Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

по курсу «ЛиОАвИЗ»

на тему «Реализация алгоритма нахождения Эйлеровых циклов»

Выполнил:

студент группы 23ВВВ2

Герасимов В. Р.

Принял:

доцент

Юрова О. В.

Пенза 2024

**Содержание**

Оглавление

[**Реферат** 5](#_Toc185287277)

[**Введение** 6](#_Toc185287278)

[**Постановка задачи** 7](#_Toc185287279)

[**Описание алгоритма программы.** 8](#_Toc185287280)

[**Описание программы** 9](#_Toc185287281)

[**Тестирование** 13](#_Toc185287282)

[**Заключение** 15](#_Toc185287283)

[**Список литературы** 16](#_Toc185287284)

[**Приложение А.** 17](#_Toc185287285)

[**Листинг программы.** 17](#_Toc185287286)

# **Реферат**

Отчет 28 стр, 11 рисунков.

Цель исследования - разработать эффективный алгоритм нахождения Эйлерова цикла.

В работе рассмотрены правила заполнения матрицы, обхода в глубину, на основе которого проверяется условие связности графа. Установлено, что с помощью данного алгоритма можно проверить любой граф на наличие Эйлерова цикла в нем.

# **Введение**

В современном мире информационные технологии играют ключевую роль во многих сферах деятельности. Разработка и анализ алгоритмов являются основой для создания эффективных и надёжных программных решений. В данном курсовом проекте рассматривается реализация алгоритма нахождения эйлерова цикла, который является одним из фундаментальных алгоритмов в теории графов. Эйлеров цикл — это путь, проходящий через все рёбра графа ровно один раз. Этот алгоритм имеет множество применений в различных областях, таких как планирование маршрутов, оптимизация транспортных сетей и анализ социальных сетей.

В ходе выполнения курсового проекта будет создан код на выбранном языке программирования, который позволит находить эйлеровы циклы в графах различной структуры и размера. Результаты тестирования покажут эффективность предложенного алгоритма и его применимость в реальных задачах.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2010, язык программирования – Си.

Целью данной курсовой работы является разработка программы на языке Си, который является широко используемым.

# **Постановка задачи**

Требуется разработать программу, которая реализует Эйлеров цикл, проверку на связность и четность степеней.

Исходный граф в программе должен задаваться матрицей смежности. Пользователь при генерации должен выбрать матрица будет ориентированная или не ориентированная. Так же граф может быть введен вручную или же скопирован из файла.

Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. Необходимо предусмотреть различные исходы обхода, чтобы программа не выдавала ошибок и работала правильно.

Устройство ввода – клавиатура и мышь.

# **Описание алгоритма программы.**

Для реализации программы использовалось 5 функций: createG1 – для генерации ориентированной матрицы, create G2 – для генерации неориентированной матрицы, printG – для вывода матрицы, Euler – для самого Эйлерового цикла, dfs – для проверки на связность граф.

В начале программы пользователь встречает заставку с полезной информацией о курсовом проекте. Далее появляется меню, состоящее из 8 пунктов (где 0 – выход из программы, а 7 – сохранение результатов в файл).

Итак, после заполнения графа любым способом из предложенных (а именно: генерация ориентированного графа, генерация неориентированного графа, ручной ввод графа, ввод из файла) пользователь выбирает 5 пункт – непосредственно обход графа.

Вводит номер вершины, с которой начнется обход. Значение заносится в переменную start.

Инициализируется массив степеней. Далее выполняется проверка на четность всех вершин графа. Если хоть одна степень будет нечетная, то обход не начнется.

Если проверка пройдена, проверяется условие связности графа. Если будет хоть одна не посещенная вершина, то обход не начнется.

И после этого уже выполняется сам обход графа.

# **Описание программы**

Для написания данной программы использован язык программирования Си. Язык программирования Си - универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

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

Данная программа является многомодульной, поскольку состоит из нескольких функций: createG1, createG2, printG, Euler, dfs.

Работа программы начинается с вывода важной информации. Далее выводится меню, пункты которого пользователю необходимо будет выбрать.

printf("Курсовая работа\nПо дисциплине ЛиОАВИЗ\nНа тему: Реализация Эйлерова цикла\nВыполнил студент 23ВВВ2: Герасимов В. Р.\nПриняла: Юрова О. В.\n\n");

printf("0.Выход\n");

printf("1.Сгенерировать ориентированную матрицу\n");

printf("2.Сгенерировать неориентированную матрицу\n");

printf("3.Ввести матрицу из файла\n");

printf("4.Ввести матрицу вручную\n");

printf("5.Обход\n");

printf("6.Изменить матрицу\n");

printf("7.Сохранить результат в файл\n");

printf("Выберите пункт меню:\n");

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

При выборе 0, программа завершает свою работу. При выборе 1, генерируется ориентированная матрица (рисунок 1).

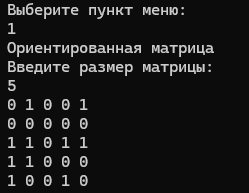


Рисунок 1 – пункт 1

При выборе пункта 2, генерируется неориентированная матрица (рисунок 2).

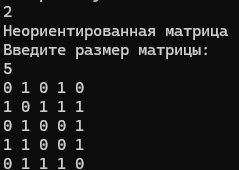


Рисунок 2 – пункт 2.

При выборе пункта 3, запросится название файла, и оттуда скопируется матрица (рисунок 3).

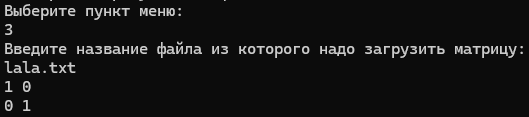


Рисунок 3 – пункт 3

При выборе пункта 4, пользователь вводит размер матрицы и затем поэлементно вводит 1/0 (рисунок 4).

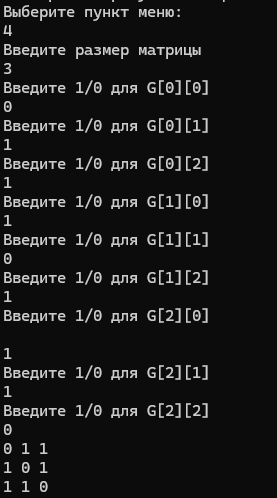


Рисунок 4 – пункт 4

При выборе пункта 5, пользователь вводит номер вершины, с которой начнется обход (рисунок 5).

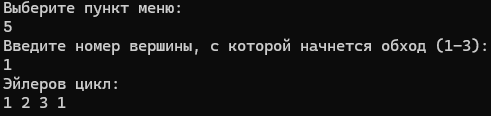


Рисунок 5- пункт 5

При выборе пункта 6, пользователю будет необходимо самостоятельно определить, как необходимо исправить граф, чтобы он имел Эйлеров цикл (рисунок 6).

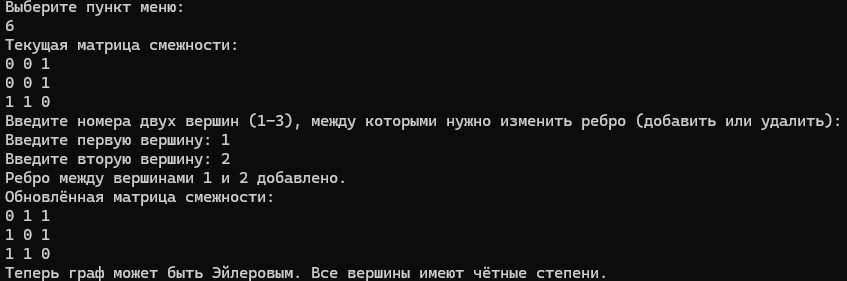
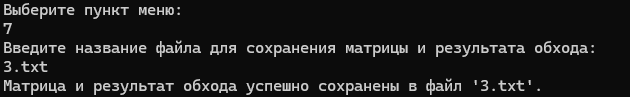


Рисунок 6 – пункт 6

При выборе пункта 7, пользователю необходимо ввести название файла, в который необходимо занести результат обхода (рисунок 7).

Рисунок 7 – пункт 7

# **Тестирование**

Среда разработки MicrosoftVisualStudio 2010 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 продемонстрированы результаты тестирования программы.

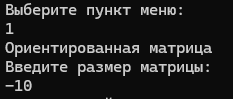


Рисунок 8 – Тестирование при вводе отрицательного размера.

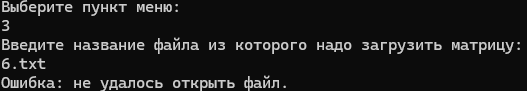


Рисунок 9 – Тестирование при вводе несуществующего файла.

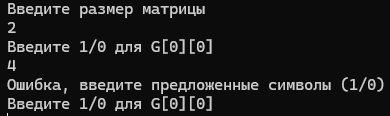


Рисунок 10 – Тестирование при вводе недопустимого символа

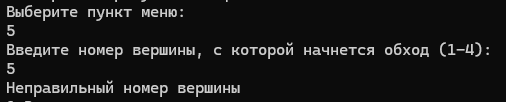
.

Рисунок 11 – Тестирование при вводе недопустимого номера вершин

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Приветственная заставка, вывод меню | Верно |
| Выбор генерации ориентированной матрицы | Ввод размера, вывод матрицы | Верно |
| Выбор генерации неориентированной матрицы | Ввод размера, вывод матрицы | Верно |
| Выбор ввода матрицы из файла | Ввод названия файла, вывод матрицы | Верно |
| Выбор ввода матрицы вручную | Ввод ребер вручную для каждых вершин, вывод матрицы | Верно |
| Выбор обхода | Ввод стартовой вершины, вывод Эйлерова цикла | Верно |
| Выбор изменения матрицы | Ввод двух вершин, вывод изменённой матрицы | Верно |
| Выбор сохранения в файл | Ввод названия файла, ввод матрицы и результата | Верно |

# 

# **Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения Эйлерова цикла в MicrosoftVisualStudio 2010.

При выполнении данной курсовой работы были получены навыки разработки программы и освоены приемы создания матриц смежностей, обхода в глубины и реализации нахождения эйлерова цикла. Углублены знания языка программирования 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. Баранов, С. В. Алгоритмы и структуры данных. — М.: Наука, 2020. — 384 с.
2. Гусев, В. Л. Основы программирования на языке C. — М.: ДМК Пресс, 2021. — 400 с.
3. Кузнецов, С. В. Графы и алгоритмы. — Самара: Самарский гос. университет, 2019. — 450 с.
4. Лебедев, А. П. Программирование на C для начинающих. — М.: Велигор, 2021. — 350 с.
5. Николаев, И. В. Введение в теорию графов. — Екатеринбург: Урал. университет, 2021. — 300 с.
6. Павлов, Р. И. Эйлеровы циклы. Алгоритмическое исследование. — Ростов: ЮФУ, 2018. — 150 с.
7. Петров, А. Н. Алгоритмы для обработки графов. — М.: Высшая школа, 2019. — 275 с.
8. Смирнов, И. Н. Основы создания программ. — М.: Гуманитарный издательский центр, 2021. — 500 с.
9. Тихонов, А. В. Современное программирование на C. — М.: БХВ-Петербург, 2022. — 480 с.

# **Приложение А.**

## **Листинг программы.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

int\*\* createG1(int size) {

int\*\* G = (int\*\*)malloc(size \* sizeof(int\*));

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

G[i] = (int\*)malloc(size \* sizeof(int));

}

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

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

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

if (i == j) {

G[i][j] = 0;

}

}

}

return G;

}

int\*\* createG2(int size) {

int\*\* G = (int\*\*)malloc(size \* sizeof(int\*));

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

G[i] = (int\*)malloc(size \* sizeof(int));

}

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

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

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

if (i == j) {

G[i][j] = 0;

}

}

}

return G;

}

void printG(int\*\* G, int size) {

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

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

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

}

printf("\n");

}

return;

}

void Euler(int\*\* graph, int v, int n, int\* degree, int\* result, int\* index) {

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

if (graph[v][u] == 1) {

graph[v][u] = graph[u][v] = 0;

degree[v]--;

degree[u]--;

Euler(graph, u, n, degree, result, index);

}

}

result[(\*index)++] = v + 1; // Сохраняем вершину в результат

}

void dfs(int v, int\* visited, int\*\* graph, int n) {

visited[v] = 1; // Помечаем вершину как посещенную

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

if (graph[v][u] == 1 && !visited[u]) {

dfs(u, visited, graph, n);

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

printf("Курсовая работа\nПо дисциплине ЛиОАВИЗ\nНа тему: Реализация Эйлерова цикла\nВыполнил студент 23ВВВ2: Герасимов В. Р.\nПриняла: Юрова О. В.\n\n");

system("PAUSE");

system("cls");

int n = 5, start, search = 1;

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

int\*\* graph = NULL;

int\*\* graph2 = NULL;

int\* result = (int\*)malloc(n \* n \* sizeof(int)); // Массив для хранения результата обхода

int index = 0; // Индекс для записи в массив результата

while (search != 0) {

printf("0.Выход\n");

printf("1.Сгенерировать ориентированную матрицу\n");

printf("2.Сгенерировать неориентированную матрицу\n");

printf("3.Ввести матрицу из файла\n");

printf("4.Ввести матрицу вручную\n");

printf("5.Обход\n");

printf("6.Изменить матрицу\n");

printf("7.Сохранить результат в файл\n");

printf("Выберите пункт меню:\n");

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

switch (search) {

case 0: {

printf("До свидания! Я рассчитываю на оценку 5! \n");

break;

}

case 1: {

printf("Ориентированная матрица\n");

printf("Введите размер матрицы: \n");

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

if(n<0){

printf("Недопустимый размер\n");

break;

}

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

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

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

}

graph = createG1(n);

printG(graph, n);

break;

}

case 2: {

printf("Неориентированная матрица\n");

printf("Введите размер матрицы: \n");

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

if (n < 0) {

printf("Недопустимый размер\n");

break;

}

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

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

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

}

graph = createG2(n);

printG(graph, n);

break;

}

case 3: {

printf("Введите название файла из которого надо загрузить матрицу:\n");

char filename[100];

scanf\_s("%s", filename, (unsigned)\_countof(filename));

FILE\* file;

fopen\_s(&file, filename, "r");

if (file == NULL) {

printf("Ошибка: не удалось открыть файл.\n");

break;

}

fscanf\_s(file, "%d", &n);

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 = 0; j < n; j++) {

fscanf\_s(file, "%d", &graph[i][j]);

}

}

fclose(file);

printG(graph, n);

break;

}

case 4: {

int timevar;

printf("Введите размер матрицы\n");

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

if (n < 0) {

printf("Недопустимый размер\n");

break;

}

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 = 0; j < n; j++) {

printf("Введите 1/0 для G[%d][%d]\n", i, j);

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

if (timevar == 1 || timevar == 0) {

graph[i][j] = timevar;

}

else {

printf("Ошибка, введите предложенные символы (1/0)\n");

i = 0;

j = -1;

}

}

}

printG(graph, n);

break;

}

case 5: {

if (graph == NULL) {

printf("Ошибка: матрица не существует. Сначала создайте или загрузите матрицу.\n");

break;

}

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

graph2 = (int\*\*)malloc(n \* sizeof(int\*));

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

graph2[i] = (int\*)malloc(n \* sizeof(int));

}

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

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

graph2[i][j] = graph[i][j];

}

}

printf("Введите номер вершины, с которой начнется обход (1-%d): \n", n);

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

if (start > n) {

printf("Неправильный номер вершины\n");

break;

}

start = start - 1;

// Проверка, что все вершины имеют четные степени

int isEulerian = 1; // Флаг для проверки, является ли граф Эйлеровым

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

degree[i] = 0; // Инициализация массива степеней

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

degree[i] += graph[i][j]; // Вычисление степени вершины i

}

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

isEulerian = 0; // Если у любой вершины нечетная степень

printf("Нечетная степень\n");

break;

}

}

// Проверка, что граф связный

if (isEulerian) {

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

visited[i] = 0;

}

dfs(start, visited, graph, n); // Начинаем DFS с заданной стартовой вершины

// Проверка, что все вершины с ненулевой степенью были посещены

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

if (degree[i] > 0 && !visited[i]) {

printf("Вершина не была посещена\n");

isEulerian = 0; // Если любая вершина с ребрами не была посещена

break;

}

else if (degree[i] == 0) {

printf("Нулевая степень\n");

isEulerian = 0; // Если у вершины нулевая степень, значит если мы в нее попадем,то выйти не сможем.

break;

}

}

}

// Если граф Эйлеров, выполняем обход Эйлера

if (isEulerian) {

printf("Эйлеров цикл:\n");

Euler(graph, start, n, degree, result, &index);

for (int i = index - 1; i >= 0; i--) { // Выводим результат в обратном порядке

printf("%d ", result[i]);

}

printf("\n");

}

else {

printf("Граф не имеет Эйлерова цикла.\n");

}

free(visited);

break;

}

case 6: {

if (graph == NULL) {

printf("Ошибка: матрица не существует. Сначала создайте или загрузите матрицу.\n");

break;

}

int v1, v2;

printf("Текущая матрица смежности:\n");

printG(graph, n);

printf("Введите номера двух вершин (1-%d), между которыми нужно изменить ребро (добавить или удалить):\n", n);

printf("Введите первую вершину: ");

scanf\_s("%d", &v1);

printf("Введите вторую вершину: ");

scanf\_s("%d", &v2);

// Преобразуем номера вершин в индексы массива

v1 -= 1;

v2 -= 1;

// Проверяем корректность ввода

if (v1 < 0 || v1 >= n || v2 < 0 || v2 >= n) {

printf("Ошибка: некорректные номера вершин.\n");

break;

}

// Изменяем ребро: если оно есть, удаляем; если его нет, добавляем

if (graph[v1][v2] == 1) {

graph[v1][v2] = 0;

graph[v2][v1] = 0;

printf("Ребро между вершинами %d и %d удалено.\n", v1 + 1, v2 + 1);

}

else {

graph[v1][v2] = 1;

graph[v2][v1] = 1;

printf("Ребро между вершинами %d и %d добавлено.\n", v1 + 1, v2 + 1);

}

// Пересчитываем степени вершин

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

degree[i] = 0;

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

degree[i] += graph[i][j];

}

}

// Выводим обновлённую матрицу смежности

printf("Обновлённая матрица смежности:\n");

printG(graph, n);

// Проверяем, стали ли все степени чётными

int isEulerian = 1;

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

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

isEulerian = 0;

printf("Вершина %d имеет нечётную степень: %d\n", i + 1, degree[i]);

}

}

if (isEulerian) {

printf("Теперь граф может быть Эйлеровым. Все вершины имеют чётные степени.\n");

}

else {

printf("Граф всё ещё не является Эйлеровым. Исправьте степени вершин.\n");

}

break;

}

case 7: {

if (graph == NULL) {

printf("Ошибка: матрица не существует. Сначала создайте или загрузите матрицу.\n");

break;

}

printf("Введите название файла для сохранения матрицы и результата обхода:\n");

char filename[100];

scanf\_s("%s", filename, (unsigned)\_countof(filename));

FILE\* file;

fopen\_s(&file, filename, "w");

if (file == NULL) {

printf("Ошибка: не удалось открыть файл для записи.\n");

break;

}

// Сохраняем размер матрицы

fprintf(file, "Размер матрицы: %d\n", n);

// Сохраняем матрицу

fprintf(file, "Матрица:\n");

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

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

fprintf(file, "%d ", graph2[i][j]);

}

fprintf(file, "\n");

}

if (result == NULL) {

printf("Ошибка: не был совершен обход.\n");

break;

}

// Сохраняем результат обхода

fprintf(file, "Результат обхода:\n");

for (int i = index - 1; i >= 0; i--) { // Записываем результат в обратном порядке

fprintf(file, "%d ", result[i]);

}

fprintf(file, "\n");

fclose(file);

printf("Матрица и результат обхода успешно сохранены в файл '%s'.\n", filename);

break;

}

default: {

printf("Неправильный выбор:\n");

break;

}

}

}

return 0;

}