

Министерство науки и высшего 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 Эйлеровых циклов»

28.12.22
обсуждено
ФВ

Выполнил:
студент группы 21ВВ2
Сорокина Елена

Приняли:
д.т.н Митрохин М.А.
к.т.н Юрова О. В.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

« 13 » сентября 20 22

ЗАДАНИЕ

на курсовое проектирование по курсу

Методы и основы алгоритмизации в инженерных задачах
Студенту Сережко Ели Андрей Группа 21BB2
Тема проекта Реализация алгоритма нахождения
Тейлоровых рядов

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данными заданием курсового проекта.

Техническое задание должно содержать:

1. Постановку задачи;
2. Требуемую часть задания;
3. Описание алгоритма поставленной задачи;
4. Пример ручного расчета задачи и/или вычисления (на небольшом участке работы алгоритма);
5. Описание самой программы;
6. Тесты;
7. Введен литературы;
8. Результаты работы программы.

Объем работы по курсу

1. Расчетная часть

Ручной расчет работы алгоритма

2. Графическая часть

Схема алгоритма в формате блок-схем

3. Экспериментальная часть

Тестирование программы;
Результаты работы программы на
тестовых данных.

Срок выполнения проекта по разделам

- 1 Исследование теоретической части курсового
- 2 Разработка алгоритмов программы
- 3 Разработка программы
- 4 Тестирование и завершение разработки программы
- 5 Проверка правильности работы
- 6
- 7
- 8

Дата выдачи задания "13" сентября 2022 г.

Дата защиты проекта " " "

Руководитель

Задание получил

"13" сентября

2022 г.

Студент

Сорокина Елена Андреевна (сорок)

Содержание

| | |
|--------------------------------|-----------|
| Реферат | 3 |
| Введение | 4 |
| 1 Постановка задачи | 5 |
| 2 Теоретическая часть задания | 6 |
| 3 Описание алгоритма программы | 9 |
| 4 Описание программы | 11 |
| 5 Тестирование | 18 |
| 6 Ручной просчет задачи | 23 |
| Заключение | 25 |
| Список литературы | 26 |
| Листинг программы | 27 |
| Файл Header.h | 27 |
| Файл menu.cpp | 28 |
| Файл source.cpp | 30 |
| Файл main.cpp | 34 |

Реферат

Отчет 39 стр, 17 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, ЭЙЛЕРОВ ПУТЬ, ЭЙЛЕРОВ ЦИКЛ, ПОИСК В ГЛУБИНУ.

Цель исследования - разработка программы, способная выявлять в графе эйлеровы циклы.

В работе рассмотрены правила поиска в глубину, с помощью которого находятся эйлеровы пути и циклы.

Введение

Темой моего курсового проекта является алгоритм поиска эйлерова цикла в графе.

Рассматриваемая задача является одной из самых старейших в теории графов, она также имеет реальное историческое начало.

В городе Кенигсберге (ныне Калининград) имелось семь мостов, соединяющих два берега реки Преголь, и два острова (рис.1). Требовалось, начав путешествие из одной точки города пройти по всем мостам по одному разу и вернуться в исходную точку.

Если поставить в соответствие мостам ребра, а участкам суши — вершины, то получится граф, в котором надо найти простой цикл, проходящий через все ребра. В общем виде эта задача была решена Эйлером в 1736 г.

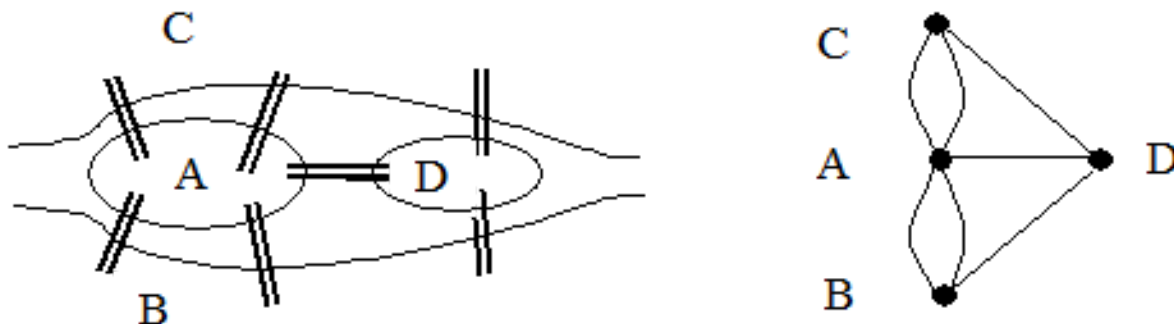


Рисунок 1 - Иллюстрация задачи

В качестве среды разработки мною была выбрана среда Microsoft Visual Studio 2022, язык программирования – Си/Си++.

Целью данной курсовой работы является разработка программы на языке Си/Си++.

1 Постановка задачи

Целью является разработать программу, которая сможет анализировать граф, заданный матрицей смежности, находить эйлеров цикл в графе и записывать результат в файл.

Опишу работу программы.

Пользователь вводит количество вершин матрицы смежности.

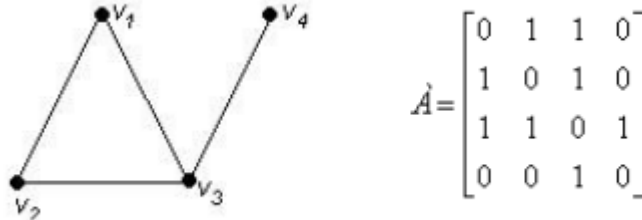
Далее программа предоставит возможность заполнения матрицы смежности автоматически и вручную. После обработки этих данных на экран должна выводиться матрица смежности.

Следующим шагом работы программы должна стать проверка графа на эйлеровость. Если граф удовлетворяет условиям, то должен быть построен эйлеров цикл и результаты работы записаны в файл. Иначе пользователю будет предложено заполнить матрицу смежности заново, завершить работу с программой или сделать матрицу эйлеровой.

Устройство ввода – клавиатура и мышь.

2 Теоретическая часть задания

Граф G (рисунок 2) задается множеством вершин V_1, V_2, \dots, V_n и множеством ребер, соединяющих между собой определенные вершины.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Рисунок 2 – Пример графа, заданного матрицей смежности

Определение 1. Эйлеровой цепью в неориентированном графе G называется простая цепь, содержащая все ребра графа G . Эйлеровым циклом называется замкнутая Эйлерова цепь. Аналогично, эйлеров путь в орграфе G — это простой путь, содержащий все дуги графа G . Эйлеров контур в орграфе G — это замкнутый эйлеров путь. Граф, в котором существует эйлеров цикл, называется эйлеровым.

Простой критерий существования эйлерова цикла в связном графе дается следующей теоремой.

Теорема 1. (Эйлер) Эйлеров цикл в связном неориентированном графе $G(X, E)$ существует только тогда, когда все его вершины имеют четную степень.

Доказательство. Необходимость. Пусть m - эйлеров цикл в связном графе G , x — произвольная вершина этого графа. Через вершину x эйлеров цикл проходит некоторое количество k ($k \geq 1$) раз, причем каждое прохождение, очевидно, включает два ребра, и степень этой вершины равна $2k$, т.е. четна, так как x выбрана произвольно, то все вершины в графе G имеют четную степень.

Достаточность. Воспользуемся индукцией по числу m ребер графа. Эйлеровы циклы для обычных (не псевдо) графов можно построить начиная с $m=3$. Легко проверить, что единственный граф с $m=3$, имеющий все вершины с четными степенями, есть граф K_3 (рис. 2). Существование эйлерова цикла в нем очевидно. Таким образом, для $m=3$ достаточность условий доказываемой теоремы имеет место. Пусть теперь граф G имеет $m > 3$ ребер, и пусть утверждение справедливо для всех связных графов, имеющих меньше, чем m ребер.

Зафиксируем произвольную вершину a графа G и будем искать простой цикл, идущий из a в a . Пусть $m(a, x)$ — простая цепь, идущая из a в некоторую вершину x . Если $x \neq a$, то цепь m можно продолжить из вершины x в некотором направлении. Через некоторое число таких продолжений мы придем в вершину $z \in X$, из которой нельзя продлить полученную простую цепь. Легко видеть, что $z = a$ так как из всех остальных вершин цепь может выйти (четные степени!); а в a она начиналась. Таким образом, нами построен цикл m , идущий из a в a .

Предположим, что построенный простой цикл не содержит всех ребер графа G . Удалим ребра, входящие в цикл m , из графа G и рассмотрим полученный граф $G'(X, E')$. В графе G' все вершины имеют четные степени. Пусть G'_1, G'_2, \dots, G'_k — компоненты связности графа G' , содержащие хотя бы по одному ребру.

Согласно предположению индукции все эти компоненты обладают эйлеровыми циклами m_1, m_1, \dots, m_k соответственно. Так как граф G связан, то цепь m встречает каждую из компонент G'_1, G'_2, \dots, G'_k . Пусть первые встречи цикла m с компонентами G'_1, G'_2, \dots, G'_k происходят соответственно в вершинах x_1, x_2, \dots, x_k . Тогда простая цепь $n(a, a) = m(a, x_1) \cup m_1(x_1, x_1) \cup m(x_1, x_2) \cup \dots \cup m_k(x_k, x_k) \cup m(x_k, a)$ является эйлеровым циклом в графе G . Теорема доказана.

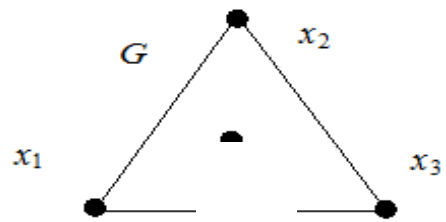


Рис. 2

Рисунок 3 – Пример графа, заданного матрицей смежности

3 Описание алгоритма программы

Для программной реализации алгоритма понадобится три массива: $G(int^{**})$ - для изменения матрицы во время обхода, $Gtemp(int^{**})$ - для хранения матрицы с эйлеровым циклом, $GG(int^{**})$ - для хранения исходной матрицы, $LIFO(int^*)$ - для хранения обхода матрицы, $C(int^{**})$ - для хранения эйлерового цикла.

Создается матрица смежности G размерности $n*n$ и записывается в файл. Запускается функция `input_auto_euler()`, которая проходит по каждому элементу и меняет матрицу G , если она не может содержать эйлеровых циклов.

Если матрица G содержит эйлеровый цикл, то инициализируются переменные `pos` - для обозначения позиции, которая занеслась в эйлеров цикл, `k=1` - для обозначения начала обхода запускается функция `euler()`. В качестве стартовой вершины обхода выбирается переменная `start` и заносим её в массив `LIFO`: `LIFO[start] = 0`, а также отмечаем начало обхода переменной `k`. Далее находим связанные с ней вершины. Если такая находится, заносим её в массив `LIFO` и отмечаем в матрице G , что ребро, соединяющее эти две вершины, пройдено, а также поднимаем флаг `p`, что ребро посещено. Если же непройденных вершин больше не остается, сохраняем результат обхода в матрицу C и выводится на экран.

Ниже представлен псевдокод некоторых функций: `euler()`.

`euler()`:

1. пока $k \neq 0$

2. флаг $p = 0$

3. для $i=0$ пока $i < n$ делать $i=i+1$

4. если начальная вершина соединена со следующей, делать
 $p = 1$

5. конец условия

6. если $p \neq 0$

7. занести в массив LIFO вершину

8. пометить ребро как прямое

9. сдвигаем позицию стека на 1 вперед ($k=k+1$)

10. иначе

11. заносим в массив C вершины в обратном порядке

12. $pos = pos + 1$

13. $k = k - 1$

4 Описание программы

Для написания данной программы использован язык программирования Си/Си++.

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

Данная программа является многомодульной, поскольку состоит из нескольких функций: `main`, `output`, `input`, `euler` и функций вывода меню.

Разработанная мною программа состоит из 3 модулей:

1. `main.cpp`
2. `menu.cpp`
3. `source.cpp`

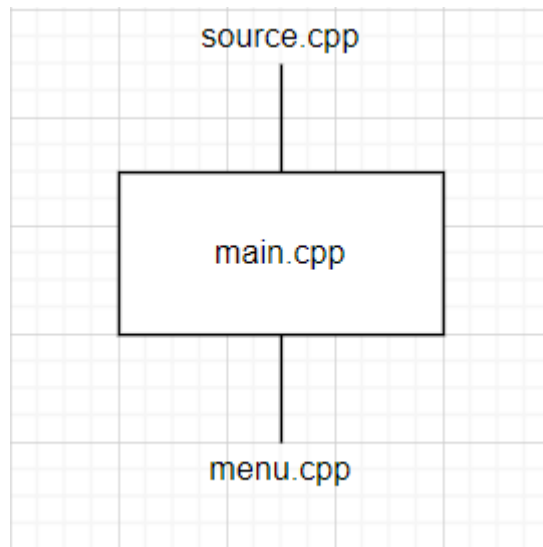


Рисунок 4 – Связь модулей программы

Файл `menu.cpp` содержит функции которые выводят текст на экран консоли.

Файл `main.cpp` содержит функцию `Main` в которой содержится цикл, вызывающий все остальные функции по требованию пользователя.

Файл source.cpp содержит следующие функции:

1. Вывод матрицы смежности (output)
2. Заполнение матрицы смежности вручную (input)
3. Заполнение матрицы смежности автоматически (input_auto)
4. Поиск эйлерова цикла в графе (euler)

Опишу работу функции euler, которая и выполняет задачу моего курсового проекта.

Для ее работы мне были необходимы следующие переменные:

i, j – необходимы для обращения к элементам матрицы смежности;

start – необходима для сохранения вершины с которой начнется обход;

pos – счетчик для занесения вершин в стек;

k – позиция вершины стека;

Массив LIFO – стек (Last In First Out);

Массив C – необходим для сохранения пути эйлерова цикла;

Работа функции начинается с занесения в стек стартовой вершины и установки вершины стека в переменной k:

```
LIFO[0] = start;  
k = 1;
```

Далее находим вершину с минимальным номером и смежную с вершиной, номер которой в стеке. Заносим вершину в стек и помечаем ребро как прямое. Позицию стека двигаем на 1 вверх иначе заносим в стек обратное ребро:

```
while (k != 0)  
{  
    p = 0;  
    for (i = 0; i < n; i++)
```

```

        if (g[LIFO[k - 1]][i] == 1)
        {
            p = 1;
            break;
        }
    if (p != 0)
    {
        LIFO[k] = i;

        g[LIFO[k - 1]][i] = 2;
        g[i][LIFO[k - 1]] = 2;
        k++;
    }
    else
    {
        C[0][pos] = LIFO[k - 1];
        C[1][pos] = LIFO[k - 2];
        pos++;
        k--;
    }
}

```

Когда работа алгоритма закончена происходит вывод эйлерова цикла на экран и запись результата в файл.

```

for (i = 0; i < pos - 1; i++)
    cout << C[0][i] << " " << C[1][i] << endl;
FILE << "Эйлеров цикл:" << endl;

for (i = 0; i < pos - 1; i++) {
    FILE << C[0][i] << " " << C[1][i] << endl;
}
FILE.close();

```

Работа программы начинается с запроса ввести порядок матрицы смежности. Это необходимо для динамического выделения памяти.

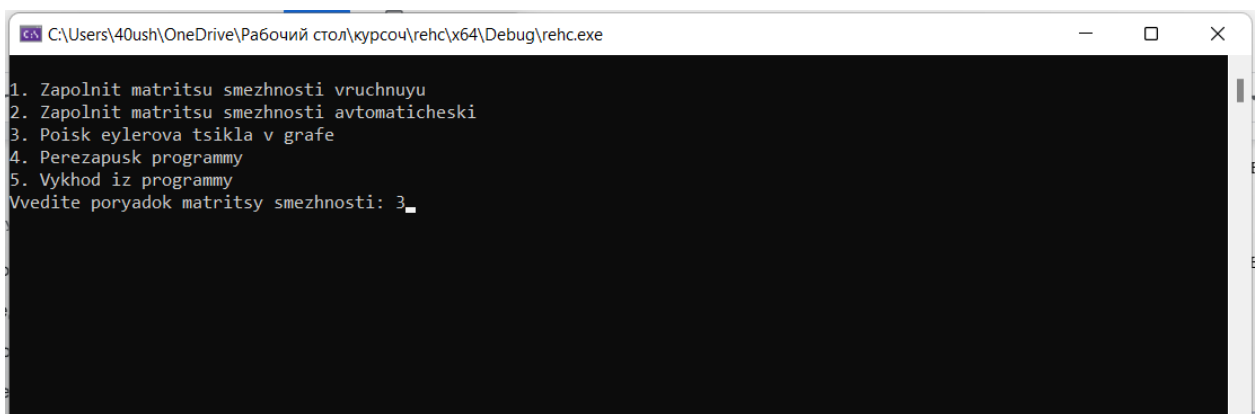
Далее выведено главное меню и пользователь сможет продолжить взаимодействовать с программой, а именно выбрать способ задания матрицы

смежности (вручную или автоматически), выполнить поиск эйлерова цикла, перезапустить программу и завершить работу с ней.

Если граф не содержит в себе искомый цикл, то пользователю будет предложено заполнить матрицу смежности заново, завершить работу программы или сделать матрицу эйлеровой.

Если цикл содержится в графе, то он будет выведен на экран и записан в файл под названием “Euler_cycle.txt”

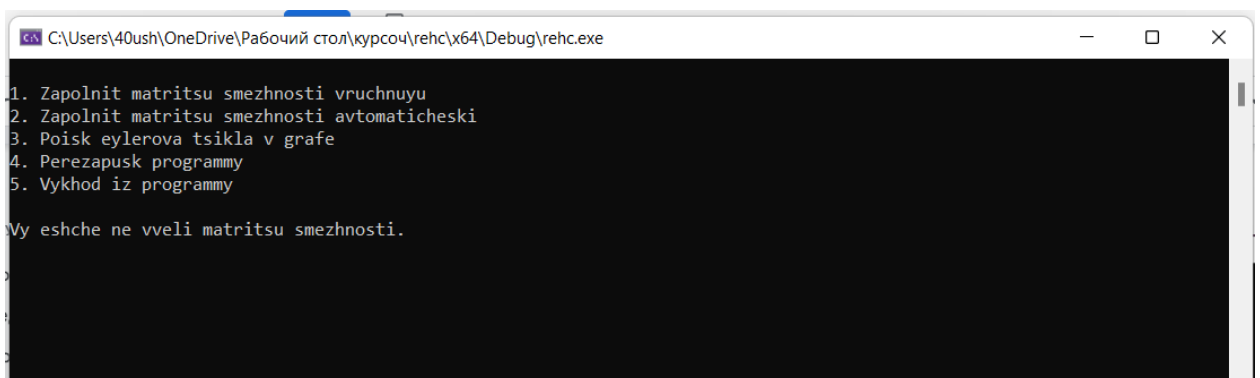
Ниже можно увидеть пример стандартной работы программы.



```
C:\Users\40ush\OneDrive\Рабочий стол\курсоч\rehc\х64\Debug\rehc.exe
1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomatichieski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy
Vvedite poriyadok matritsy smezhnosti: 3_
```

Рисунок 5 – Пользователь вводит порядок для матрицы смежности

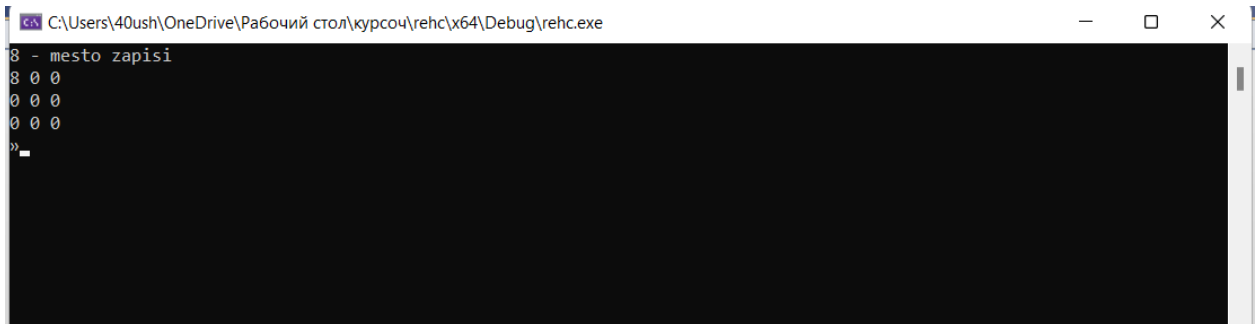
После происходит переход в главное меню.



```
C:\Users\40ush\OneDrive\Рабочий стол\курсоч\rehc\х64\Debug\rehc.exe
1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomatichieski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy
Vy eshche ne vveli matritsu smezhnosti.
```

Рисунок 6 – Выбор пункта меню

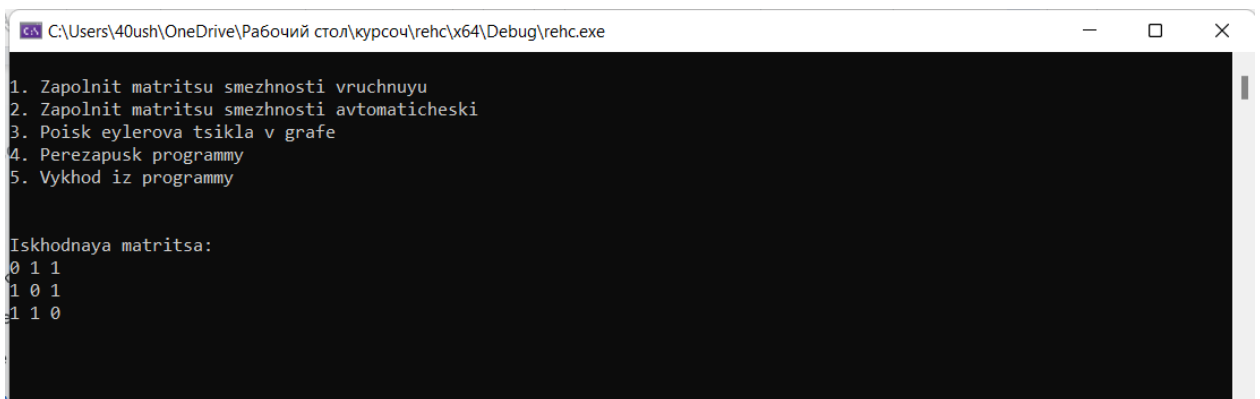
Выбираем первый пункт меню, заполняем матрицу вручную.



```
C:\Users\40ush\OneDrive\Рабочий стол\курсоч\rehc\х64\Debug\rehc.exe
8 - mesto zapisi
8 0 0
0 0 0
0 0 0
»
```

Рисунок 7 – Заполнение матрицы вручную

Возвращаемся в главное меню.



```
C:\Users\40ush\OneDrive\Рабочий стол\курсоч\rehc\х64\Debug\rehc.exe
1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomaticheskii
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy

Iskhodnaya matritsa:
0 1 1
1 0 1
1 1 0
```

Рисунок 8 – Результат ввода вручную

Выбираем пункт под номером 3 «Поиск эйлера цикла в графе».

```
C:\Users\40ush\OneDrive\Рабочий стол\хуроч\rehc\x64\Debug\rehc.exe

1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomaticheski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vыход iz programmy

Iskhodnaya matritsa:
0 1 1
1 0 1
1 1 0

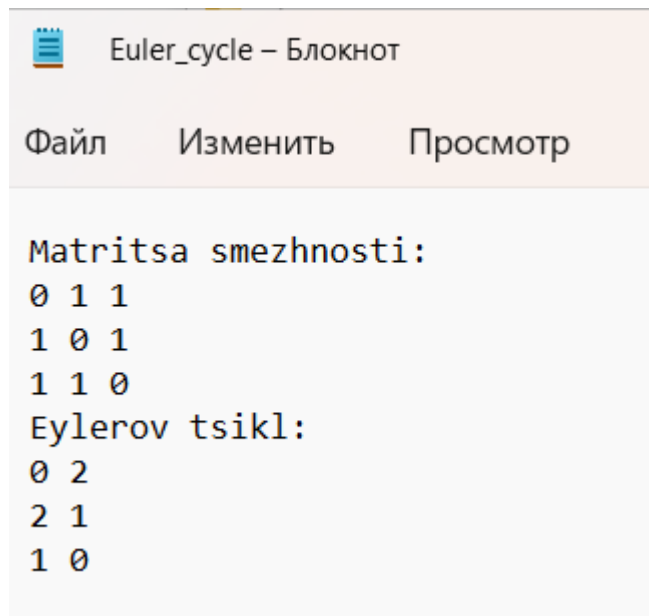
3

Eylerovyy tsikl:
0 2
2 1
1 0
Press any key to continue . . .
```

Рисунок 9 – Результат поиска эйлера цикла

| | | | | | |
|--------|----------------------|---|------------------|-----------------------|------|
| Folder | x64 | ✓ | 27.12.2022 23:52 | Папка с файлами | |
| File | Euler_cycle | ↻ | 28.12.2022 11:26 | Текстовый докум... | 1 КБ |
| File | Header.h | ✓ | 28.12.2022 1:42 | C/C++ Header | 1 КБ |
| File | main.cpp | ✓ | 28.12.2022 10:21 | C++ Source | 4 КБ |
| File | menu.cpp | ✓ | 28.12.2022 2:09 | C++ Source | 2 КБ |
| File | rehc.vcxproj | ✓ | 27.12.2022 23:52 | VC++ Project | 7 КБ |
| File | rehc.vcxproj.filters | ✓ | 27.12.2022 23:52 | VC++ Project Filte... | 2 КБ |
| File | rehc.vcxproj.user | ✓ | 27.12.2022 23:49 | Per-User Project O... | 1 КБ |
| File | source.cpp | ✓ | 28.12.2022 8:41 | C++ Source | 3 КБ |

Рисунок 10 – Файл создан



The image shows a Notepad window with the title "Euler_cycle – Блокнот". The menu bar contains "Файл", "Изменить", and "Просмотр". The text content is as follows:

```
Matritsa smezhnosti:  
0 1 1  
1 0 1  
1 1 0  
Eulerov tsikl:  
0 2  
2 1  
1 0
```

Рисунок 11 – Содержание файла

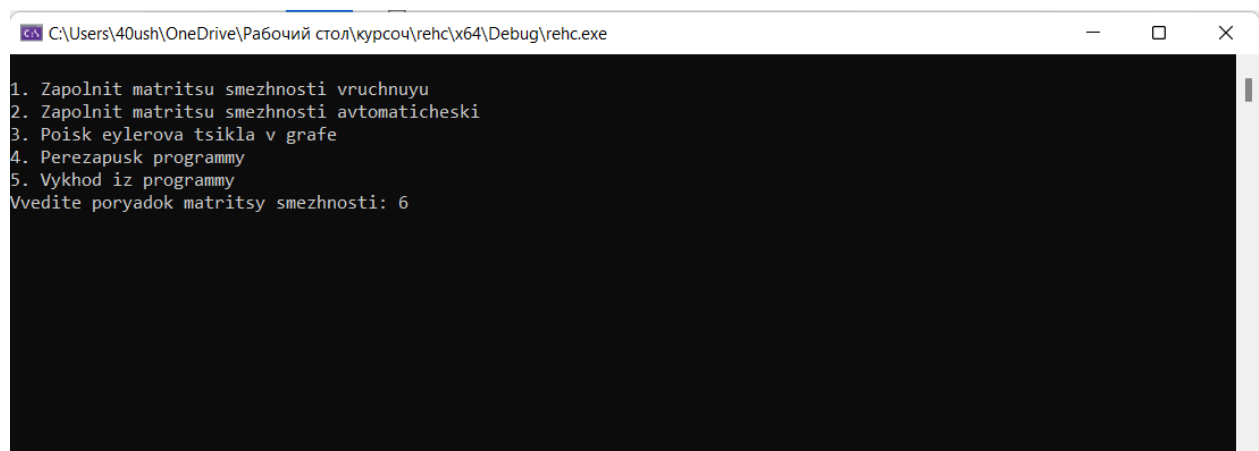
5 Тестирование

Среда разработки Microsoft Visual Studio 2022 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, связанных с вводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже приведено тестирование для случайно сгенерированной матрицы смежности.

Снова вводим порядок матрицы.



```
C:\Users\40ush\OneDrive\Рабочий стол\курсов\rehc\x64\Debug\rehc.exe
1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomaticheski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy
Vvedite poryadok matritsy smezhnosti: 6
```

Рисунок 12 – Пользователь вводит порядок для матрицы смежности

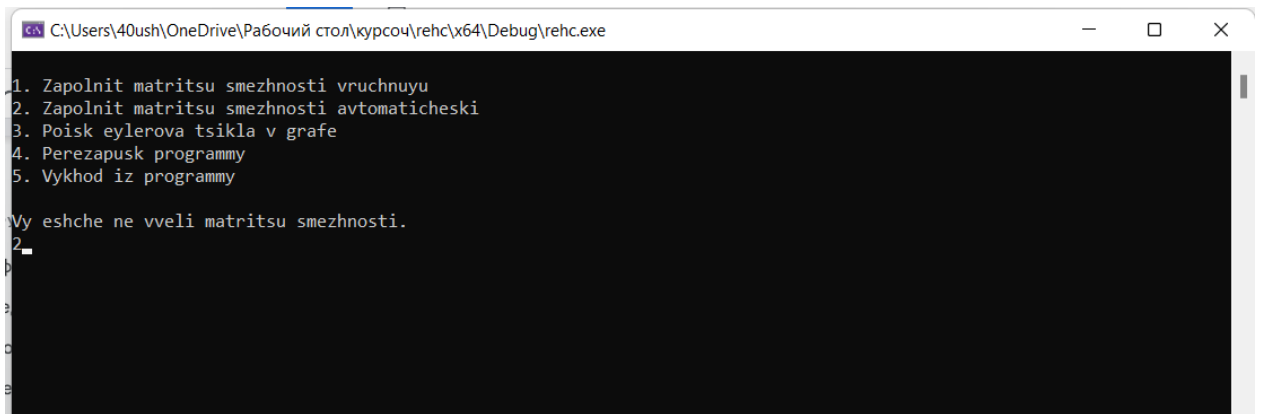


Рисунок 13 – Пользователь выбирает автоматическое заполнение

Далее снова попадаем в главное меню и видим результат генерации

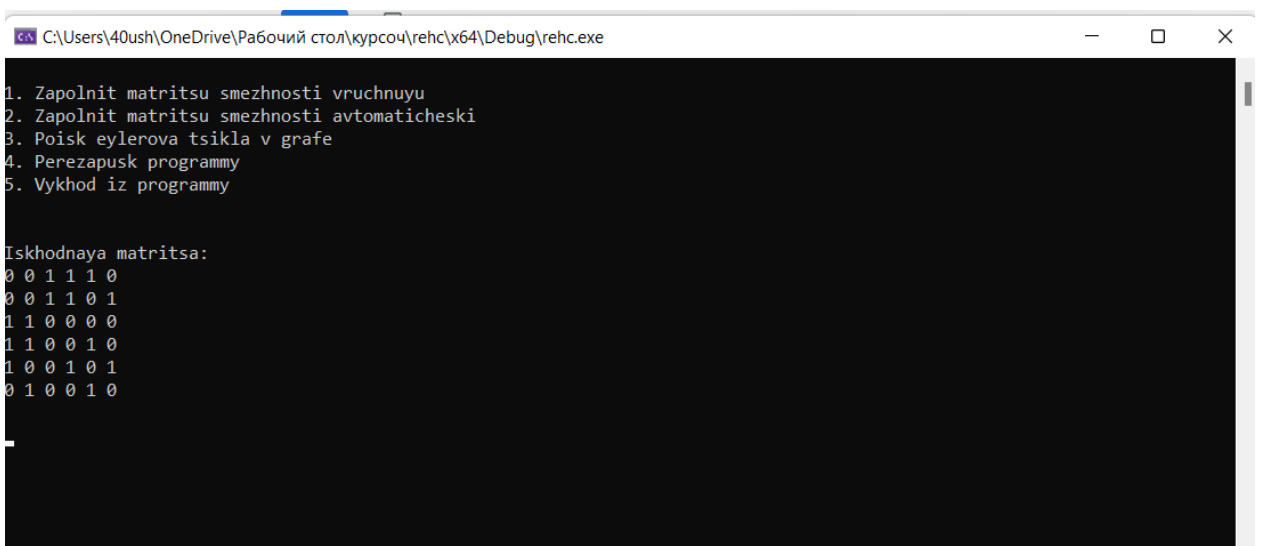


Рисунок 14 – Результат автоматической генерации

Сгенерированная матрица, очевидно, не содержит эйлерова цикла, программа сообщает об этом и предоставляет пользователю выбор: сгенерировать матрицу заново, закрыть программу или сделать матрицу эйлеровой.

```
C:\Users\40ush\OneDrive\Рабочий стол\кыроч\rehc\x64\Debug\rehc.exe

1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomatichieski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy

Iskhodnaya matritsa:
0 0 1 1 1 0
0 0 1 1 0 1
1 1 0 0 0 0
1 1 0 0 1 0
1 0 0 1 0 1
0 1 0 0 1 0

3
Graf ne sodержit Eylerov tsikl. Zapolnit matritsu smezhnosti zanovo?
1. Da
2. Net. zavershit rabotu programmy
3. Net. sdelat matritsu eylerovoy!
>
```

Рисунок 15 – Результат поиска эйлерова цикла

Выбираем «Нет, сделать матрицу эйлеровой!» и программа выводит нам исходную и доработанную матрицу с эйлеровым циклом.

```
C:\Users\40ush\OneDrive\Рабочий стол\кыроч\rehc\x64\Debug\rehc.exe

1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomatichieski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vykhod iz programmy

Iskhodnaya matritsa:
0 0 1 1 1 0
0 0 1 1 0 1
1 1 0 0 0 0
1 1 0 0 1 0
1 0 0 1 0 1
0 1 0 0 1 0

Matritsa s eylerovym tsiklom:
0 0 1 1 1 1
0 0 1 1 1 1
1 1 0 0 0 0
1 1 0 0 1 1
1 1 0 1 0 1
1 1 0 1 1 0
```

Рисунок 16 – Исходная матрица и доработанная

Далее выбираем «Поиск эйлерова цикла в графе» уже в доработанной матрице

```
C:\Users\40ush\OneDrive\Рабочий стол\хуроч\rehc\x64\Debug\rehc.exe
0 0 1 1 0 1
1 1 0 0 0 0
1 1 0 0 1 0
1 0 0 1 0 1
0 1 0 0 1 0

Matritsa s eylerovym tsiklom:
0 0 1 1 1 1
0 0 1 1 1 1
1 1 0 0 0 0
1 1 0 0 1 1
1 1 0 1 0 1
1 1 0 1 1 0

3

Eylerovyy tsikl:
0 5
5 4
4 3
3 5
5 1
1 4
4 0
0 3
3 1
1 2
2 0

Press any key to continue . . .
```

Рисунок 17 – Результат поиска эйлеровых циклов в доработанной матрице

Таблица 1 - Описание поведения программы при тестировании

| Описание теста | Ожидаемый результат | Полученный результат |
|--|--|----------------------|
| Запуск программы | Вывод меню вывод размерности матрицы | Верно |
| Выбор самостоятельного ввода матрицы | Вывод сообщения о разрешенном вводе матрицы | Верно |
| Выбор автоматической генерации матрицы | Вывод сгенерированной матрицы | Верно |
| Выбор нахождения эйлерового цикла | Вывод эйлерового цикла или сообщения о том, что в матрице невозможно его существование | Верно |
| Выбор изменения матрицы | Вывод измененной матрицы | Верно |

В результате тестирования было выявлено, что программа корректно анализирует входные данные и выводит ожидаемо верные результаты.

6 Ручной просчет задачи

Проведем проверку программы посредством ручного просчета на примере графа с 4 вершинами. (Рисунок 7). Для наглядности была создана модель.

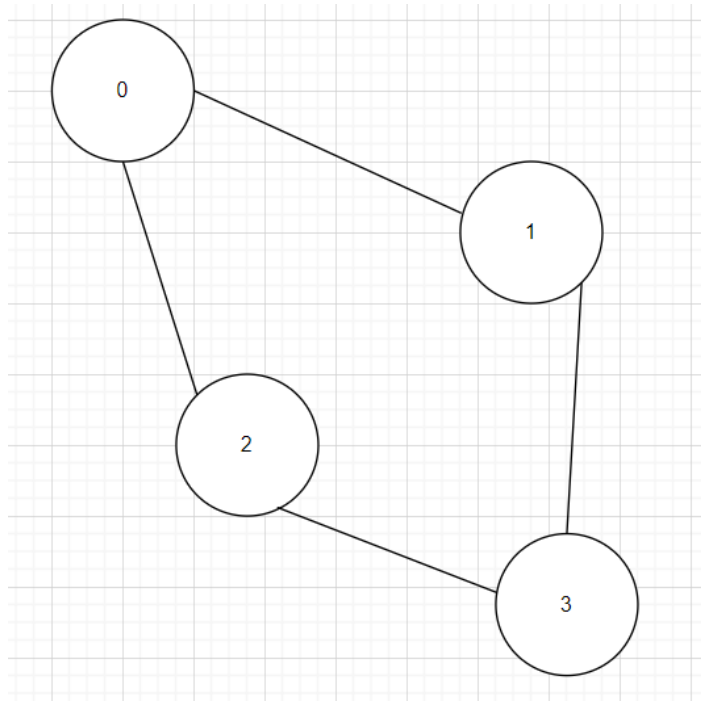


Рисунок 18 – Граф с рисунка 9

Проанализируем количество ребер у каждой вершины, их количество 2. Количество четное, следовательно, граф содержит эйлеров цикл.

За стартовую вершину возьмем вершину под номером 0, и пойдем против часовой стрелки. Эйлеров цикл для данного графа будет следующий:

Из вершины 0 в вершину 2;

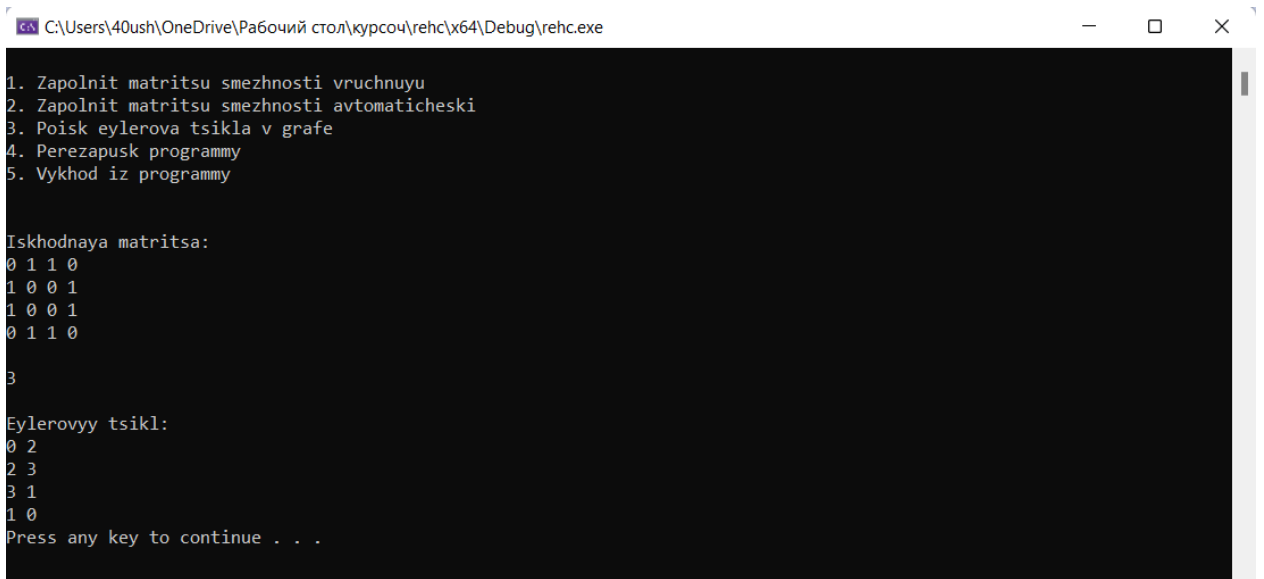
Из вершины 2 в вершину 3;

Из вершины 3 в вершину 1;

Из вершины 1 в вершину 0;

Цикл верный, и он замкнулся.

Можно сделать вывод что программа работает корректно.



```
C:\Users\40ush\OneDrive\Рабочий стол\хурпоч\rehc\x64\Debug\rehc.exe

1. Zapolnit matritsu smezhnosti vruchnuyu
2. Zapolnit matritsu smezhnosti avtomaticheski
3. Poisk eylerova tsikla v grafe
4. Perezapusk programmy
5. Vыход iz programmy

Iskhodnaya matritsa:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0

3

Eylerovyy tsikl:
0 2
2 3
3 1
1 0
Press any key to continue . . .
```

Рисунок 19 – Результат поиска эйлера цикла

Заключение

Во время создания данного проекта была разработана программа реализующая алгоритм нахождения эйлера цикла в графе в Microsoft Visual Studio 2022.

При выполнении данной работы были получены навыки разработки многомодульных программ и освоены приемы создания матриц смежностей, основанных на теории орграфов. Приобретены навыки по осуществлению алгоритма поиска в глубину. Углублены знания языка программирования Си/Си++. Были улучшены навыки отладки больших работ в среде Microsoft Visual Studio.

Недостатком программы является примитивный интерфейс.

Список литературы

1. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978, 296 с.
2. Зыков А.А. Основы теории графов. - М.:Наука, 1987, 384 с.
3. Мельников О.И. Теория графов в занимательных задачах. Изд.3, испр. и доп. 2009. 232 с.
4. Динман М.И. С++. Освой на примерах. – СПб.: БХВ – Петербург, 2006 – 384 с.
5. Харари Ф. Теория графов / Пер.с англ. и предисл. В. П. Козырева. Под ред. Г. П. Гаврилова. Изд. 2-е. - М.: Едиториал УРСС, 2003. - 296 с.

Листинг программы

Файл Header.h

```
//menu.cpp
void main_menu();
void menu_euler_viv();
void menu3();
void menu_empty_matrix();
void menu_enter_matrix();
void menu_euler_true();
void mat_euler();
void mat();
void note();
//source.cpp
void output(int** p, int n);
int** input(int** p, int n);
int** input_auto(int** p, int n);
int** euler(int** g, int n);
int** input_auto_euler(int** p, int n);
#pragma once
```

Файл menu.cpp

```
#include <iostream>

using namespace::std;

void main_menu() //vivod menu
{
    cout << endl << "1. Zapolnit matritsu smezhnosti vruchnuyu" << endl << "2. Zapolnit matritsu smezhnosti avtomaticheskii" << endl << "3. Poisk eylerova tsikla v grafe" << endl << "4. Perezapusk programmy" << endl << "5. Vykhod iz programmy" << endl;
}

void menu3() //menu posle proverki grafa
{
    cout << "Graf ne sodержit Eylerov tsikl. Zapolnit matritsu smezhnosti zanovo? " << endl << "1. Da" << endl << "2. Net. zavershit rabotu programmy" << endl << "3. Net. sdelat matritsu eylerovoy!" << endl << "»";
}

void menu_empty_matrix() //matritsi net
{
    cout << "Vy eshche ne vveli matritsu smezhnosti." << endl;
}

void menu_enter_matrix() //vvod poryadka
{
    cout << "Vvedite poryadok matritsy smezhnosti: ";
}

void menu_euler_true() // vivod esli est eylerov tsikl
{
    cout << "Graf yavlyayetsya Eylerovym i sodержit eylerov tsikl!" << endl;
}

void menu_euler_viv() //vivod eylerova tsikla
{
    cout << endl << "Eylerovyy tsikl: ";
}

void mat_euler() // vivod izmenennoy matritsi
{

```

```
        cout << endl << "Matritsa s eylerovym tsiklom: ";
    }

    void mat() // vivod ishodnoy matritsi
    {
        cout << endl << "Iskhodnaya matritsa:";
    }

    void note() //vivod esli malenkaya razmernost
    {
        cout << endl << "Matritsa ne mozhet byt eylerovoy. esli poryadok menshe 3!" << endl <<
        "Vvedite poryadok zanovo: ";
    }
```

Файл source.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <conio.h>
#include <fstream>
#include <stdlib.h>
#include "Header.h"
using namespace::std;

void output(int** p, int n) //vivod marritsi
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cout << p[i][j] << " ";
        }
        cout << endl;
    }
}

int** input(int** g, int n) //zapolnenie vruchnuyu
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            g[i][j] = 0;
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            system("cls");
            g[i][j] = 8;
            cout << "8 - mesto zapisi" << endl;
        }
    }
}
```

```

        output(g, n);
        cout << "»";
        cin >> g[i][j];
        cout << endl;
    }
}
return g;
}

int** input_auto(int** g, int n) //avto zapolnenie
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
            {
                g[i][j] = 0;
            }
            else
            {
                g[i][j] = rand() % 2;
            }
            g[j][i] = g[i][j];
        }
    }
    return g;
}

int** input_auto_euler(int** g, int n) // zapolnenie matritsi s eylerovim tsiklom
{
    int two = 0;
    int sum = 0;
    for (int i = 0; i < n; i++)
    {
        sum = 0;
        for (int j = 0; j < n; j++)
        {
            if (g[i][j] == 1) sum++;
        }
    }
}

```

```

        if ((g[i][j] == 0) && (i != j)) two = j;
    }
    if (sum % 2 == 1)
    {
        g[i][two] = 1;
        g[two][i] = g[i][two];
    }
}
return g;
}

```

```

int** euler(int** g, int n) //algoritm poiska eylerova tsikla
{

```

```

    int i, j;
    int start, pos = 0, p, k, LIFO[100], C[2][100];
    start = 0;
    int temp = 0;
    int one = 0;

```

```

    ofstream FILE("Euler_cycle.txt");

```

```

    FILE << "Matritsa smezhnosti:" << endl;

```

```

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            FILE << g[i][j] << " ";
        }
        FILE << endl;
    }

```

```

    cout << endl;

```

```

    LIFO[0] = start; // vnosim v stek start vershinu
    k = 1; // positsiya vershini steka
    while (k != 0)
    {
        p = 0;

```

```

v steke // Nakhodim vershinu s minimalnym nomerom i smezhnuyu s vershinoy nomer kotoroy

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

    if (g[LIFO[k - 1]][i] == 1)
    {
        p = 1;
        break;
    }
if (p != 0)
{
    LIFO[k] = i; //zanosim vershinu v stek
    //pomechaem rebro kak proydennoe
    g[LIFO[k - 1]][i] = 2;
    g[i][LIFO[k - 1]] = 2;
    k++; // sdvig pozitsii steka
}
else
{
    // vinosim obratnoe rebro
    C[0][pos] = LIFO[k - 1];
    C[1][pos] = LIFO[k - 2];
    pos++;
    k--;
}
}

// vivod resultata
for (i = 0; i < pos - 1; i++)
    cout << C[0][i] << " " << C[1][i] << endl;
// zapis v fale
FILE << "Eylarov tsikl:" << endl;

for (i = 0; i < pos - 1; i++) {
    FILE << C[0][i] << " " << C[1][i] << endl;
}
FILE.close();
return 0;
}

```

Файл main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <conio.h>
#include <fstream>
#include <stdlib.h>
#include <malloc.h>
#include "Header.h"
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int n, i, j, z;

    while (true)
    {
        main_menu();
        menu_enter_matrix();
        cin >> n;
        while (n < 3)
        {
            note();
            cin >> n;
        }
        int check = 0;
        int** G, ** Gtemp, ** GG;
        G = (int**)malloc(n * sizeof(int*));
        for (i = 0; i < n; i++)
        {
            G[i] = (int*)malloc(n * sizeof(int));
        }

        Gtemp = (int**)malloc(n * sizeof(int*));
        for (i = 0; i < n; i++)
        {
            Gtemp[i] = (int*)malloc(n * sizeof(int));
        }
    }
}
```



```

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

while (true)
{
    int choose;
    system("cls");
    main_menu();
    if (check == 1)
    {
        cout << endl;
        mat();
        cout << endl;
        output(Gtemp, n);
        cout << endl;
    }

    if (check == 2)
    {
        cout << endl;
        mat();
        cout << endl;
        output(GG, n);
        cout << endl;
        mat_euler();
        cout << endl;
        output(Gtemp, n);
        cout << endl;
    }

    if (check < 1) {
        cout << endl;
        menu_empty_matrix();
    }
}

```

```

cin >> choose;

if (choose == 1) // rychnoe zapolnenie
{
    G = input(G, n);

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            Gtemp[i][j] = G[i][j];
            GG[i][j] = G[i][j];
        }
    }

    check = 1;
    system("pause");
    continue;
}

if (choose == 2) //avtomaticheskoe zapolnenie
{
    G = input_auto(G, n);

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            Gtemp[i][j] = G[i][j];
            GG[i][j] = G[i][j];
        }
    }

    check = 1;
    system("pause");
    continue;
}

if (choose == 3) // poisk eylerova tsikla

```

```
{
```

```
int start, pos = 0, p, k, choosen /*LIFO[100], C[2][100]*/;  
start = 0;  
int temp = 0;  
int one = 0;  
int z = 0;  
int* LIFO;  
int** C;  
LIFO = (int*)malloc((n + 1) * sizeof(int));  
C = (int**)malloc(n * sizeof(int*));  
for (i = 0; i < n; i++)  
{  
    C[i] = (int*)malloc(2 * sizeof(int));  
}  
for (i = 0; i < n; i++)  
{  
    for (j = 0; j < n; j++)  
        if (G[i][j] == 1) one++;  
    if (one % 2 == 1) {  
        z = 1;  
        menu3();  
        cin >> choosen;  
        if (choosen == 1)  
        {  
            system("pause");  
            temp = 1;  
            break;  
        }  
        else if (choosen == 2)  
        {  
            goto END;  
        }  
        else if (choosen == 3)  
        {  
            input_auto_euler(G, n);  
  
            for (i = 0; i < n; i++)  
            {
```

```

        for (j = 0; j < n; j++)
        {
            Gtemp[i][j] = G[i][j];
        }
    }

    z = 1;
    check = 2;
    break;
}

}

}
if (temp == 1)
{
    break;
}
if (z == 0)
{
    menu_euler_viv();
    euler(G, n);
}
system("pause");
continue;
free(LIFO);
free(C);
}

if (choose == 4) // perezapusk
{
    break;
}

if (choose == 5) // vihod
{
    goto END;
}
}
system("cls");
continue;

```

```
        free(G);
        free(Gtemp);
        free(GG);
    }
END:
    return 0;
}
```