Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма нахождения Эйлеровых циклов»

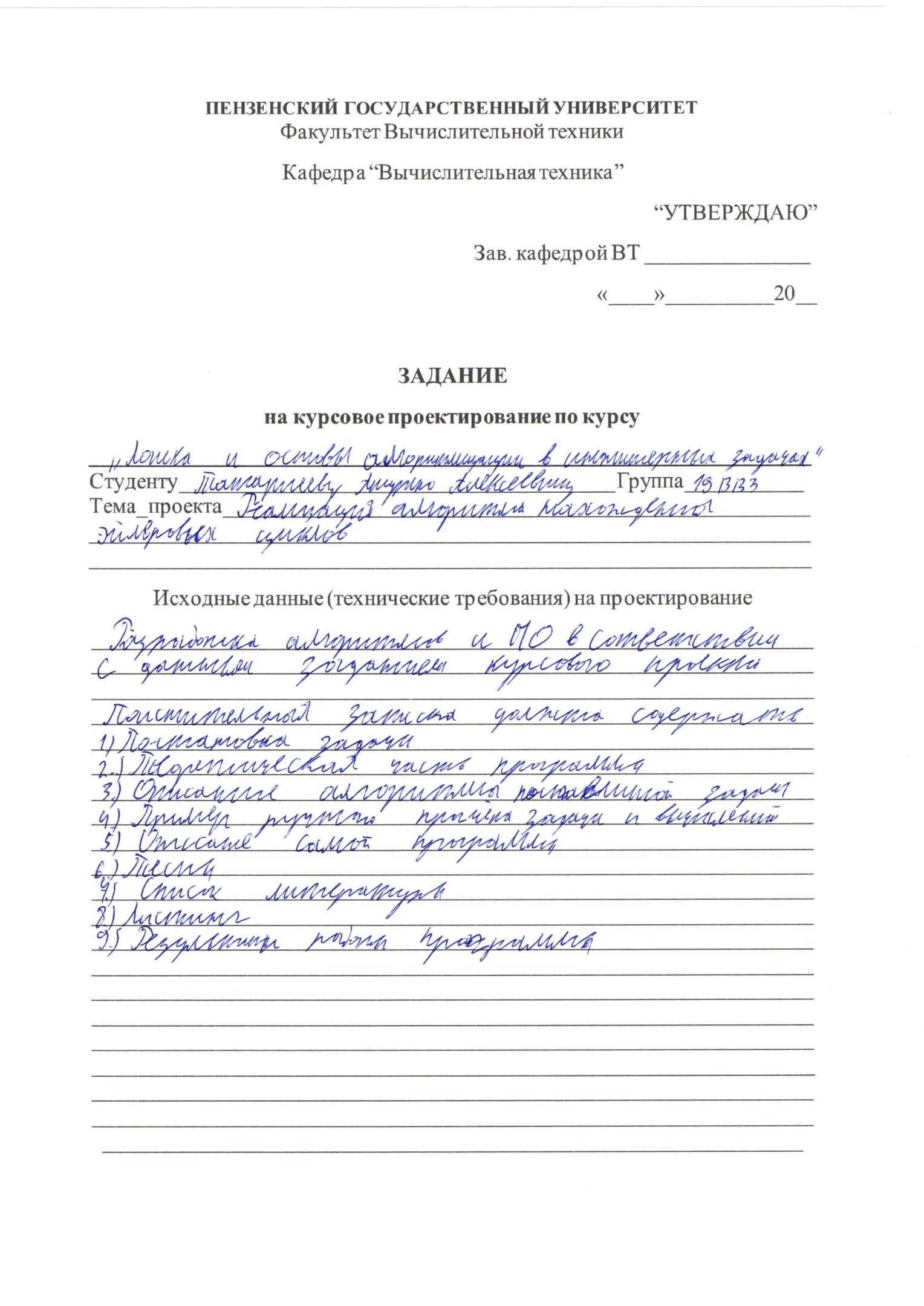
***Выполнил работу студент группы 19ВВ3:***

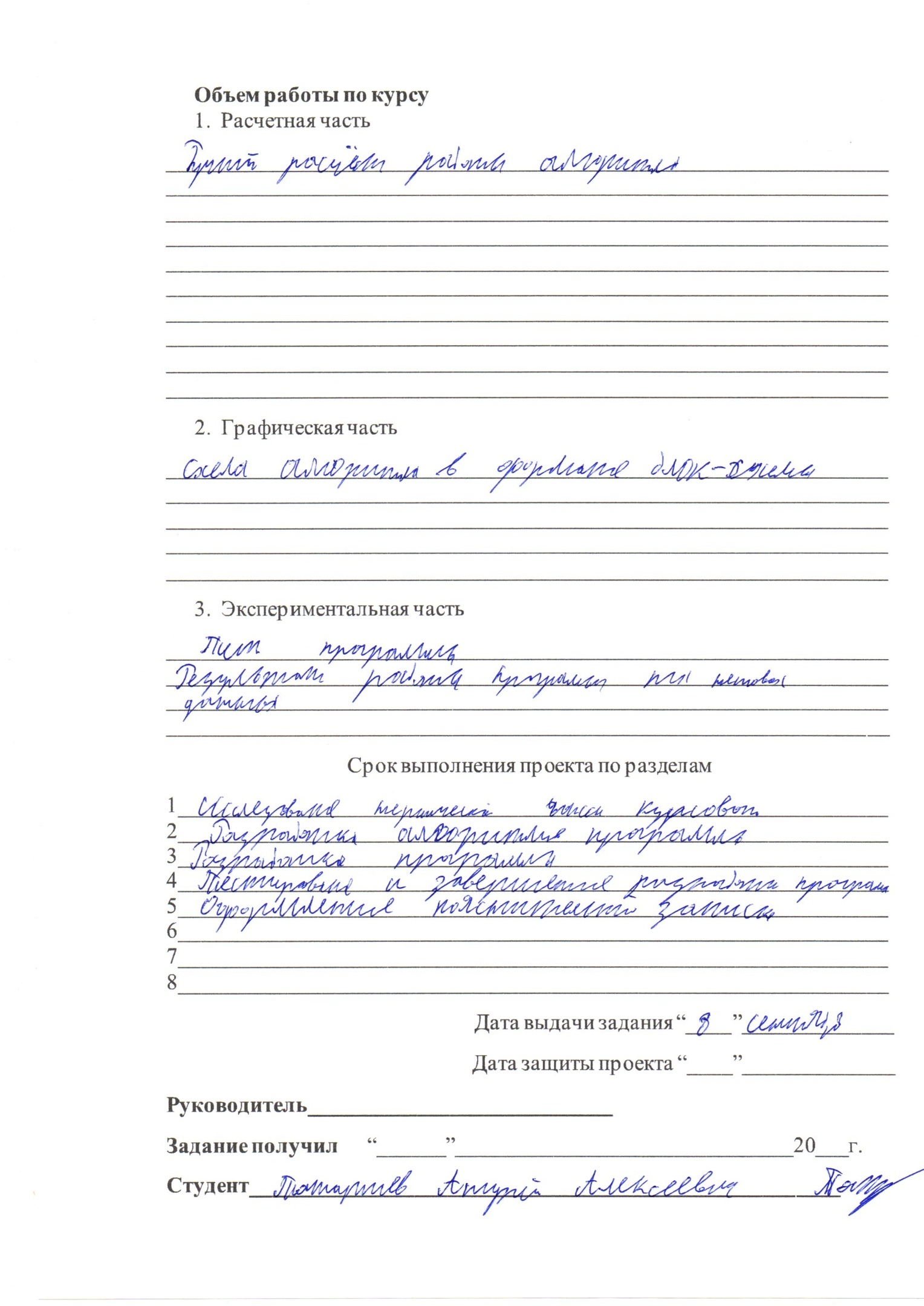
Татаршев А.А.

**Приняли:**

д.т.н. профессор Митрохин М. А.

Пенза 2020





**Содержание**

[Реферат 5](#_Реферат.)

[Введение 6](#_Введение)

1. [Постановка задачи 7](#_TOC_250007)
2. [Теоретическая часть задания 8](#_TOC_250006)
3. [Описание алгоритма программы 11](#_TOC_250005)
4. [Описание программы 1](#_Описание_алгоритма_программы)4
5. [Тестирование](#_TOC_250003) 21

[Заключение 32](#_TOC_250001)

[Список литературы 33](#_TOC_250000)

Листинг программы 34

# Реферат.

Отчет по курсовой работе: 35 страницы, 25 рисунков, 2 таблицы, 3 источника

Цель работы — создание программы для реализации нахождения Эйлеровых циклов

Ключевые слова: эйлеров цикл, эйлеров путь, программа, алгоритм, массив, матрица смежности

Требуется найти цикл, проходящий по каждой дуге ровно один раз. Эту задачу впервые поставил и решил Леонард Эйлер, чем и заложил основы теории графов, а соответствующие циклы теперь называются эйлеровыми. Фигуры, которые требуется обрисовать, не прерывая и не повторяя линии, также относятся к эйлеровым циклам.

# Введение

Задача возникла из предложенной Эйлеру головоломки, получившей название "проблема кенигсбергских мостов". Река Прегель, протекающая через Калининград (прежде город назывался Кенигсбергом), омывает два острова. Берега реки связаны с островами, как показано на рисунке 1

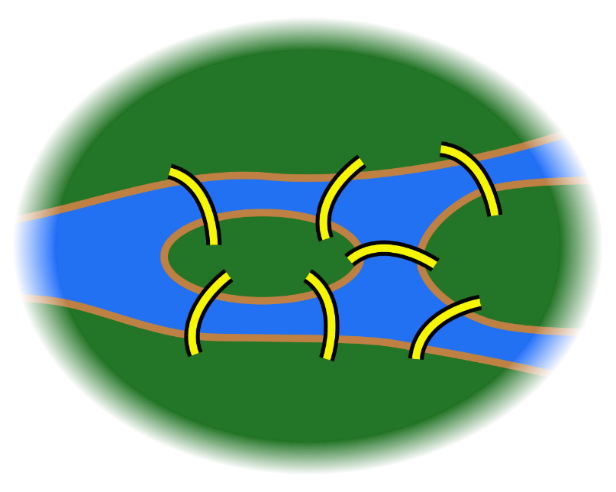


Рисунок 1- кенигсбергские мосты

В головоломке требовалось найти маршрут, проходящий по всем участкам суши таким образом, чтобы каждый из мостов был пройден ровно один раз, а начальный и конечный пункты маршрута совпадали.

# Алгоритм нахождения эйлерова цикла – это алгоритм нахождения пути в графе, проходящий через все его рёбра, являющейся циклом. Для лучшего понимания алгоритма в пример можно привести задачу о домино.

Имеется N домино, как известно, на двух концах домино записано по одному числу (обычно от 1 до 6, но в нашем случае не важно). Требуется выложить все домино в ряд так, чтобы у любых двух соседних домино числа, записанные на их общей стороне, совпадали. Домино разрешается переворачивать.

Переформулируем задачу. Пусть числа, записанные на домино, - вершины графа, а домино - рёбра этого графа (каждая домино с числами (a,b) - это ребра (a,b) и (b,a)). Тогда наша задача сводится к нахождению пути в графе, проходящий через все его рёбра, являющейся циклом.

# Постановка задачи

# Необходимо написать программу, которая реализует алгоритм нахождения Эйлеровых циклов.

Программа должна работать в текстовом режиме работы интерфейса. Данный тип интерфейса использует при вводе-выводе и представлении информации исключительно набор буквенно-цифровых символов и символов псевдографики. Характеризуется малой требовательностью к ресурсам аппаратуры ввода-вывода (в частности, памяти) и высокой скоростью отображения информации.

Средой разработки является Microsoft Visual Studio 2019, языками программирования C++. Программа должна быть разработана для работы в операционной системе Microsoft Windows.

# Теоретическая часть задания

Рассматриваемая задача является одной из самых старейших в теории графов. В городе Кенигсберге (ныне Калининград) имелось семь мостов, соединяющих два берега реки Преголь, и два основа на ней друг с другом (рис. 2). Требуется, начав путешествие из одной точки города пройти по всем мостам по одному разу и вернуться в исходную точку.

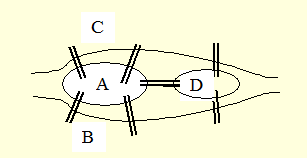


Рисунок 2- кенигсбергские мосты

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

**Определение.** 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.**

(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 ³1) 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.

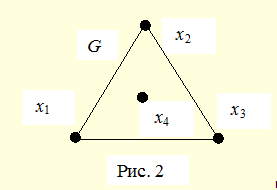


Рисунок 3 – Граф K3

**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. 3). 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 ¹a , то цепь m можно продолжить из вершины x в некотором направлении. Через некоторое число таких продолжений мы придем в вершину z ÎX, из которой нельзя продлить полученную простую цепь. Легко видеть, что z = a так как из всех остальных вершин цепь может выйти (четные степени!); a 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 все вершины имеют четные степени. Пусть G1’, G2’,…,Gk — компоненты связности графа G’ , содержащие хотя бы по одному ребру. Согласно предположению индукции все эти компоненты обладают эйлеровыми циклами m1 , m1 , …, mk соответственно. Так как граф G связан, то цепь m встречает каждую из компонент G1’, G2’,…,Gk. Пусть первые встречи цикла m с компонентами

G1’, G2’,…,Gk происходят соответственно в вершинах x 1 , x 2 , …, xk . Тогда простая цепь n(a , a )=m(a , x 1 ) Um1 (x 1 , x 1 ) Um(x 1 , x 2 ) U…Umk (xk , xk ) Um(xk , a ) является эйлеровым циклом в графе G . Теорема доказана.

# Описание алгоритма программы

Сначала проверим, существует ли эйлеров путь. Затем найдём все простые циклы и объединим их в один - это и будет эйлеровым циклом. Если граф таков, что эйлеров путь не является циклом, то, добавим недостающее ребро, найдём эйлеров цикл, потом удалим лишнее ребро.

Искать все циклы и объединять их будем мы будем при помощи данного алгоритма

stack St;  
в St кладём любую вершину (стартовая вершина);  
пока St не пустой  
 пусть V - значение на вершине St;  
 если степень(V) = 0, то  
 добавляем V к ответу;  
 снимаем V с вершины St;  
 иначе  
 находим любое ребро, выходящее из V;  
 удаляем его из графа;  
 второй конец этого ребра кладём в St;

Приведенная ниже алгоритм ищет и выводит эйлеров цикл или путь в графе, или выводит -1, если его не существует.

Сначала программа проверяет степени вершин: если вершин с нечётной степенью нет, то в графе есть эйлеров цикл, если есть 2 вершины с нечётной степенью, то в графе есть только эйлеров путь (эйлерова цикла нет), если же таких вершин больше 2, то в графе нет ни эйлерова цикла, ни эйлерова пути. Чтобы найти эйлеров путь, поступим таким образом: если V1 и V2 - это две вершины нечётной степени, то просто добавим ребро (V1,V2), в полученном графе найдём эйлеров цикл (он, очевидно, будет существовать), а затем удалим из ответа "фиктивное" ребро (V1,V2). Эйлеров цикл будем искать в точности так, как описано выше, и заодно по окончании этого алгоритма проверим, связный был граф или нет (если граф был не связный, то по окончании работы алгоритма в графе останутся некоторые рёбра, и в этом случае нам надо вывести -1).

1.Создаём вектор deg(n)

2. Для i=0 пока i<n делать i++

3. Для j=0 пока j<n делать j++

4. deg[i] += arr[i][j]

5. Конец цикла

6. Конец цикла

7. Для i=0 пока i<n делать i++

8. Если deg[i] & 1

9. Если v1=-1

10. v1=i

11. Инче если v2= -1

12. v2= i

13. Иначе bad = true

14. конец условия

15. Конец условия

16. Конц условия

17. Конец цикла

18. stack St;

19. Пока st не пустой

20. V – значение на вершине st

21. Если степень V = 0

22. Добавляем V к ответу

23. Снимаем V с вершины St

24. Иначе

25. Находим любое ребро, выходящее из V

26. Удаляем его из графа

27. Второй конец этого ребра кладём в St

28. Конец условия

29.Если v1 не равно -1

30. Для i=0 пока i+1<res.size() делать i++

31. Если res[i] =v1 и res[i+1]=v2 или res[i]= v2 и res[i+1]=v1

32. Для j=i+1 пока j<res.size() делать j++

33. Добавляем res[j] в конец

34. Конец цикла

35. Для j=1 пока j<=i делать j++

36. Добавляем res[j] в конец

37. Конец цикла

38. res=res2;

39. Конец условия

40. Конец цикла

41. Для i=0 пока i<n делать i++

42. Для j=0 пока j<n делать j++

43. Если arr[i][j] != 0

44. bad = true

45. Конец условия

46. Конец цикла

47. Конец цикла

# Описание программы

Программа начинается с того, что пользователю предлагают выбрать создания матрицы смежности

printf("Чтобы создать рандомный массив введите 1\n");

printf("Чтобы открыть массив из файла введите 2\n");

printf("Чтобы ввести массив вручную введите 3\n");

printf("Чтобы завершить программу введите 0\n");

scanf("%d", &select);

Если пользователь выберет первый вариант, матрица смежности будет задана рандомно

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

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

if (j > i) {

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

arr[j][i] = arr[i][j];

}

else if (i == j) arr[i][j] = 0;

}

}

Далее массив выдоиться в консоль

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

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

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

}

printf("\n");

}

Если пользователь выберет второй вариант, то матрица смежности будет считана из файла

За считывание матрицы из файла отвечает функция in();

Программа проверяет количество элементов в файле и записывает их в переменную m, != EOF означает что программа дошла до конца файла. После этого вызывается функция rewind, которая переносит указатель на начало файла. Далее находим размер полученный матрицы, путем вычисления квадратного корня из переменной m. После этого, программа безошибочно считывает матрицу из файла.

int in() {

int m = 0;

int s = 0, n = 0;

while ((fscanf\_s(f\_input, "%d", &s) != EOF)) {

if (!f\_input) break;

m += 1;

}

rewind(f\_input);

n = sqrt(m);

arr = dynamic(n);

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

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

fscanf\_s(f\_input, "%d", &arr[i][j]);

}

}

return n;

}

Если пользователь выберет 3 вариант, то матрицу смежности можно будет задать вручную. Функция ввод scanf запрашивает ввод у пользователя и передаёт введённое значение в массив arr

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

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

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

scanf("%d", &arr[i][j]);

}

После того как как программа задал граф матрицей смежности. Программа выводит матрицу в файл out.txt при помощи функции print(), результат показан на рисунке 4 . На вход в фунцию подаётся матрица и порядок матрицы

void print(int\*\* arr, int n) {

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

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

fprintf(f\_output, "%d ", arr[i][j]);

}

fprintf(f\_output, "\n");

}

fclose(f\_output);

}

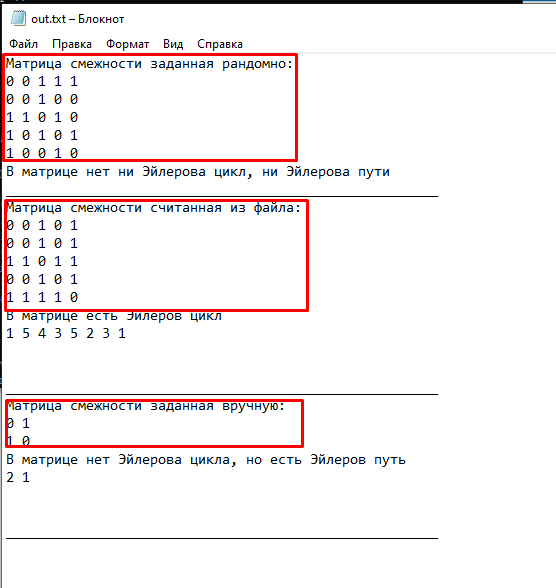


Рисунок 4- Вывод матрицы смежности в файл

Далее запускается алгоритм поиска эйлерова цикла, подробная работа которого описана выше, в 3 пункте

Далее происходит вывод результатов алгоритма в консоль и в файл out.txt

if (bad) {

puts("-1");

fprintf(f\_output1, "В матрице нет ни Эйлерова цикл, ни Эйлерова пути\n");

fprintf(f\_output1, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

else {

В этом куске кода мы проверяем, что нашёл наш алгоритм, эйлеров путь или цикл

if (res[0] == res[res.size()-1]) {

printf("В матрице есть Эйлеров цикл\n ");

fprintf(f\_output1, "В матрице есть Эйлеров цикл\n");

}

else {

printf("В матрице нет Эйлерова цикл, но есть Эйлеров путь\n");

fprintf(f\_output1, "В матрице нет Эйлерова цикла, но есть Эйлеров путь\n");

m = 1;

}

Далее происходит вывод результатов в файл и на консоль, результат показана на рисунке 5 и 6

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

printf("%d ", res[i] + 1);

fprintf(f\_output1, "%d ", res[i] + 1);

}

fprintf(f\_output1, "\n\n\n");

fprintf(f\_output1, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

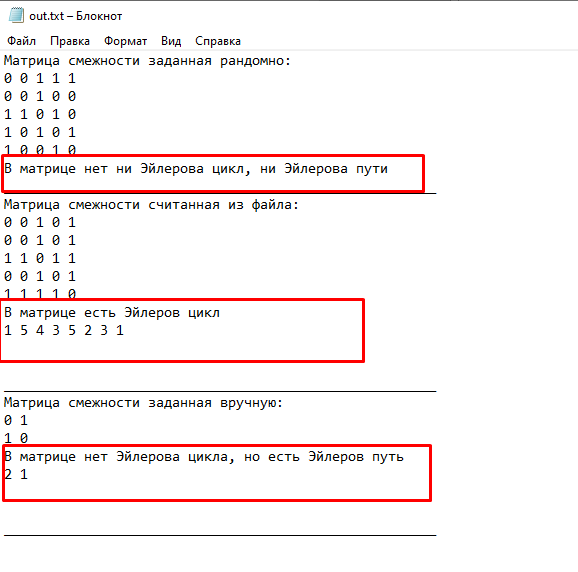


Рисунок 5- Вывод результата работы алгоритма в файл

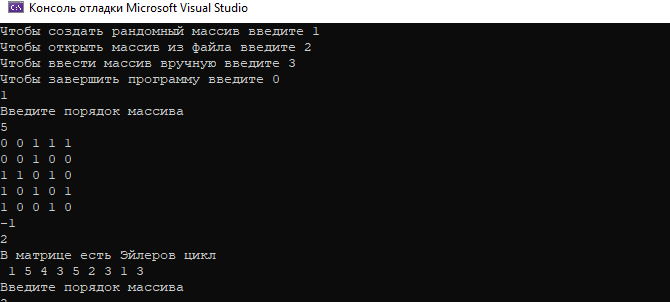


Рисунок 6- Вывод результата работы алгоритма в консоль

Также пользователю предоставляется возможность преобразования эйлерова пути в цикл.

Для этого, ещё на этапе формировании матрицы смежности она дублируется при помощи функции copy();.В которую передаются два массива. Уже сгенерированный и пустой

void copy(int\*\* arr1, int\*\* arr, int n) {

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

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

arr1[i][j] = arr[i][j];

}

После этого, когда программа в первый раз запускаем алгоритм она возвращает из него переменную m, которая показывает нам есть ли эйлеров путь в графе, если такой путь имеется, то программа предлагает выбрать, хочет ли пользователь преобразовать эйлеров путь в граф или нет.

if (m == 1) {

printf("\nЧтобы преобразовать эйлеров путь в цикл, введите один, для выхода введите 0\n");

scanf("%d", &select2);

if (select2 == 1) {

el(n, arr1);

alg(n, arr1);

}

}

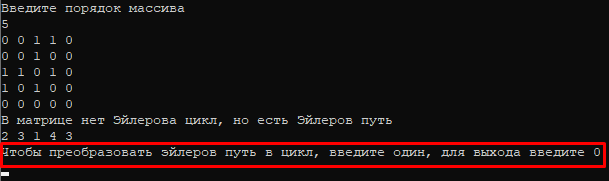


Рисунок 7*-* Результат работы программы после преобразования графа

Если пользователь выбирает преобразовать, то запускается функция el();

void el(int n, int\*\* arr) {

int\* mas;

int v1 = -1, v2 = -1;

В этом куске кода программа создаёт массив с размерностью равной порядком графа

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

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

mas[i] = 0;

}

Далее программа ищет количество рёбер у вершин путём складывания всех рёбер вершины

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

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

mas[i] = mas[i] + arr[i][j];

В этом куске кода программа ищет две нечётные вершины, для того чтобы проверить вершину на нечётность, нужно чтобы при делении на 2 остаток был не равен нулю

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

if (mas[i] % 2 != 0) {

if (v1 == -1)

v1 = i;

else if (v2 == -1)

v2 = i;

}

}

printf("\n");

Далее программа просматривает есть ли ребро между вершинами или нет. Если между вершинами есть ребро, то алгоритм его удаляет, если нет, то добавляет.

if (arr1[v1][v2] == 0) {

printf("Эйлеров цикл был получен путём добовления ребра между вершинами %d и %d\n", v1, v2);

arr1[v1][v2] = 1;

arr1[v2][v1] = 1;

}

else {

printf("Эйлеров цикл был получен путём удаления ребра между вершинами %d и %d\n", v1, v2);

arr1[v1][v2] = 0;

arr1[v2][v1] = 0;

}

}

После того, как функция el() завершатся, функция alg(),(Алгоритм поиска эйлерова цикла) запускается второй раз.

Далее происходит вывод преобразованной матрицы в консоль. Также программ выводит между какими вершинами было добавлено или удалено ребро, результат показан на рисунке 8

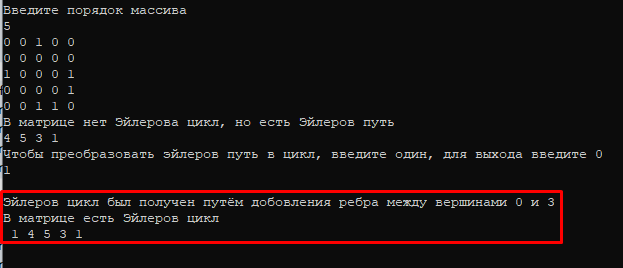


Рисунок 8-Вывод между какими вершинами было добавлено ребро

# Тестирование

Среда разработки MicrosoftVisualStudio 2019 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 и выводом данных, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.

Ниже в таблице 1 продемонстрирован план и результат тестирования программы при выборе разных пунктов программы.

|  |  |  |  |
| --- | --- | --- | --- |
| **Описание теста** | **Проделанные действия** | **Ожидаемый результат** | **Полученный р результат** |
| Рандомная генерация матрицы с указанием размера графа | Переход в меню, выбор пункта 1 (ввод 1), выбор размера матрицы (ввод 4) | Вывод рандомно  сгенерированной матрицы  (Рисунок 9) | Верно |
| Вывод матрицы  из файла | Переход в меню,  выбор пункта 2 | Вывод самостоятельно заданной матрицы из файла (рисунок 10, 11) | Верно |
| Вывода матрицы, сгенерированной вручную в консоли с указанием размера | Переход в меню, выбор пункта 3 (ввод 3), выбор размера матрицы (ввод 3) | Вывод самостоятельно заданной матрицы с размером 3 (рисунок 12) | Верно |
| Вывод результата работы алгоритма, при рандомной матрицы, введённой из файла и ручном вводе | Переход в меню, выбор пунктов 1, 2,3. Выбор размера матрицы и ввод матрицы вручную | Вывод эйлерова цикла или эйлерова пути или их отсутствие (Рисунок 13) | Верно |
| Вывод результатов работы программы в файл | Выход из пунктов программы после вывода результата программы в консоль. | Успешное сохранение результатов  программы (Рисунок 14) | Верно |
| Вывод результатов преобразования эйлерова пути в цикл | Переход в меню (ввод 1),  выбор размера матрицы (ввод 4), выбор преобразования эйлерова пути в цикл | Вывод преобразованного эйлерова пути  (Рисунок 15) | Верно |
| Ошибка открытия файла | Переход в меню(ввод 2) | Вывод “Не удалось открыть файл” (Рисунок 17) | Верно |
| Неправильный ввод данных в меню | Переход в меню (ввод1, 2,3,4) | Вывод “Данные введены не верно” (рисунок18) | Верно |

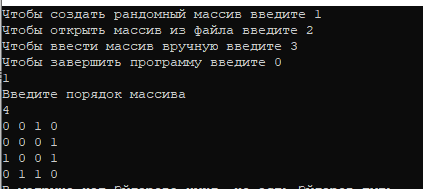


Рисунок 9-Вывод рандомно сгенерированной матрицы

На рисунке 9 продемонстрирован вывод рандомно сгенерированной матрицы смежности

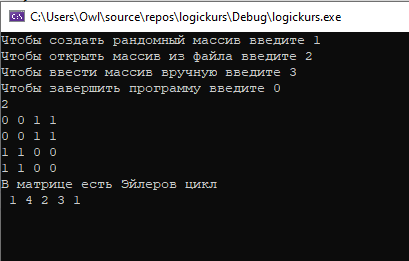


Рисунок 10-Вывод самостоятельно заданной матрицы из файла

На рисунке 10 продемонстрирован вывод самостоятельно заданной матрицы из файла в консоль

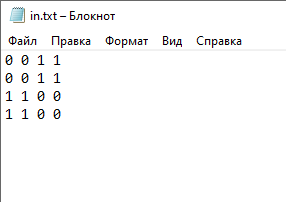


Рисунок 11-Вывод самостоятельно заданной матрицы из файла

На рисунке 11 продемонстрирован вывод самостоятельно заданной матрицы из файла в выходной файл

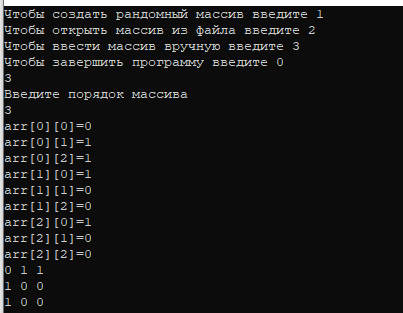


Рисунок 12-Вывод самостоятельно заданной матрицы

На рисунке 12 продемонстрирован вывод самостоятельно заданной матрицы

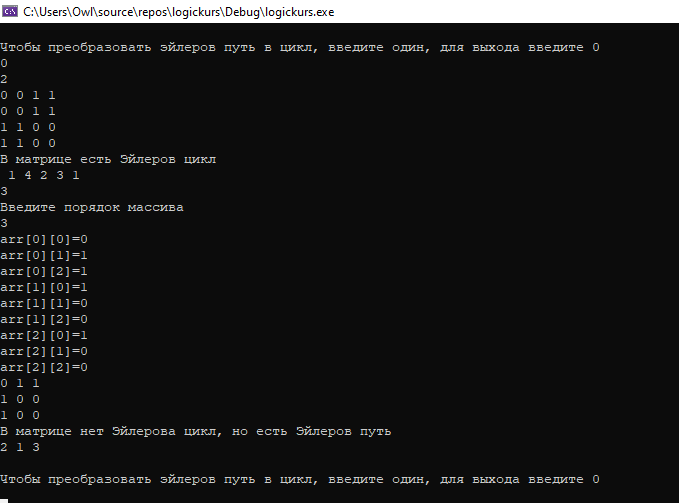


Рисунок 13-Вывод эйлерова цикла или эйлерова пути или их отсутствие

На рисунке 13 продемонстрирован вывод эйлерова цикла или эйлерова пути или их отсутствия, после окончания работы алгоритма программы

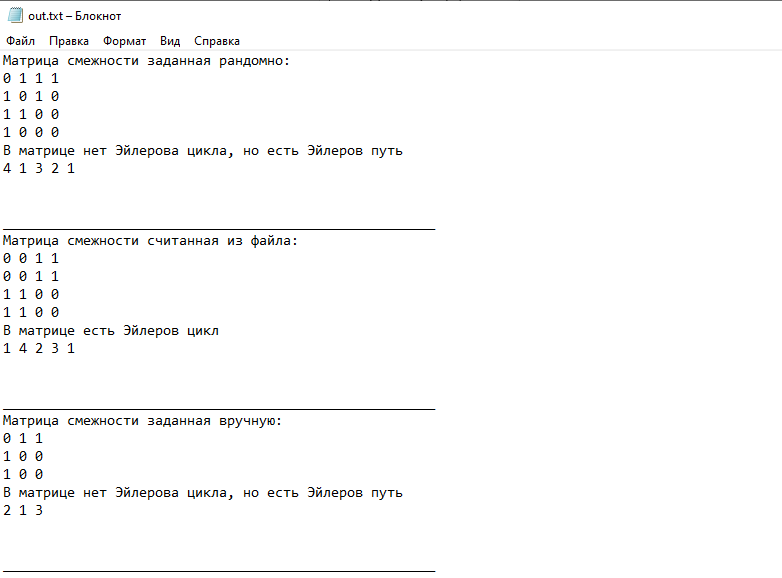


Рисунок 14-Успешное сохранение результатов программы в файл

На рисунке 14 продемонстрировано успешное сохранение результатов программы в файл, после завершения работы программы

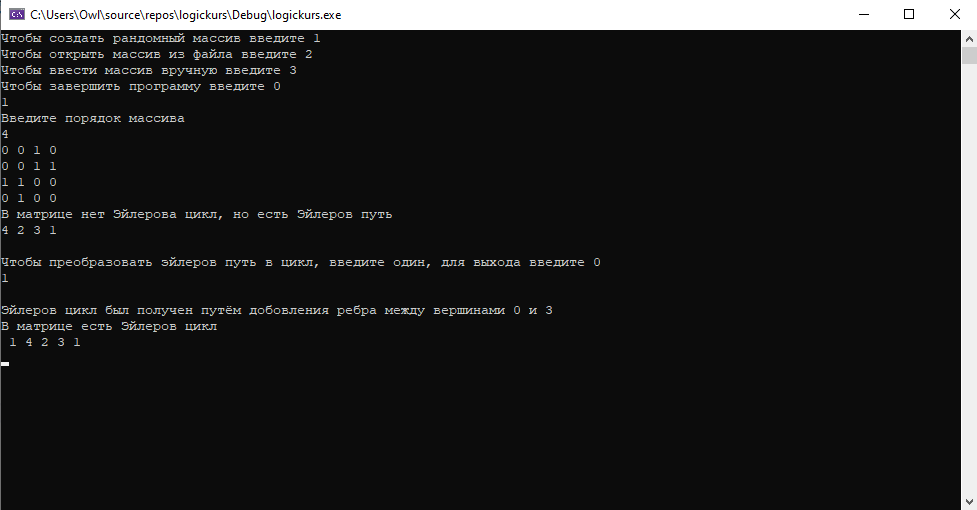


Рисунок 15-Вывод преобразованного эйлерова пути

На рисунке 15 продемонстрирован вывод преобразованной эйлерова пути в цикл, после ввода пользователем 1 или 0

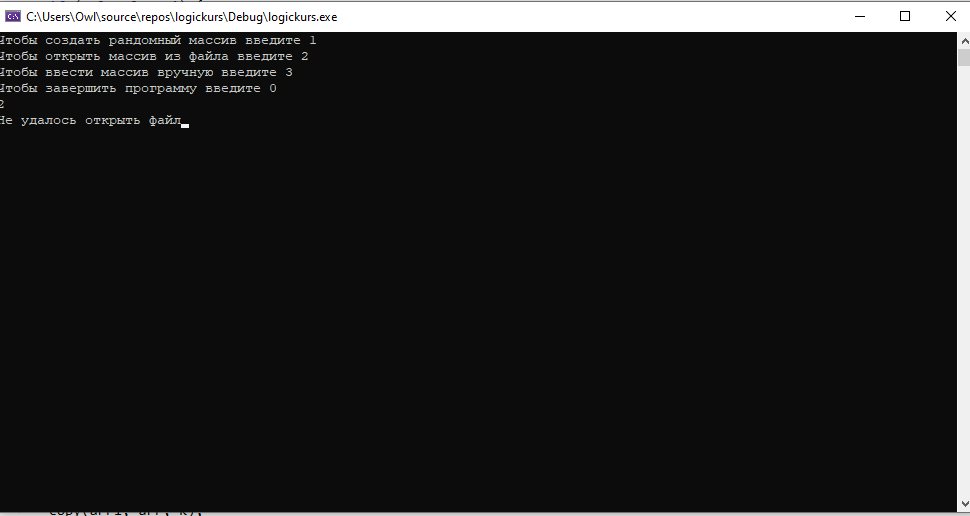


Рисунок 17-Вывод ошибки при неверном файле

На рисунке 17 продемонстрирован вывод ошибки на консоль, если не удаться открыть файл

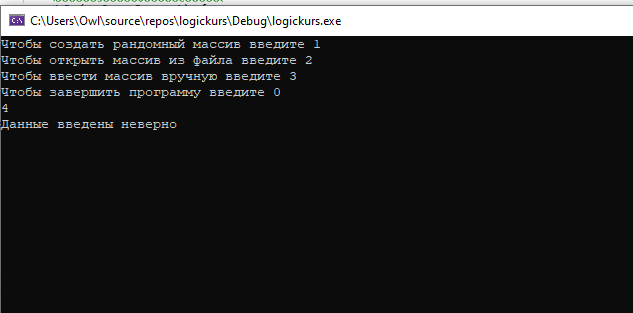


Рисунок 18-Введение неверных данных в меню

На рисунке 18 продемонстрирован вывод ошибки на консоль, если данные в меню были введены неверно

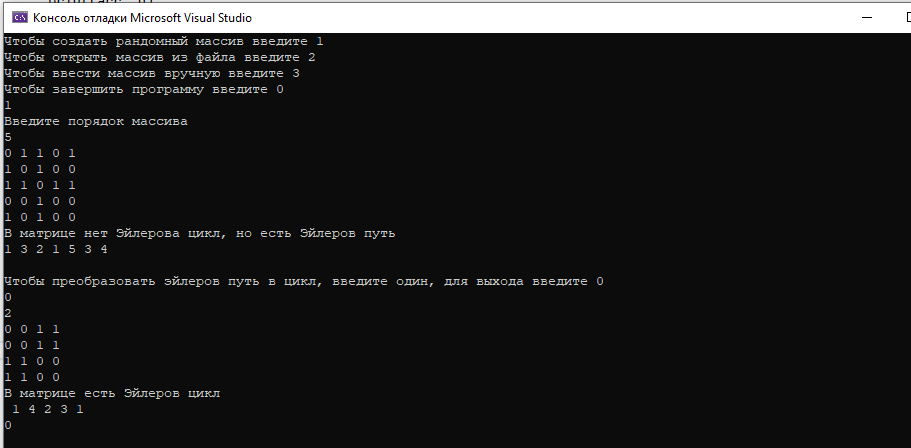


Рисунок 19-Вывод результатов в консоль

**Ручной просчет программы.**

Проведем проверку программы посредством ручных вычислений на примере неориентированного графа с 4 вершинами

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |

Построим этот граф(Рисунок 20)

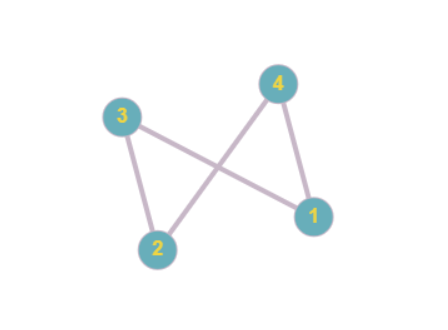


Рисунок 20 граф

Для поиска эйлерова цикла совершим обход по графу представленном на рисунке 20, обход начнём с первой вершины.

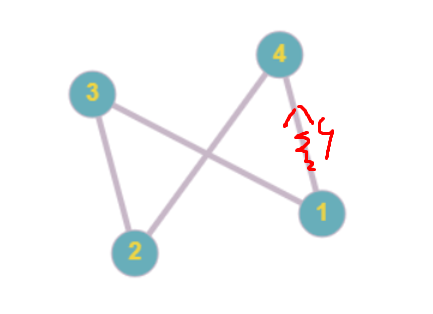


Рисунок 21- Путь по графу

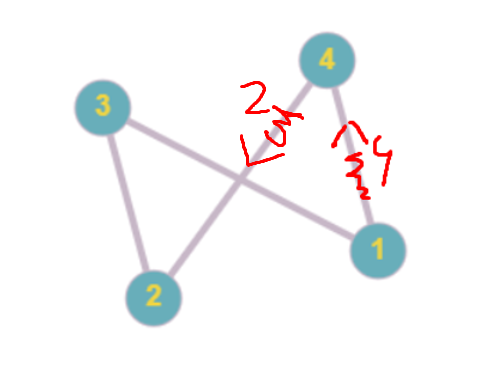


Рисунок 22- Путь по графу

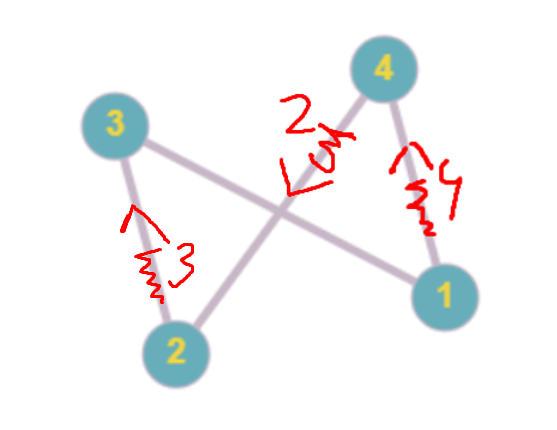


Рисунок 23- Путь по графу

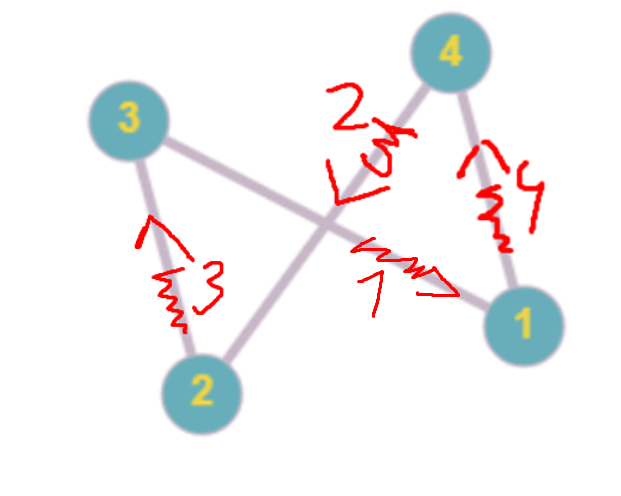


Рисунок 24- Путь по графу

На рисунков 21,22,23,24 виден эйлеров цикл: 1-4-2-3-1

Теперь сравним ручной просчёт результатами программы(Рисунок 25)

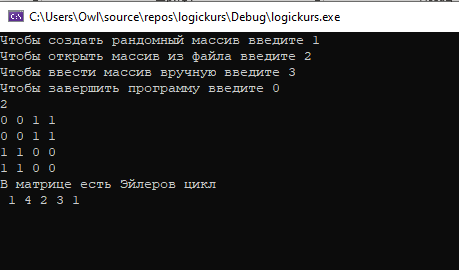


Рисунок 25 - Результат работы

Ручной расчёт совпал с результатом работы программы, таким образом, можно сделать вывод, что программа работает верно

# Заключение

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения Эйлеровых циклов в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц. Приобретены навыки работы с файлами, а также работы с командной строкой. Углублены знания языка программирования C++.

**Список источников.**

1. <https://www.cyberforum.ru/>
2. Кристофидес Н. «Теория графов. Алгоритмический подход» - Мир, 1978
3. <https://graphonline.ru/>

Приложение А

**Листинг программы.**

Файл header.h

#pragma once

#define Function

#ifdef Function

int in();

int alg(int n, int\*\* arr);

int\*\* dynamic(int count);

void print(int\*\* arr, int n);

void el(int n, int\*\* arr);

void copy(int\*\* arr1, int\*\* arr, int n);

#endif

Файл header2.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <conio.h>

#include <queue>

#include <iostream>

#include <vector>

#include <stack>

Файл func.cpp

#include "Header.h"

#include "Header2.h"

using namespace std;

FILE\* f\_output1;

int\*\* dynamic(int count) // Создание динамического массива.

{

int\*\* mass;

mass = (int\*\*)malloc(count \* sizeof(int\*));

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

mass[i] = (int\*)malloc(count \* sizeof(int));

return mass;

}

int alg(int n, int\*\* arr) {

int m = 0;

f\_output1 = fopen("out.txt", "a");

vector<int> deg(n);

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

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

deg[i] += arr[i][j];

int first = 0;

while (!deg[first]) ++first;

int v1 = -1, v2 = -1;

bool bad = false;

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

if (deg[i] & 1)

if (v1 == -1)

v1 = i;

else if (v2 == -1)

v2 = i;

else

bad = true;

if (v1 != -1)

++arr[v1][v2], ++arr[v2][v1];

stack<int> st;

st.push(first);

vector<int> res;

while (!st.empty())

{

int v = st.top();

int i;

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

if (arr[v][i])

break;

if (i == n)

{

res.push\_back(v);

st.pop();

}

else

{

--arr[v][i];

--arr[i][v];

st.push(i);

}

}

if (v1 != -1)

for (size\_t i = 0; i + 1 < res.size(); ++i)

if (res[i] == v1 && res[i + 1] == v2 || res[i] == v2 && res[i + 1] == v1)

{

vector<int> res2;

for (size\_t j = i + 1; j < res.size(); ++j)

res2.push\_back(res[j]);

for (size\_t j = 1; j <= i; ++j)

res2.push\_back(res[j]);

res = res2;

break;

}

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

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

if (arr[i][j])

bad = true;

if (bad) {

puts("-1");

fprintf(f\_output1, "В матрице нет ни Эйлерова цикл, ни Эйлерова пути\n");

fprintf(f\_output1, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

else {

if (res[0] == res[res.size()-1]) {

printf("В матрице есть Эйлеров цикл\n ");

fprintf(f\_output1, "В матрице есть Эйлеров цикл\n");

}

else {

printf("В матрице нет Эйлерова цикл, но есть Эйлеров путь\n");

fprintf(f\_output1, "В матрице нет Эйлерова цикла, но есть Эйлеров путь\n");

m = 1;

}

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

printf("%d ", res[i] + 1);

fprintf(f\_output1, "%d ", res[i] + 1);

}

printf("\n");

fprintf(f\_output1, "\n\n\n");

fprintf(f\_output1, "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

fclose(f\_output1);

return m;

}

Файл logickurs.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <windows.h>

#include "Header.h"

#include "Header2.h"

FILE\* f\_output;

FILE\* f\_input;

int\* num;

int\*\* arr;

int\*\* arr1;

int\* ar;

int n;

int m;

int main() {

using namespace std;

vector<int> ress;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

int select, select2;

printf("Чтобы создать рандомный массив введите 1\n");

printf("Чтобы открыть массив из файла введите 2\n");

printf("Чтобы ввести массив вручную введите 3\n");

printf("Чтобы завершить программу введите 0\n");

scanf("%d", &select);

f\_output = fopen("out.txt", "w");

fclose(f\_output);

while (select != 0) {

f\_output = fopen("out.txt", "a");

if (select == 1) {

n = 0;

printf("Введите порядок массива\n");

while (n==0)

{

scanf("%d", &n);

if (n == 0) {

printf("Порядок массива не может быть равнен нулю\n");

}

}

arr = dynamic(n);

arr1 = dynamic(n);

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

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

if (j > i) {

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

arr[j][i] = arr[i][j];

}

else if (i == j) arr[i][j] = 0;

}

}

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

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

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

}

printf("\n");

}

copy(arr1, arr, n);

fprintf(f\_output, "Матрица смежности заданная рандомно: \n");

print(arr, n);

m=alg(n, arr);

if (m == 1) {

printf("\nЧтобы преобразовать эйлеров путь в цикл, введите один, для выхода введите 0\n");

scanf("%d", &select2);

if (select2 == 1) {

el(n, arr1);

alg(n, arr1);

}

}

}

if (select == 2) {

if ((f\_input = fopen("in.txt", "r")) == NULL)

{

printf("Не удалось открыть файл");

getchar();

}

else {

int k;

k = in();

fprintf(f\_output, "Матрица смежности считанная из файла: \n");

print(arr, k);

for (int i = 0; i < k; i++) {

for (int j = 0; j < k; j++) {

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

}

printf("\n");

}

arr1 = dynamic(k);

copy(arr1, arr, k);

m = alg(k, arr);

if (m == 1) {

printf("\nЧтобы преобразовать эйлеров путь в цикл, введите один, для выхода введите 0\n");

scanf("%d", &select2);

if (select2 == 1) {

el(k, arr1);

alg(k, arr1);

}

}

fclose(f\_input);

}

}

if (select == 3) {

printf("Введите порядок массива\n");

n = 0;

while (n == 0)

{

scanf("%d", &n);

if (n == 0) {

printf("Порядок массива не может быть равнен нулю\n");

}

}

arr = dynamic(n);

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

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

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

scanf("%d", &arr[i][j]);

}

}

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

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

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

}

printf("\n");

}

arr1 = dynamic(n);

fprintf(f\_output, "Матрица смежности заданная вручную: \n");

print(arr, n);

copy(arr1, arr, n);

m = alg(n, arr);

if (m == 1) {

printf("\nЧтобы преобразовать эйлеров путь в цикл, введите один, для выхода введите 0\n");

scanf("%d", &select2);

if (select2 == 1) {

el(n, arr1);

alg(n, arr1);

}

}

}if (select > 3) {

printf("Данные введены неверно");

}

scanf("%d", &select);

}

}

int in() {

int m = 0;

int s = 0, n = 0;

while ((fscanf\_s(f\_input, "%d", &s) != EOF)) {

if (!f\_input) break;

m += 1;

}

rewind(f\_input); // перематываем файл для повторного чтения

n = sqrt(m);

arr = dynamic(n);

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

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

fscanf\_s(f\_input, "%d", &arr[i][j]);

}

}

return n;

}

void print(int\*\* arr, int n) {

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

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

fprintf(f\_output, "%d ", arr[i][j]);

}

fprintf(f\_output, "\n");

}

fclose(f\_output);

}

void el(int n, int\*\* arr) {

int\* mas;

int v1 = -1, v2 = -1;

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

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

mas[i] = 0;

}

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

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

mas[i] = mas[i] + arr[i][j];

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

if (mas[i] % 2 != 0) {

if (v1 == -1)

v1 = i;

else if (v2 == -1)

v2 = i;

}

}

printf("\n");

if (arr1[v1][v2] == 0) {

printf("Эйлеров цикл был получен путём добовления ребра между вершинами %d и %d\n", v1, v2);

arr1[v1][v2] = 1;

arr1[v2][v1] = 1;

}

else {

printf("Эйлеров цикл был получен путём удоления ребра между вершинами %d и %d\n", v1, v2);

arr1[v1][v2] = 0;

arr1[v2][v1] = 0;

}

}

void copy(int\*\* arr1, int\*\* arr, int n) {

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

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

arr1[i][j] = arr[i][j];

}