

Содержание

[Реферат 5](#_Toc123025902)

[Введение 6](#_Toc123025903)

[1. Постановка задачи 7](#_Toc123025904)

[2. Теоретическая часть задания 8](#_Toc123025905)

[3. Описание алгоритма программы 9](#_Toc123025906)

[4. Описание программы 11](#_Toc123025907)

[5. Тестирование 15](#_Toc123025908)

[6. Ручной расчёт задачи 19](#_Toc123025909)

[Заключение 21](#_Toc123025910)

[Список литературы 22](#_Toc123025911)

[Приложение А 23](#_Toc123025912)

# Реферат

Отчёт стр. 25 , рисунков 24, таблиц 1.

ГРАФ, ТЕОРИЯ ГРАФОВ, ДОСТИЖИМОСТЬ, ПОИСК В ГЛУБИНУ, ЭЙЛЕРОВЫЕ ЦИКЛЫ, ЭЙЛЕРОВЫЕ ГРАФЫ.

Цель исследования – разработка программы, которая находит и выводит Эйлеровый цикл.

В работе рассмотрены правила поиска в глубину, на основе которых находятся Эйлеровый циклы.

# Введение

Целью данной курсовой работы является создание программы, которая будет искать Эйлеровые циклы в графе.

Эйлеров цикл – Эйлеров путь, являющийся циклом, то есть замкнутый путь, проходящий через каждое ребро графа по одному разу. Для того, чтобы в графе существовал Эйлеров цикл необходимо следующее. Во-первых, граф должен быть связанным: для любых двух вершин должен существовать путь, их соединяющий. Во-вторых, для неориентированных графов число ребер в каждой вершине должно быть четным. На самом деле этого оказывается достаточно.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2019, язык программирования – С.

# Постановка задачи

Требуется разработать программу, которая представит граф в виде матрицы смежности, проверит его на возможность нахождения цикла и найдёт циклы Эйлера.

Программа должна иметь текстовое или графическое меню.

При генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы.

После обработки этих данных пользователь вручную заполнял матрицу. Необходимо сделать возможность случайного заполнения матрицы.

Необходимо предусмотреть различные исходы поиска, чтобы программа не выдавала ошибок и работала правильно.

Результат выполнения программы должен сохраняться в файл.

# Теоретическая часть задания

Для того, чтобы найти Эйлеровый цикл в графе необходима знать два основных правила. Во-первых, граф должен быть связный, то есть не должно быть изолированных вершин. Во-вторых, все степени вершин графа должны быть чётные.

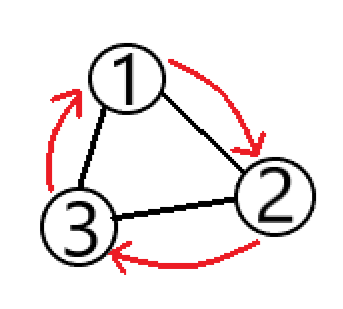
Рассмотрим пример графа из 3 вершин.

Рисунок 1 – пример графа

Как мы видим на (Рисунок 1) граф удовлетворяет всем условия и имеет Эйлеровый цикл.

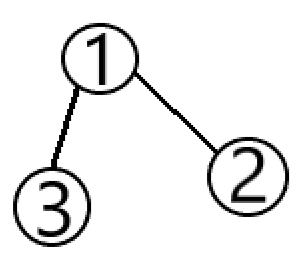
Рассмотрим другой граф из 3 вершин.

Рисунок 2 – пример графа

Как мы видим на (Рисунок 2) граф не удовлетворяет всем условия, степени 2-ой и 3-й вершины не чётные. Граф не имеет Эйлеровый цикл.

# Описание алгоритма программы

Для реализации алгоритма программы понадобиться две функции: eiler() и DFS().

Описание функции DFS, которая проходит по графу и ищет Эйлеровый цикл.

void DFS(int start, int n, int\*\* G) { /\* start – первая вершина, n – количество вершин, G – матрица \*/

st.push(start); // заносим вершину в стек

пока стек не пустой

{

int v = st.top(); // возвращаем верхний элемент стека в v

int i; // переменная для цикла

для (i = 0;i < n;i++) // цикл прохода по всем вершинам

если есть связь между стартовой вершиной и i

прервать;

если весь цикл пройден,

{

вставляем элемент из стека в конец вектора посещений

удаляет элемент из начала стека

}

иначе

{

удаляем связь между 2 найденными вершинами

записываем найденную вершину в стек

}

}

}

Описание функции eiler, которая проверяет граф на наличие Эйлерового цикла.

int eiler(int n, int\*\* G, int vert) {

vector<int> deg(n); // динамический массив данных

для (int i = 0; i < n; ++i) // цикл прохода по матрице

для (int j = 0; j < n; ++j)

deg[i] += G[i][j]; // в ячейку массива записывается кол-во связей одной вершин

для (int i = 0;i < n;i++)

если ((deg[i] > 0) и (deg[i] % 2 == 0)) // если вершина не изолированная и степени вершин четные, то флаг = true

иначе

{

// flag на false, выводим сообщение, выходим из функции

}

Выводим сообщение об нахождении цикла

Если цикл найден, создаем новую матрицу

для (int k = 0;k < n;k++) // Запускаем цикл, чтобы пройти граф из каждой вершины

{

для (int i = 0;i < n;i++)

для (int j = 0;j < n;j++)

H[i][j] = G[i][j]; // Копируем матрицу

вызываем функцию DFS с новой матрицей

для (size\_t j = 0; j < res.size();++j) // выводим вектор с результатом в консоль // и файл

очищаем вектор и выходим из функции

# Описание программы

Функция eiler() нужна для проверки заданного графа на Эйлеровость. Для этого заполню вектор deg степенью связи для каждой вершины, тем самым возможно будет проверить сразу все условия для Эйлерового цикла.

Перебрав все вершины в простом цикле for и прогнав через условие if ((deg[i] > 0) && (deg[i] % 2 == 0)) можно узнать имеет ли граф Эйлеровый цикл или нет. Если условие не выполняется, то в консоль выводится сообщение и программа заканчивается. Если условие выполняется для всех вершин графа, то функция eiler() вызовет обход в глубину DFS() из каждой вершины.

Обход в глубину будет модифицирован. В функции используется стек и вектор. При прохождении вершина не помечается как посещенная, а удаляется ребро, по которому мы попали в эту вершину.

Вершины, по мере посещения записываются в стек. Если программа заходит в «тупик», то есть в вершину у которой удалены все рёбра, то значение из стека переходит в вектор, а обход возвращается на шаг назад, в итоге в векторе будет искомый цикл.

Программа начинается с вывода в консоль текстового меню. Пользователю предлагается задать количество вершин в графе. Если пользователь ввёл меньше 1 вершины, программа выведет в консоль сообщение «Задано слишком мало вершин!» и запросит повторный ввод (Рисунок 3).

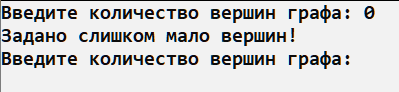


Рисунок 3 – пример ошибки ввода вершин

Если пользователь ввел правильно число вершин, то программа выведет текстовое меню. В меню пользователю предлагают выбрать вариант заполнения матрицы смежности: заполнить матрицу случайным образом с помощью функции rand (), нажав клавишу 1, либо заполнить матрицу вручную с клавиатуры, нажав клавишу 2. Если пользователь не нажал одну из этих клавиш, программа выведет в консоль сообщение «Выбран неверный вариант!» и запросит выбор повторно (Рисунок 4). Если пользователь нажмет клавишу 3 программа завершится. (Рисунок 5).

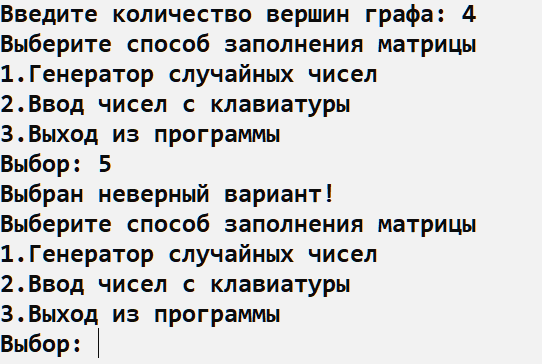


Рисунок 4 – пример ошибки выбора

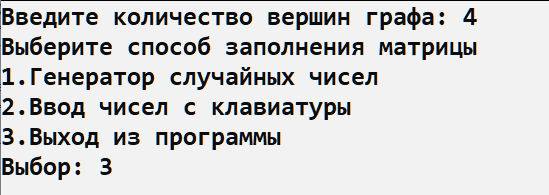


Рисунок 5 –выбор выхода из программы

Выбор пользователем режима заполнения матрицы был реализован с помощью функции switch ().

При заполнении матрицы смежности пользователем учитывается, что граф будет не взвешенный, то есть можно будет ввести только 0 и 1.

При случайно заполнении матрицы учитывается, что размер матрицы больше 1.

После заполнения матрицы программа выводит её в консоль и в файл, который называется «база.txt» (Рисунок 6,7)

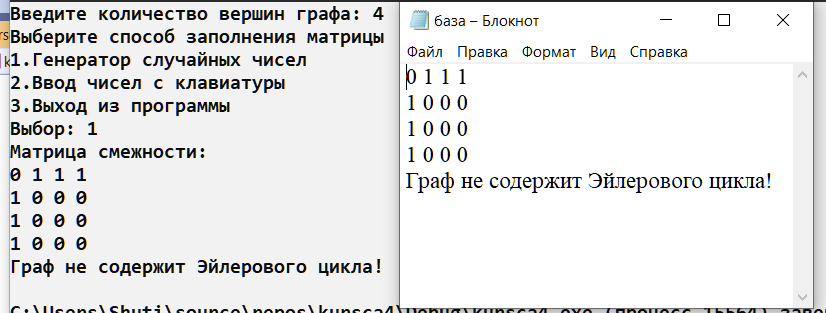


Рисунок 6 – вывод случайно матрицы в консоль и файл

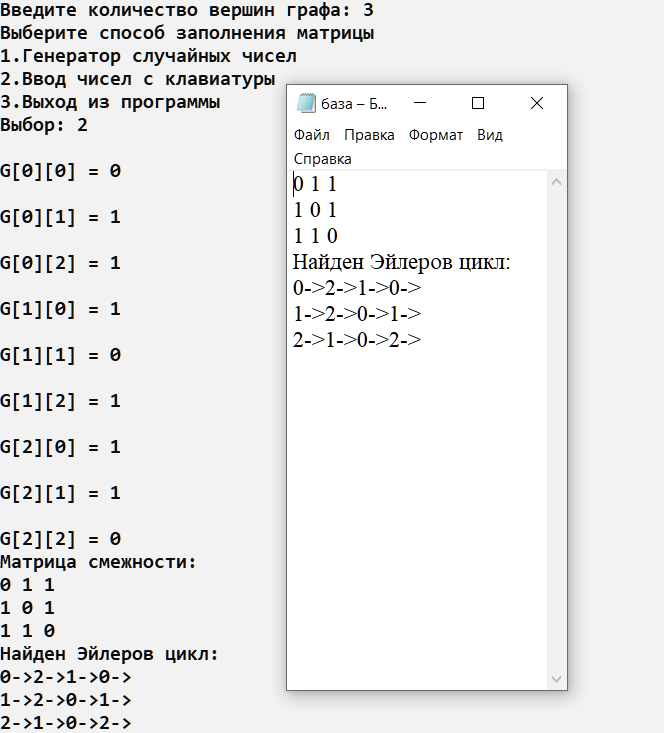


Рисунок 7 – ручной ввод матрицы, вывод в консоль и файл

После создания матрицы программа вызывает функцию eiler (), работа которой объяснена в «Описание алгоритма программы»

В результате программа выведет либо «Граф не содержит Эйлеровый цикл!» (Рисунок 9), либо сам цикл (Рисунок 8).

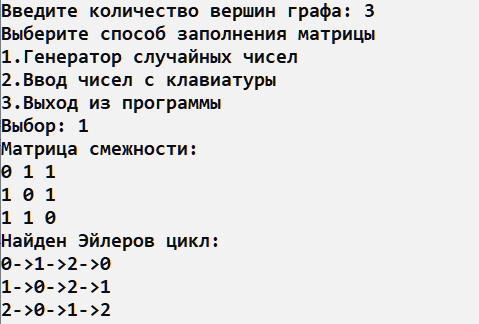


Рисунок 8 – вывод циклов

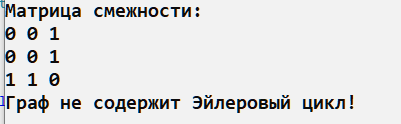


Рисунок 9 – вывод сообщения

Если граф не содержит Эйлеровый цикл, программа вернет пользователя в меню выборы (Рисунок 10).

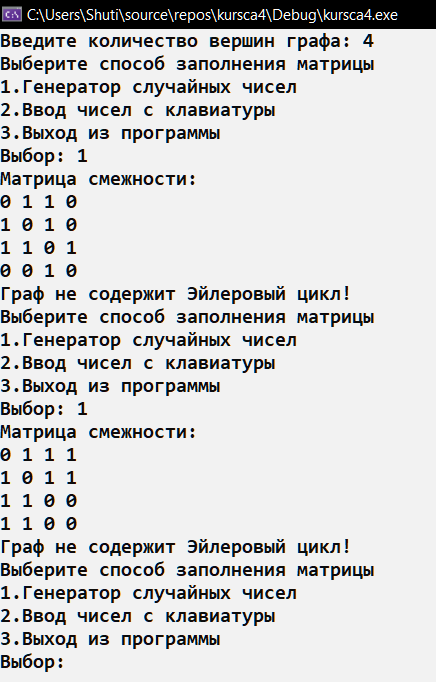


Рисунок 10 – возвращение в меню

# Тестирование

Выполнение программы с 3 вершинами в графе при случайном заполнении матрицы (Рисунок 11)

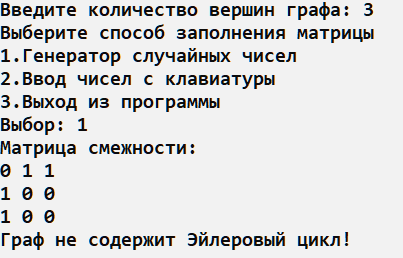


Рисунок 11 – тестирование случайного заполнения

Выполнение программы с 3 вершинами в графе при ручном заполнении матрицы (Рисунок 12)

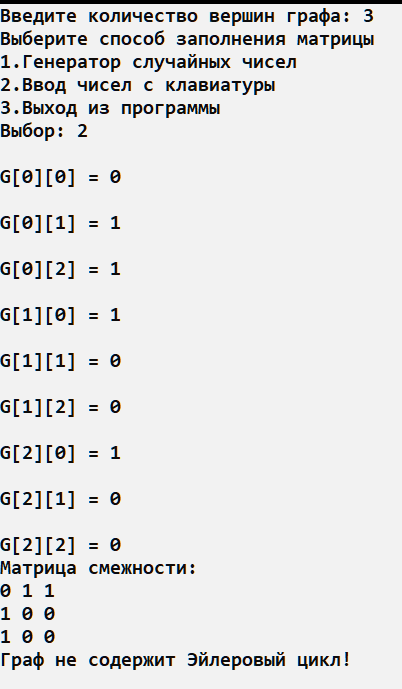


Рисунок 12 – тестирование ручного заполнение

Выполнение программы с не корректным вводом вершин

(Рисунок 13)

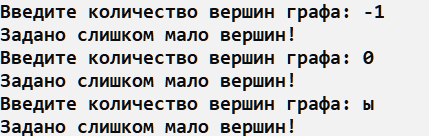


Рисунок 13 – тестирование матрицы с неправильным вводом

Выполнение программы с не корректным вводом рёбер

(Рисунок 14)

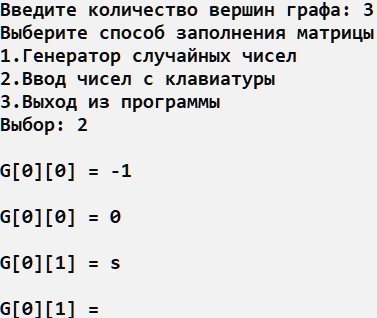


Рисунок 14 – Тестирование с неправильным вводом

Выполнение программы с выводом цикла (Рисунок 15)

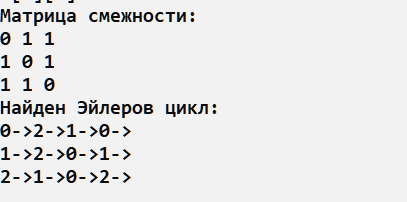


Рисунок 15 – Вывод цикла

Выполнение программы с выводом сообщения (Рисунок 16)

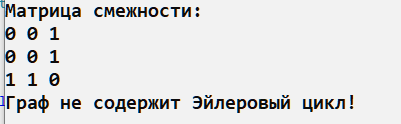


Рисунок 16 – Вывод сообщения

Выполнение программы с возвращением в меню (Рисунок 17).

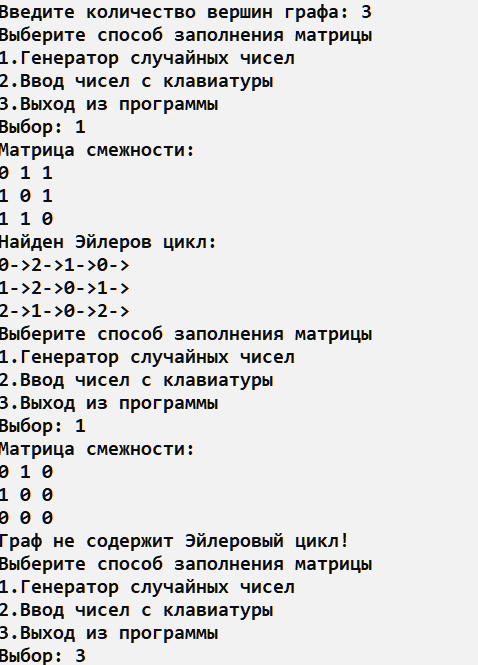


Рисунок 17 – возвращение в меню

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод текстового меню с выбором режима генерации матрицы смежности графа | верно |
| Выбор генерации матрицы | Вывод сообщения о выборе количества вершин в графе | верно |
| Выбор количества вершин в графе | Вывод матрицы, если был выбран автоматический метод генерации либо заполнение и вывод матрицы, если был выбран ручной метод генерации | верно |
| Проверка графа на наличие цикла Эйлера | Если цикл не найден вывод в консоль сообщения об этом.  Если цикл найден, вывод цикла | верно |

В результате тестирования было выявлено, что программа успешно проверяет данные на соответствие необходимым требованиям.

# Ручной расчёт задачи

Пример работы программы для графа с 5 вершинами (Рисунок 18)

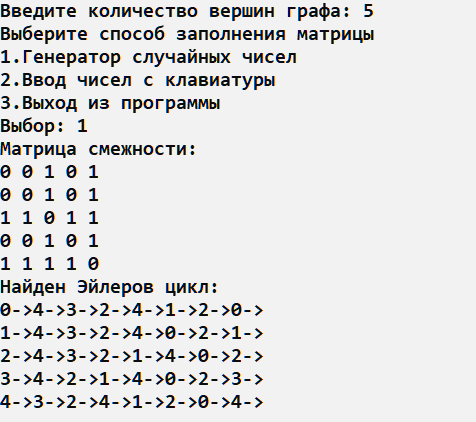


Рисунок 18 – выполнение для графа из 5 вершин

Воссоздадим граф по заданной матрице смежности (Рисунок 19)

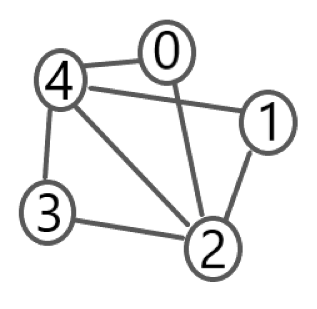


Рисунок 19 – граф из 5 вершин

Пройдемся по всем стартовым вершинам.

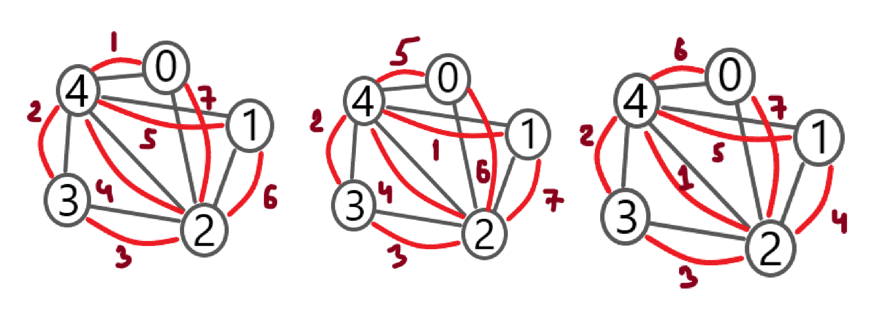


Рисунок 20 - 0 вершина Рисунок 21 – 1 вершина Рисунок 22 – 2 вершина

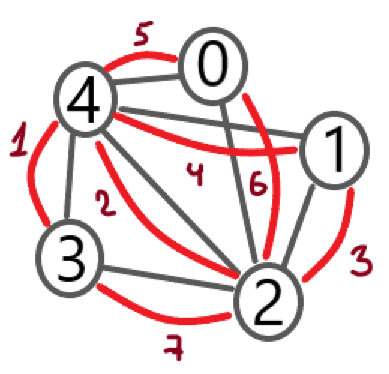
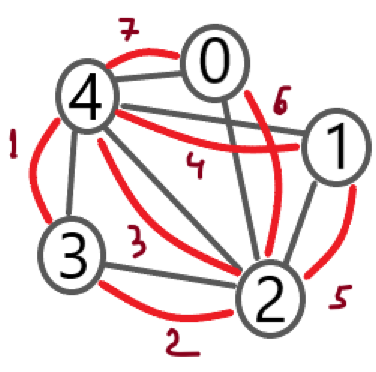


Рисунок 23 – 3 вершина Рисунок 24 – 4 вершина

Ручной расчёт совпал с программным, следовательно, программа работает верно.

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая нахождение цикла Эйлера в MicrosoftVisualStudio 2019. При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей, основанных на теории графов. 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**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <vector>

#include <stack>

#include <iostream>

using namespace std;

FILE\* f;

stack<int> st;

vector<int> res;

bool flag;

void DFS(int start, int n, int\*\* G) {

st.push(start);

while (!st.empty())

{

int v = st.top();

int i;

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

if (G[v][i]) // если есть связь прерываем цикл

break;

if (i == n) // когда пройдет весь цикл,

{

res.push\_back(v); // вставляем элемент в конец вектора посещений

st.pop(); // удаляет элемент из начала стека

}

else

{

--G[v][i];

--G[i][v];

st.push(i);

}

}

}

int eiler(int n, int\*\* G, int vert) {

vector<int> deg(n); // динамический массив данных

for (int i = 0; i < n; ++i) // цикл прохода по матрице

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

deg[i] += G[i][j]; // в ячейку массива записывается кол-во связей одной вершины

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

if ((deg[i] > 0) && (deg[i] % 2 == 0))

flag = true;

else

{

flag = false;

printf("Граф не содержит Эйлеровый цикл!\n");

fprintf(f, "Граф не содержит Эйлеровый цикл!\n");

return 0;

}

printf("Найден Эйлеров цикл:\n");

fprintf(f, "Найден Эйлеров цикл:\n");

int\*\* H = new int\* [n];

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

H[i] = new int[n];

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

{

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

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

H[i][j] = G[i][j];

DFS(k, n, &(&H)[0][0]);

for (size\_t j = 0; j < res.size();++j)

{

printf("%d->", res[j]);

fprintf(f, "%d->", res[j]);

}

printf("\n");

fprintf(f, "\n");

res.clear();

}

return 0;

}

int main() {

int how = -1;

int n = 0;

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

while (n <= 1)

{

printf("Введите количество вершин графа: ");

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

if (n <= 1)

printf("Задано слишком мало вершин!\n");

}

int\*\* G = new int\* [n];

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

{

G[i] = new int[n];

}

while (how != 1 and how != 2)

{

menu:

printf("Выберите способ заполнения матрицы\n1.Генератор случайных чисел\n2.Ввод чисел с клавиатуры\n3.Выход из программы\nВыбор: ");

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

switch (how)

{

case(1):

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

{

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

{

if (i == j)

{

G[i][j] = 0;

}

else if (i != j)

{

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

G[j][i] = G[i][j];

}

}

}

break;

case(2):

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

{

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

{

while (G[i][j] != 0 and G[i][j] != 1)

{

printf("\nG[%d][%d] = ", i, j);

scanf\_s("%d", &G[i][j]);

}

}

}

break;

case(3):

return 0;

default:

printf("Выбран неверный вариант!\n");

}

}

f = fopen("база", "w+");

if (how == 1 || how == 2)

{

printf("Матрица смежности:\n");

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

{

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

{

printf("%d ", G[i][j]);

fprintf(f, "%d ", G[i][j]);

}

printf("\n");

fprintf(f, "\n");

}

}

do

{

eiler(n, &(&G)[0][0], 0);

goto menu;

} while (flag != true);

free(G);

fclose(f);

}