетn+p-1. Таким образом, имеем:

n+p-1 =d(v)+d(V) ≥d(w)+d(v) ≥p/2+n+p/2+n= 2n+p.

Следовательно, 0 ≥ n+1, что противоречит тому, что n > 0.

**Теорема Оре.** Если число вершин графа G(V, E) n≥ 3 и для любых двух несмежных вершин u и v выполняется неравенство:

d(u)+d(v) ≥nи(u, v)E, то граф G— гамильтонов.

Граф G имеет гамильтонов цикл если выполняется одно из следующих условий:

**Условие Поша:** d(vk) ≥k+1 для k < n/2.

**Условие Бонди:** из d(vi) ≤i и d(vk)≤ k => d(vi)+d(vk)≥n(k≠i)

**Условие Хватала:** из d(vk)≤ k ≤ n/2 => d(vn-k)≥ n-k.

1. **Описание алгоритма программы**

Программа работает следующим образом: пользователь выбирает один вариант заполнения матрицы смежности из двух (заполнение случайными числами или считывание матрицы из файла). Если пользователь выбрал первый вариант, то программа просит ввести размер матрицы смежности. Если же он выбирает второй вариант, то программа просит ввести имя файла, с которого будет считана матрица смежности. Далее пользователь вводит вершину, с которой хочет начать обход, и программа вызывает процедуру определения гамильтоновых циклов в графе. Вызывается процедура, основанная на обходе в глубину для определения всех гамильтоновых циклов. После завершения алгоритма пользователю выводится на экран все гамильтоновы циклы.

Ниже представлен псевдокод алгоритма

Функция hamCycle

Вход: неориентированный граф graph, представленный матрицей смежности, N - количество вершин в графе, versh начальная вершина

Алгоритм

1. Создать пустой список path и массив посещённых вершин visited
2. Поместить versh в список
3. Пометить versh как посещённую вершину
4. Вызвать дополнительную функцию поиска гамильтоновых циклов FindHamCycle

Фунцкия FindHamCycle

Вход: неориентированный граф graph, представленный матрицей смежности, список вершин path, массив посещённых вершин visited, количество вершин N, начальную вершину versh

Выход: гамильтоновые циклы на экране

Алгортим

1. Если (все вершины посещены)
   1. Если (если последняя вершина смежна с начальной)
      1. Поместить начальную вершину versh в список
      2. Для (int i = 0; i <размера path; i++)
         1. Вывод на экран path[i]
         2. Вытолкнуть вершину из списка

1.2 Возврат

2. Для каждой v<N выполнять

2.1 Если (вершина v смежна с ранее проверяемой вершиной и она не посещена)

2.1.1 Поместить v в список path

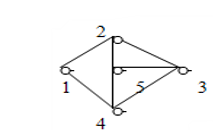
2.1.2 Пометить v как посещённую

2.1.3 Вызвать рекурсивно функцию FindHamCycle

2.1.4 Пометить v как непосещённую

2.1.5 Вытащить вершину из списка

1. **Пример ручного расчёта задачи**

Пусть матрица смежности задана графом G (рис. 1). Возьмём вершину 1. Для вершины 1 смежными являются вершины 2 и 4. Пойдём в вершину 2. Для неё смежными являются вершины 3 и 5. Пойдём в вершину 3. Для этой вершины смежными являются вершины 4 и 5. Следуем в вершину 5, для того чтобы мы смогли обойти все вершины графа. Остаётся только вершина 4, а из неё мы попадаем в нашу начальную вершину.

Таким образом, наш гамильтонов путь 1->2->3->5->4->1.

Аналогично находим все остальные пути 1->2->5->3->4->1, 1->4->3->5->2->1, 1->4->5->3->2->1.

1. **Тестирование**

Выбираем нужное для нас действие, например 1. Рандомный граф.

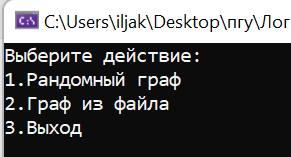


Рис. 2

Программа прости ввести размер матрицы смежности. Вводим 6.

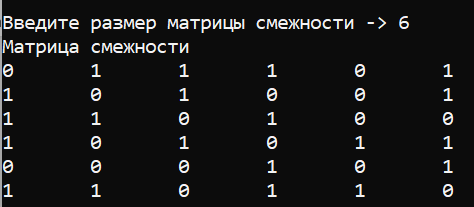


Рис. 3

Далее вводим нужную для нас начальную вершину обхода графа от 1 до 6.

Если мы введём вершину, не принадлежащую множеству от 1 до 6, программа оповестит нас о неправильном вводе и попросит заново ввести вершину.



Рис. 4

При правильном вводе вершины программа начинает обход графа и выводит на экран все гамильтоновые циклы. На этом поиск завершается.

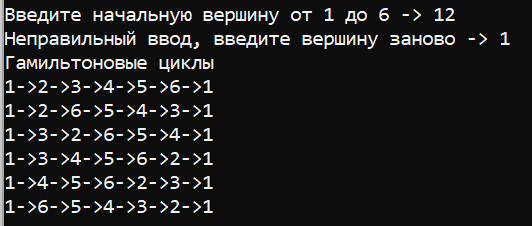


Рис. 5

Если же в графе отсутствуют гамильтоновые циклы, программа оповестит об этом.

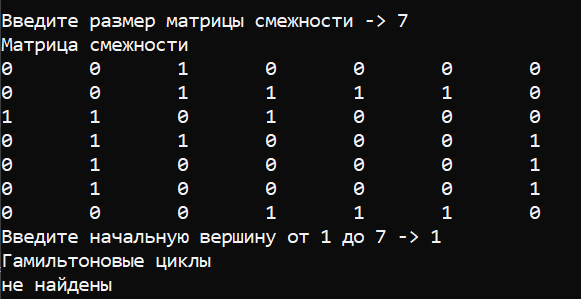


Рис. 6

Проверка на различных наборах данных

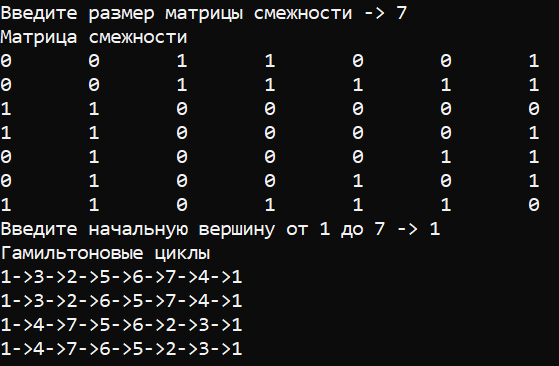


Рис. 7

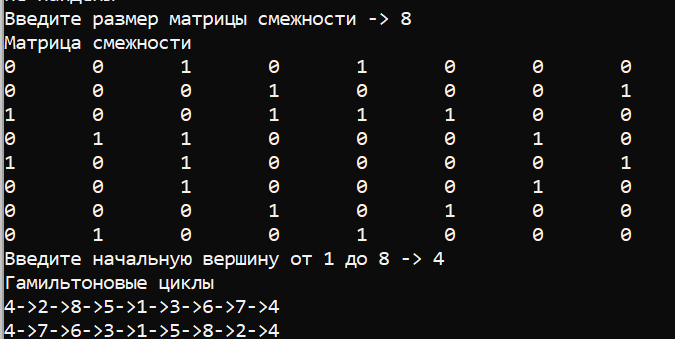


Рис. 8

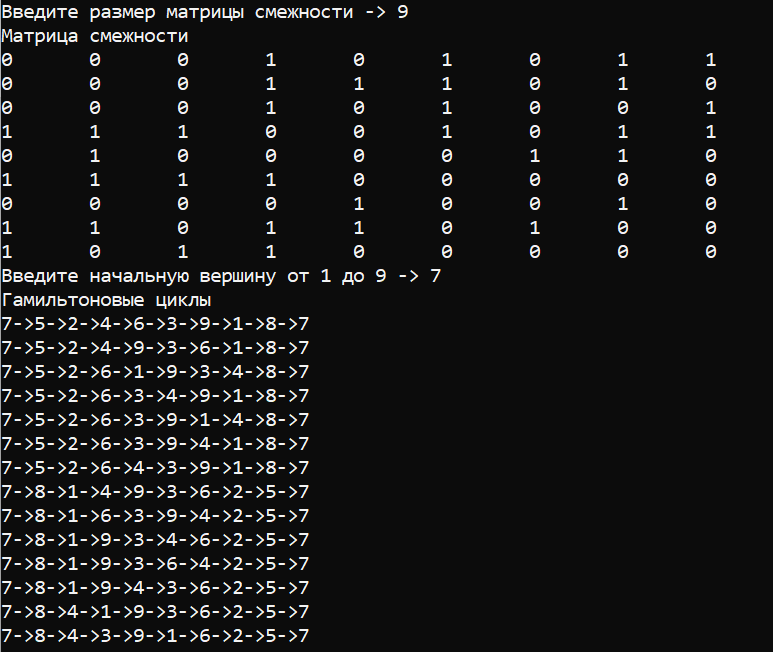


Рис. 9

2. Граф из файла.

Программа прости ввести файл, из которого будет считана матрица смежности (для удобства в папке с проектом создан файл “F1.txt”). Вводим имя нужного нам файла.

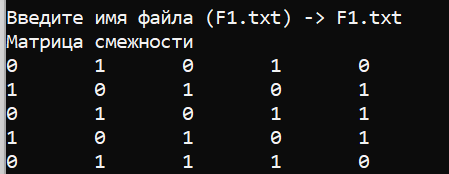


Рис. 10

Как и в первом случае, программа просит ввести нужную нам начальную вершину. После этого запускается алгоритм поиска всех гамильтоновых циклов.

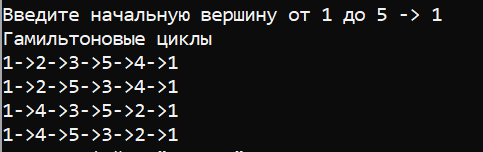


Рис. 11

Для удобства проверки ответ записывается в файл “F2.txt”.



Рис. 12

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод меню программы | Верно |
| Выбор генерации матрицы | Переход к выбранному действию | Верно |
| Вывод матрицы, введенной автоматически, задание начальной вершины и поиск гамильтоновых путей в графе | Вывод матрицы, ввод начальной вершины вывод и результата поиска | Верно |
| Вывод матрицы из файла и поиск гамильтоновых путей в графе | Вывод матрицы, ввод начальной вершины и вывод результата поиска | Верно |
| Запись в файл матрицы и результата поиска | Запись производится в файл | Верно |

Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм поиска гамильтоновых путей в графе в MicrosoftVisualStudio 2022.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей. Приобретены навыки по осуществлению алгоритма поиска гамильтоновых путей. Углублены знания языка программирования Cи и Си++.

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. Язык программирования С [2009] Керниган, Ритчи
2. Expert C Programming Deep C Secrets Peter van der Linden
3. Полный справочник по C [2002] Шилдт Герберт
4. Алгоритмы и программы язык C++ [2017] Е. А. Коннова, Г. А. Поллак
5. Стивен Прата "Язык программирования С++. Лекции и упражнения

**Приложение A. Листинг программы**

Файл “menu.cpp”

#include<stdio.h>

#include<Windows.h>

void menu()

{

printf("Курсовая работа на тему: Алгоритм нахождения гамильтоновых циклов в графе\n");

printf("Выполнил: Костюков Илья 21ВВ3");

Sleep(3000);

system("cls");

printf("Выберите действие:\n");

printf("1.Рандомный граф\n");

printf("2.Граф из файла\n");

printf("3.Выход\n\n");

}

Файл “ham.cpp”

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<vector>

#include<stdio.h>

#include<Windows.h>

bool hasCycle;

FILE \*F2;

char F2name[20], copy;

void printgraph(intgraph[5][5])// вывод матрицы смежности на экран

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

printf\_s("%d \t", graph[i][j]);

}

printf\_s("\n");

}

}

bool isSafe(intv, intgraph[5][5], std::vector<int>path, intpos) //фунцкияпроверкивершинынаналичиевгамильтоновомцикле

{

if (graph[path[pos - 1]][v] == 0)//проврека на смежность ранее добавленной вершины с проверяемой

{

returnfalse;//если они несмежны

}

for (int i = 0; i <pos; i++)//исключение повтора вершины в массиве

{

if (path[i] == v)

{

returnfalse;//если она уже была в гамильтоновом пути

}

}

returntrue;

}

void FindHamCycle(intgraph[5][5], intpos, std::vector<int>path, boolvisited[], intN, intversh) //

{

if (pos == N) //если все вершины просмотрены

{

if (graph[path[path.size() - 1]][path[0]] != 0)//если последняя веришна смежна с начальной, вывод на экран гамильтонового цикла

{

path.push\_back(versh);

F2 = fopen(F2name, "a+");

for (int i = 0; i <path.size(); i++)

{

printf\_s("%d->", path[i] + 1);

fprintf(F2, "%d->", (path[i] + 1));

}

printf\_s("\b\b");

printf\_s("");

printf\_s("\n");

fprintf(F2, "\n");

path.pop\_back();

hasCycle = true;

}

return;

}

for (int v = 0; v <N; v++)//просмотр разных вершин на наличие в гамильтоновом цикле

{

if (isSafe(v, graph, path, pos) && !visited[v])//проверка данной вершины на наличие в гамильтоновом цикле

{

path.push\_back(v);

visited[v] = true;

FindHamCycle(graph, pos + 1, path, visited, N, versh);//рекурсиядляпроверкидругихвершин

visited[v] = false;

path.pop\_back();

}

}

}

void hamCycle(intgraph[5][5], intN, intversh, FILE\* F1)//началоалгоритмапоискагамильтоновогоциклавграфе

{

SetConsoleCP(1251);

printf\_s("Введите имя файла, в который будет записан ответ: ");

scanf\_s("%s", &F2name, 20);

F1 = fopen("F1.txt", "r");

F2 = fopen(F2name, "w+");

fprintf(F2, "Матрицасмежности:\n");

while ((copy = getc(F1)) != EOF)

{

putc(copy, F2);

}

fprintf(F2, "\n");

fprintf(F2, "Гамильтоновыециклы\n");

printf\_s("Гамильтоновые циклы\n");

hasCycle = false;

std::vector<int> path;

path.push\_back(versh);

bool visited[5];

for (int i = 0; i <N; i++)

{

visited[i] = false;

}

visited[versh] = true;

FindHamCycle(graph, 1, path, visited, N, versh);//вызовфунцкиипоискагамильновогоцикла

if (!hasCycle) {

printf\_s("ненайдены");

return;

}

fclose(F2);

}

void printgraph(int\*\* graph, intn)

{

for (int i = 0; i <n; i++)

{

for (int j = 0; j <n; j++)

{

printf\_s("%d \t", graph[i][j]);

}

printf\_s("\n");

}

}

bool isSafe(intv, int\*\* graph, std::vector<int>path, intpos)

{

if (graph[path[pos - 1]][v] == 0)

{

returnfalse;

}

for (int i = 0; i <pos; i++)

{

if (path[i] == v)

{

returnfalse;

}

}

returntrue;

}

void FindHamCycle(int\*\* graph, intpos, std::vector<int>path, boolvisited[], intN, intversh)

{

if (pos == N)

{

if (graph[path[path.size() - 1]][path[0]] != 0)

{

path.push\_back(versh);

for (int i = 0; i <path.size(); i++)

{

printf\_s("%d->", path[i] + 1);

}

printf\_s("\b\b");

printf\_s("");

printf\_s("\n");

path.pop\_back();

hasCycle = true;

}

return;

}

for (int v = 0; v <N; v++)

{

if (isSafe(v, graph, path, pos) && !visited[v])

{

path.push\_back(v);

visited[v] = true;

FindHamCycle(graph, pos + 1, path, visited, N, versh);

visited[v] = false;

path.pop\_back();

}

}

}

void hamCycle(int\*\* graph, intN, intversh)

{

hasCycle = false;

std::vector<int> path;

path.push\_back(versh);

bool\* visited = (bool\*)malloc(N\*sizeof(bool));

for (int i = 0; i <N; i++)

{

visited[i] = false;

}

visited[versh] = true;

FindHamCycle(graph, 1, path, visited, N, versh);

if (!hasCycle) {

printf\_s("ненайдены\n");

return;

}

}

Файл “Курсовая.cpp”

#define\_CRT\_SECURE\_NO\_WARNINGS

#include"menu.h"

#include"ham.h"

#include<iostream>

#include<vector>

#include<conio.h>

#include<Windows.h>

#include<time.h>

usingnamespace std;

FILE\* F1;

char F1name[20] = ("F1.txt");

void gif()

{

int graph[5][5];

char name[10];

printf\_s("Введитеимяфайла (F1.txt) ->");

scanf\_s("%s", &name, 10);

if (strcmp(name, F1name) == 0)

{

if (F1 = fopen(F1name, "r"))

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

fscanf(F1, "%d", &graph[i][j], 5, 5);

}

}

printf\_s("Матрица смежности\n");

printgraph(graph);

int versh;

printf\_s("Введите начальную вершину от 1 до %d ->", 5);

do

{

scanf\_s("%d", &versh);

if (versh < 1 || versh > 5)

{

printf\_s("Неправильный ввод, введите вершину заново ->");

}

} while (versh < 1 || versh > 5);

versh--;

hamCycle(graph, 5, versh, F1);

fclose(F1);

}

else

{

printf\_s("Файланет");

}

}

else

{

printf\_s("Такого файла не существует\n");

}

}

void rg()

{

srand(time(NULL));

int n, versh;

printf("Введите размер матрицы смежности ->");

scanf\_s("%d", &n);

int\*\* graph = (int\*\*)malloc(n \* sizeof(int\*));

for (int i = 0; i < n; i++)

{

graph[i] = (int\*)malloc(n \* sizeof(int));

}

for (int i = 0; i < n; i++)

{

for (int j = i; j < n; j++)

{

graph[i][j] = rand() % 2;

if (i == j)

{

graph[i][j] = 0;

}

graph[j][i] = graph[i][j];

}

}

printf("Матрицасмежности\n");

printgraph(graph, n);

printf\_s("Введите начальную вершину от 1 до %d ->", n);

do

{

scanf\_s("%d", &versh);

if (versh < 1 || versh > n)

{

printf\_s("Неправильный ввод, введите вершину заново ->");

}

} while (versh < 1 || versh > n);

versh--;

printf\_s("Гамильтоновые циклы\n");

hamCycle(graph, n, versh);

}

int main()

{

SetConsoleOutputCP(1251);

menu();

while (true)

{

int s = \_getch();

switch (s)

{

case'1': rg(); break;

case'2': gif(); break;

case'3': return 0;

}

}

}

**Приложение Б. Результаты работы программы**

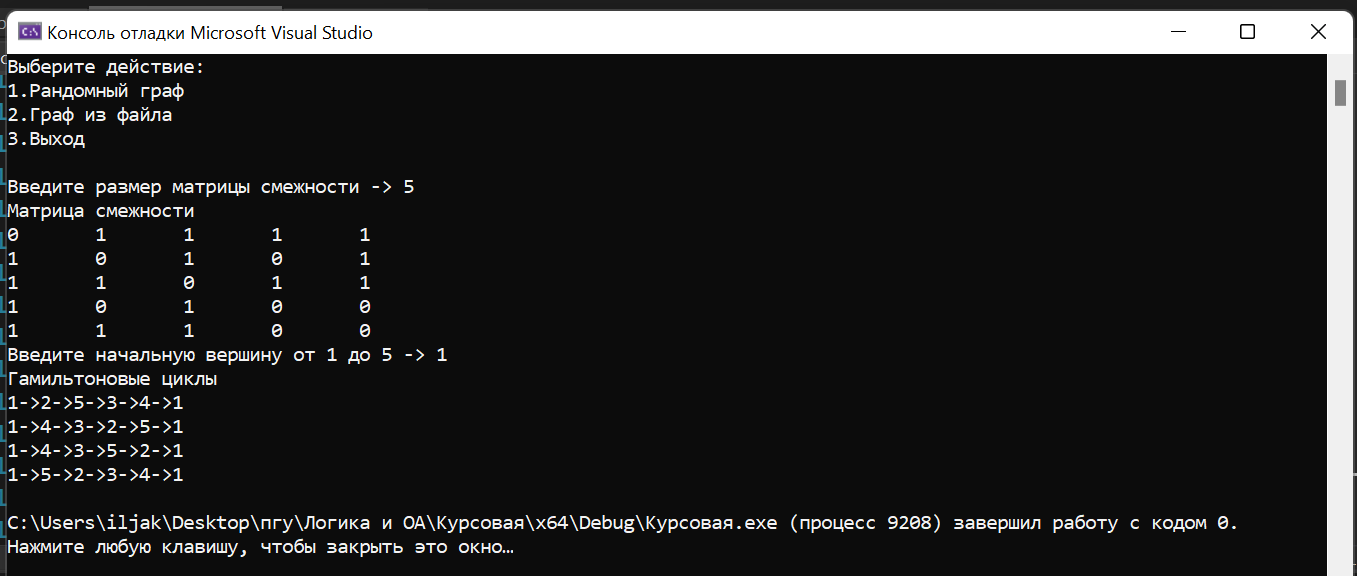
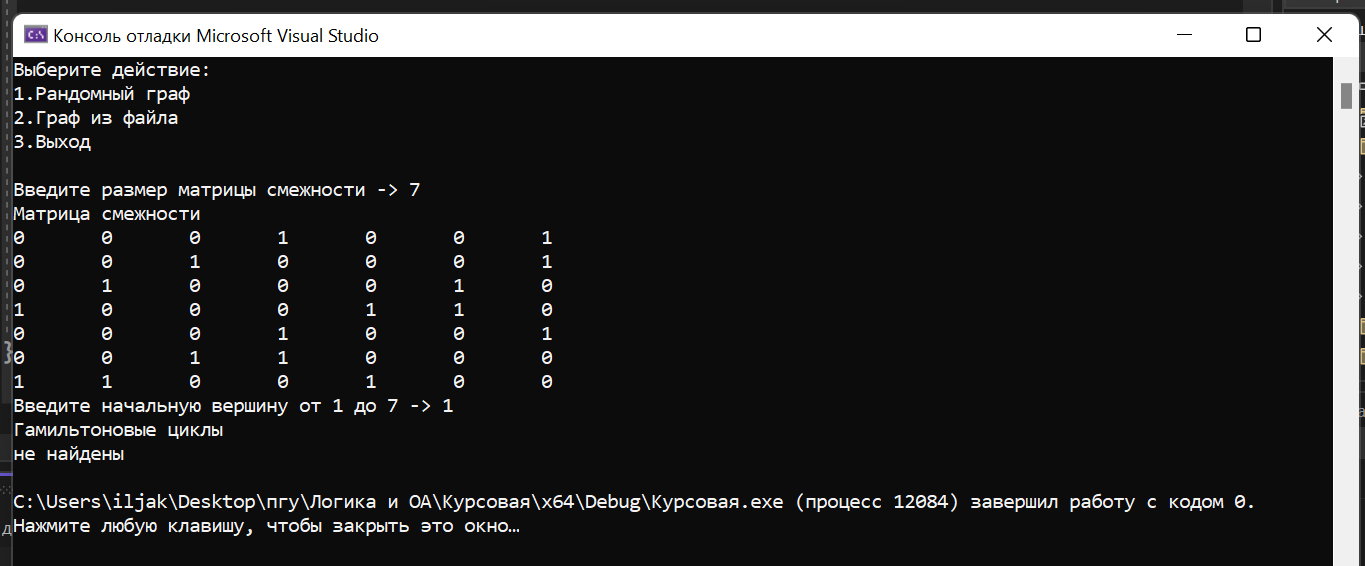


Рис 13.

 Рис. 14

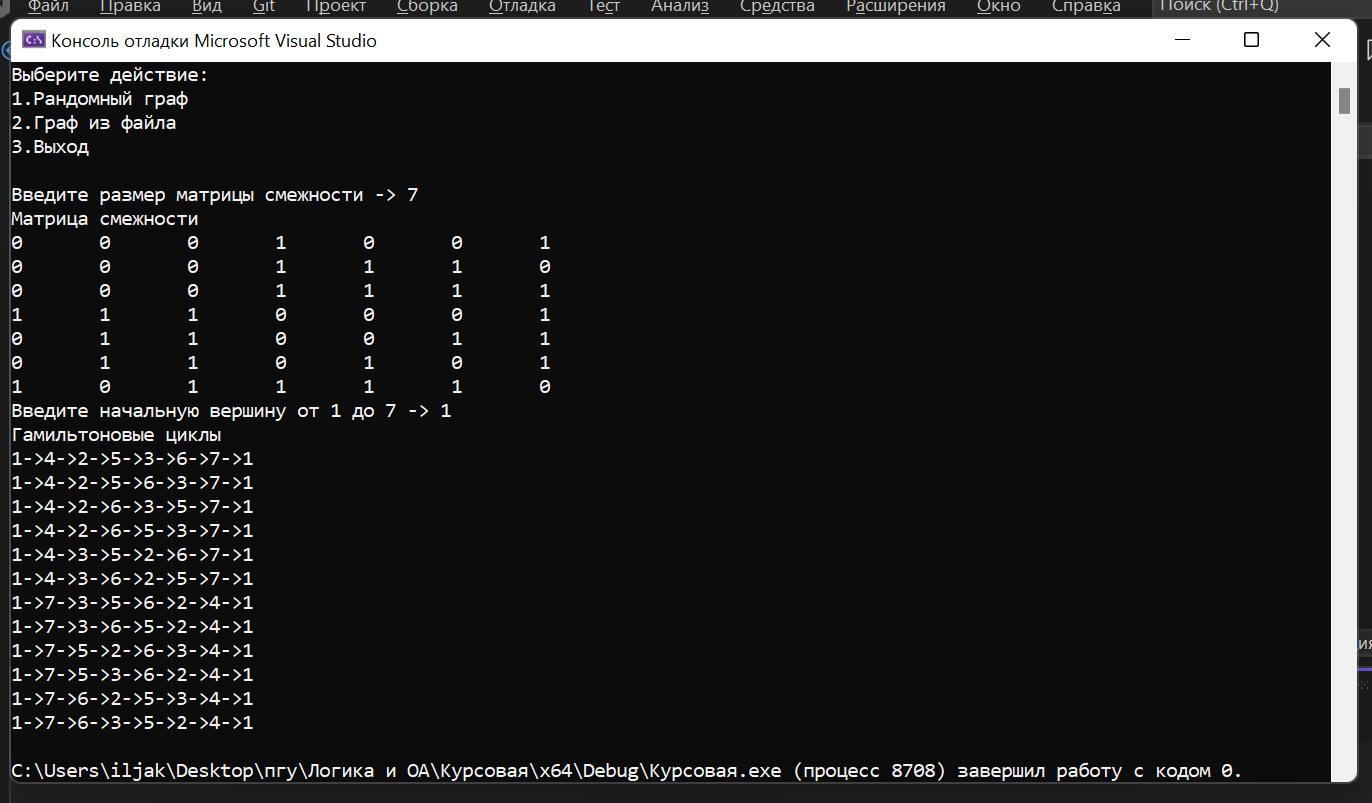


Рис. 15

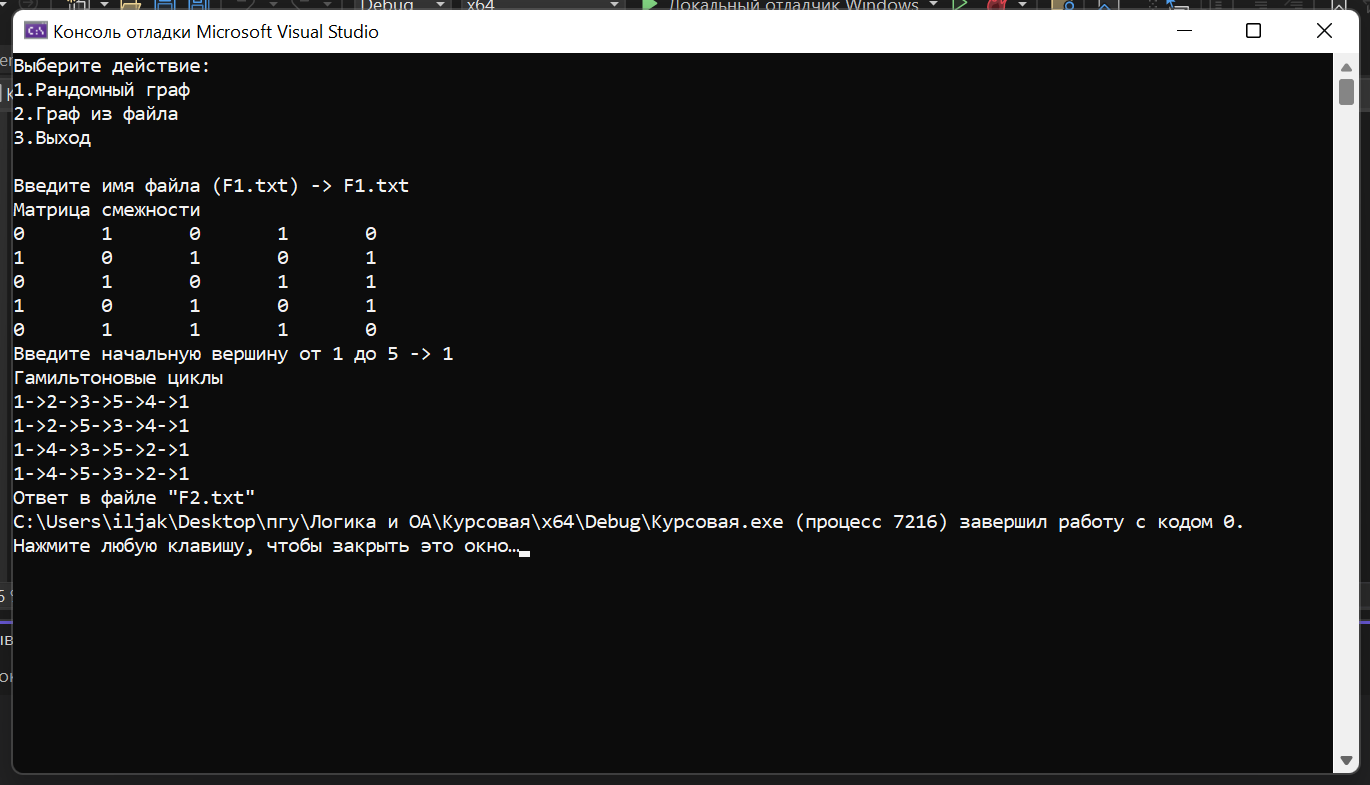


Рис. 16