

**Содержание**

[Реферат 5](#_Toc154464772)

[Введение 6](#_Toc154464773)

[1. Постановка задачи 7](#_Toc154464774)

[2. Теоретическая часть задания 8](#_Toc154464775)

[3. Описание алгоритма программы 9](#_Toc154464776)

[4. Описание программы 10](#_Toc154464777)

[5. Тестирование 18](#_Toc154464778)

[6. Ручной расчет задачи 21](#_Toc154464779)

[Заключение 23](#_Toc154464780)

[Список литературы 24](#_Toc154464781)

[Приложение А. Листинг программы. 25](#_Toc154464782)

# Реферат

Отчет 34стр,17 рисунков, 1 таблица, 1 приложение

ГРАФ,ТЕОРИЯ ГРАФОВ, АЛГОРИТМ ФЛОЙДА,ОБХОД ГРАФА

Цель исследования – разработка программы, способной эффективно находить кратчайшие расстояния между каждой парой вершин графа с использованием алгоритма Флойда.

В работе рассмотрены основные этапы алгоритма Флойда, предназначенного для нахождения кратчайших путей в взвешенном ориентированном или неориентированном графе.

# Введение

Алгоритм Флойда-Уоршелла - это классический метод поиска кратчайших путей между всеми вершинами графа, представляющий из себя динамическое программирование, способное работать со взвешенными графами.

В отличие от алгоритма Дейкстры, который работает только с положительными весами рёбер, алгоритм Флойда-Уоршелла поддерживает графы с рёбрами произвольных весов. Результатом работы алгоритма является матрица кратчайших расстояний между всеми парами вершин.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio2022, язык программирования – Си.

Целью данной курсовой работы является разработка программы на языке Си, который является широко используемым. Именно с его помощью в данном курсовом проекте реализуется алгоритм поиска в глубину, осуществляющий поиск компоненты сильной связности орграфа.

# Постановка задачи

Требуется разработать программу, которая найдет кратчайшие расстояния между всеми вершинами, используя алгоритм Флойда.

Исходный граф в программе должен задаваться матрицей весов ребер,причѐм при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации графа. После обработки этих данных на экран должна выводиться матрица кратчайших расстояний между всеми парами вершин. Необходимо предусмотреть различные исходы, такие как некорректный ввод данных, чтобы программа не выдавала ошибок и работала правильно.Устройство ввода – клавиатура и мышь.

# 

# Теоретическая часть задания

Граф, изображенный на рисунке 1, задается множеством вершин 0,1,…,n. и множеством ребер, соединяющих между собой определенные вершины. Каждое ребро в этом графе имеет свой вес, который представляет стоимость прохождения между соответствующими вершинами. Такой граф называется взвешенным.

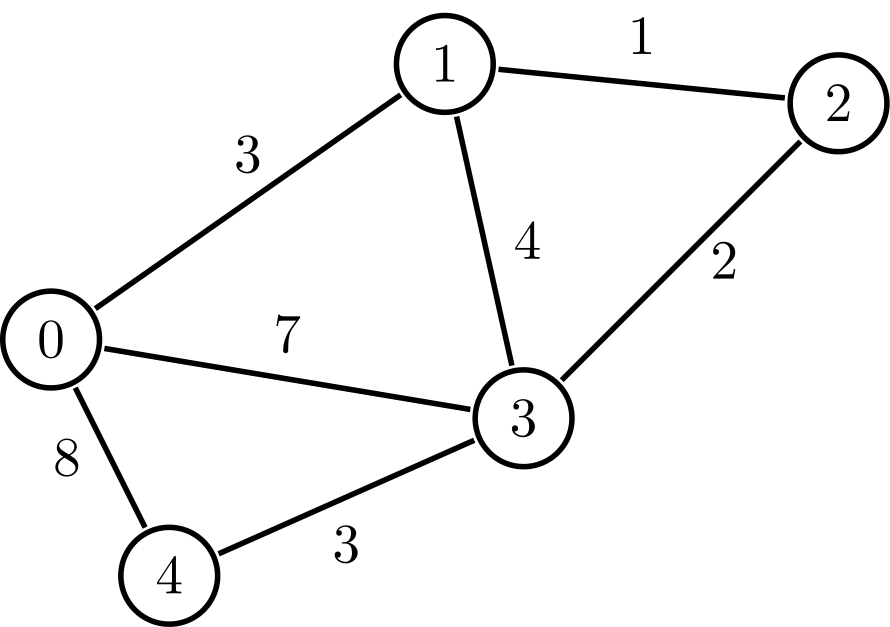


Рисунок 1 – Пример взвешенного графа

Представление графа в виде матрицы смежности позволяет хранить информацию о ребрах графа в квадратной матрице, где значение элемента матрицы указывает на вес ребра между соответствующими вершинами.

Алгоритм Флойда позволяет найти минимальные пути между всеми парами вершин в графе. Он работает путем итеративного обновления матрицы расстояний. Начиная с исходной матрицы смежности, алгоритм постепенно обновляет значения элементов матрицы, находя более короткие пути через другие вершины. Процесс продолжается до тех пор, пока все возможные пути не будут исследованы и найдены минимальные пути между всеми парами вершин в графе.

# Описание алгоритма программы

Алгоритм состоит из трех вложенных циклов. Внешний цикл выполняется для каждой вершины k и представляет собой промежуточную вершину между i и j. Внутренние циклы выполняются для каждой пары вершин i и j и обновляют значение в матрице смежности, если найден более короткий путь через вершину k.

Каждая итерация внутренних циклов проверяет, может ли путь от i до j быть улучшен, проходя через вершину k. Если сумма расстояний от i до k и от k до j меньше текущего расстояния от i до j, то значение в матрице обновляется.

После завершения алгоритма, матрица смежности будет содержать кратчайшие расстояния между каждой парой вершин.

Ниже представлен псевдокод функции алгоритма Флойда:

для k = 0 пока k < size делать k++

для i = 0 пока i < size делать i++

для j = 0 пока j < size делать j++

если matrix[i][k] + matrix[k][j] < matrix[i][j]

matrix[i][j] = matrix[i][k] + matrix[k][j]

# 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 Win64 (Visual C++).

Работа программы начинается с вывода общей информации о

курсовой работе (рис. 2) и (рис 3).

printf("Курсовая работа\n");

printf("По курсу\"Логика и основы алгоритмизации в инженерных задачах\"\n");

printf("На тему\"Реализация алгоритма Флойда\"\n");

Sleep(5000);

system("cls");

printf("Выполнила: Широкова Ирина Дмитриевна\n");

printf("Группа: 22ВВП2\n");

Sleep(5000);

system("cls");

Далее выводится основное меню программы (рис. 4)

printf("Меню\n");

printf("1. Алгоритм Флойда\n");

printf("2. Выход\n");

printf("Введите номер вашего варианта:");

if (scanf("%d", &nomer1) != 1) {

printf("Ошибка: введите пожалуйста номер пункта меню\n");

while (getchar() != '\n'); // очищаем буфер ввода

continue;

}

if (nomer1 < 1 || nomer1 > 3) {

printf("Ошибка: выбран некорректный пункт меню\n");

while (getchar() != '\n'); // очищаем буфер ввода

continue;

}

Если пользователем был выбран первый пункт, программа просит выбрать способ заполнения матрицы значениями(рис. 5).

При выборе автоматической генерации также предлагается выбрать ориентированный или неориентированный граф (рис. 6) и ввести размер матрицы (рис. 7)

Созданная матрица заполняется случайными значениями и выводится на экран при помощи функций **create\_matrix, fillMatrixRandom, print\_matrix.** (рис. 8)

При помощи функции **floydWarshall** находятся кратчайшие расстояния между всеми парами вершинами. (рис. 9)

В конце программы пользователю предлагается сохранить матрицу кратчайших расстояний в файл (рис. 10)

switch (nomer2) {

case 1: {

printf("1.Ориентированный граф\n");

printf("2.Неориентированный граф\n");

printf("Введите номер вашего варианта:");

if (scanf("%d", &nomer3) != 1) {

printf("Ошибка: введите пожалуйста номер пункта меню\n");

while (getchar() != '\n');

continue;

}

if (nomer3 < 1 || nomer3 > 2) {

printf("Ошибка: выбран некорректный пункт меню\n");

while (getchar() != '\n');

continue;

}

system("cls");

switch (nomer3) {

case 1:

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

fillMatrixRandom(adjacencyMatrix, size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

case 2: {

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

fillMatrixRandomNeor(adjacencyMatrix, size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

}

break;

}

При выборе заполнения графа вручную пользователю также предлагается ввести размер и построчно заполнить матрицу значениями(рис. 11)

case 2: {

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

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

printf("Введите элементы %d строки матрицы смежности:\n", i);

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

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

}

}

system("cls");

printf("Ваша матрица:\n ");

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

При выборе загрузки графа из матрицы пользователю предлагается ввести имя файла для считывания. Затем на экран выводится исходная матрица и матрица кратчайших расстояний.(рис. 12)

case 3: {

int adjacencyMatrix[SIZE][SIZE];

int size;

readGraphFromFile(adjacencyMatrix, &size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

Ниже представлена работа программы:

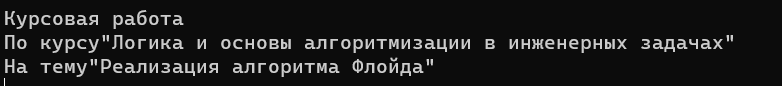


Рисунок 2 – Общая информация\_1



Рисунок 3 – Общая информация\_2

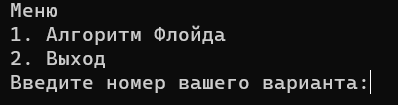


Рисунок 4 – Меню программы\_1

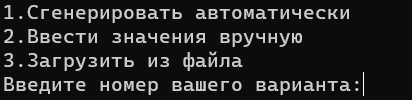


Рисунок 5 – Меню программы\_2

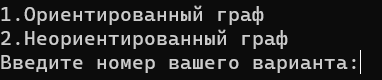


Рисунок 6 – Меню программы\_3



Рисунок 7 – Меню программы\_4

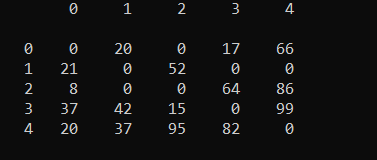
****

Рисунок 8 – Матрица расстояний

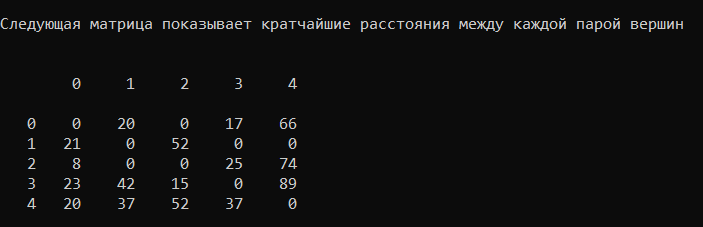
****

Рисунок 9 – Матрица кратчайших расстояний

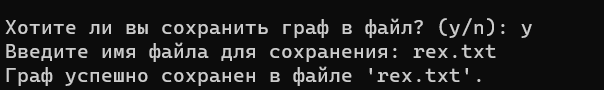
****

Рисунок 10 – Сохранение в файл

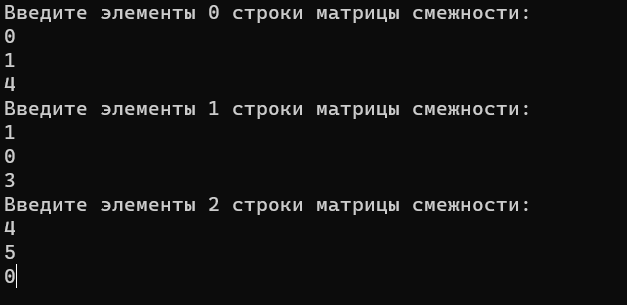
****

Рисунок 11 – Ручной ввод значений

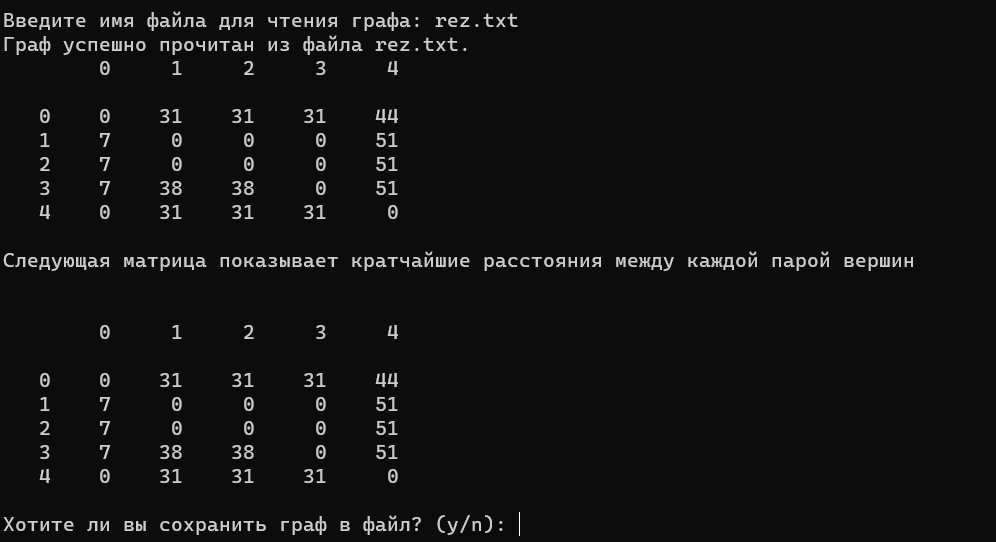
****

Рисунок 12 – Загрузка графа из файла

# Тестирование

Среда разработки Microsoft Visual Studio 2022 предоставляет все

средства, необходимые при разработке и отладке многомодульной

программы. Тестирование проводилось в процессе разработки, после

завершения написания программы. В ходе тестирования проблем не было выявлено.

Ниже продемонстрирован результат тестирования программы при

вводе пользователем некорректных данных:

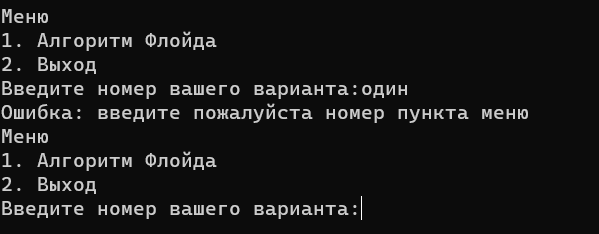
****

Рисунок 13 – Тестирование при неверном выборе пункта меню

****

Рисунок 14 – Тестирование при вводе неверного размера матрицы

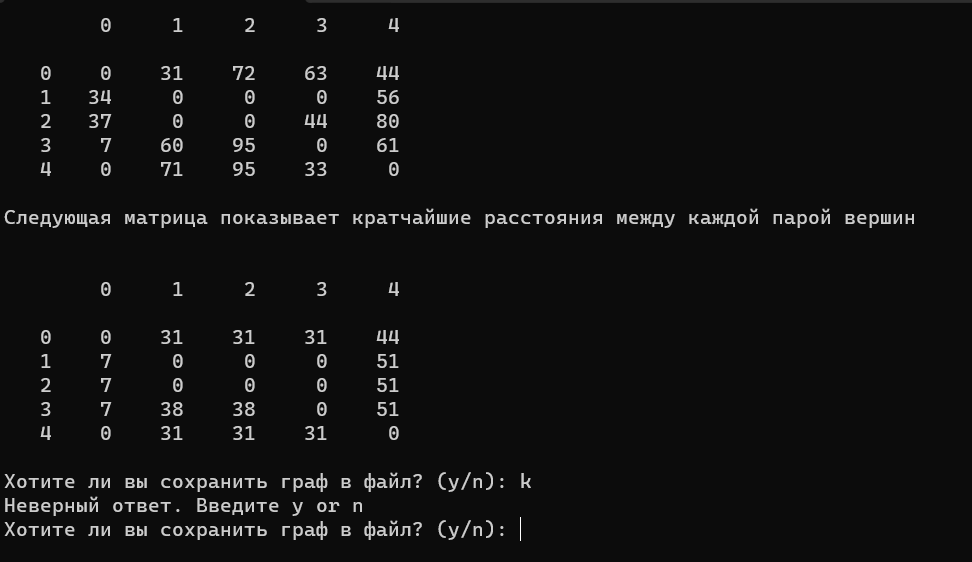
****

Рисунок 15 – Тестирование при неверном выборе сохранения графа в файл

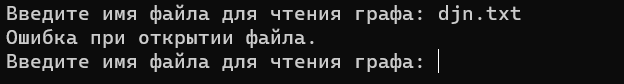
****

Рисунок 16 – Тестирование при открытии несуществующего файла

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Тестирование на неправильный выбор основного пункта меню | Вывод сообщения об ошибке. Предложение повторного ввода | Верно |
| Тестирование на неправильный размер матрицы | Вывод сообщения об ошибке. Предложение повторного ввода | Верно |
| Тестирование при неверном выборе сохранения графа в файл | Вывод сообщения об ошибке. Предложение повторного ввода | Верно |
| Тестирование при открытии несуществующего файла | Вывод сообщения об ошибке. Предложение повторного ввода | Верно |

В результате тестирования было выявлено, что программа успешно

проверяет данные на соответствие необходимым требованиям.

# Ручной расчет задачи

Проведём проверку программы посредством ручных вычислений(рис. 17)

Расстояние от вершины 0 до вершины 1 = 97

Расстояние с промежуточной вершиной от 0 до 2 = 70 и расстояние от 2 до 1 = -9

Получилось расстояние = 61

61<70, поэтому запишем это число в матрицу кратчайших расстояний.

Расстояние от 0 до 2 = 70

от 0 до 1 = 97

от 1 до 2=64

97+64=161>70

Расстояние от 1 до 0 = 75

от 1 до 2 = 64

от 2 до 0 =40

64+40=104>75

Расстояние от 1 до 2 = 64

от 1 до 0 = 75

от 0 до 1 =97

75+97=172>64

Расстояние от 2 до 0 = 40

от 2 до 1 = -9

от 1 до 2 =64

-9+64=55>40

Расстояние от 2 до 1 = -9

от 2 до 0 = 40

от 0 до 1 =97

40+97=137>-9

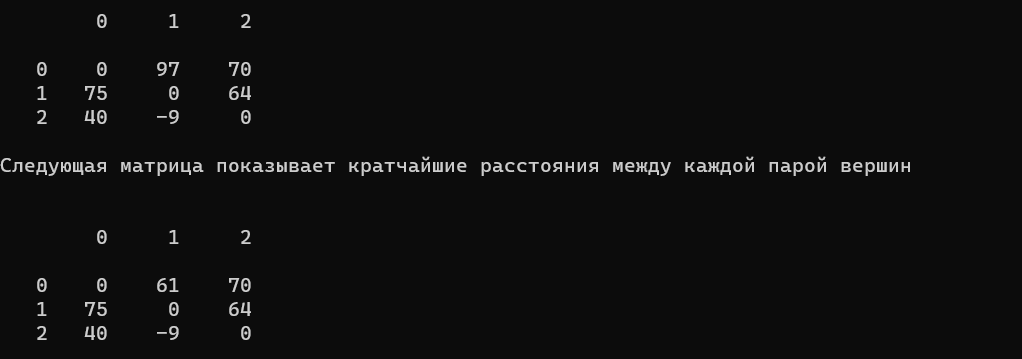
****

Рисунок 17 – Результат проверки

# Заключение

Таким образом, в процессе создания данного проекта была

разработана программа, осуществляющая алгоритм Флойда для поиска кратчайших расстояний между всеми парами вершин при помощи

среды Microsoft Visual Studio 2022.

При выполнении курсовой работы были получены практические

навыки по разработке программня , освоены техники

создания матриц смежности, опирающихся на теорию

графов. Также были углублены знания языка программирования Си.

Программа имеет небольшой, но достаточный для использования

функционал возможностей.

# Список литературы

1. Брайан Керниган, Деннис Ритчи «Язык программирования Си»
2. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977. 208 с.
3. Кормен Т., Лейзерсон Ч., Ривест Р. «Алгоритмы: Построение

и анализ» - М.: МЦНМО, 2001. - 960 с.

1. 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 <stdlib.h>

#include <time.h>

#include <clocale>

#include <Windows.h>

#define SIZE 100

#define MAX\_VERTICES 100

void create\_matrix(int arr[SIZE][SIZE], int size) {

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

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

if (i == j) {

arr[i][j] = 0;

}

}

}

}

void print\_matrix(int arr[SIZE][SIZE], int size) {

printf(" ");

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

printf("%4d ",i );

}

printf("\n");

printf("\n");

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

printf("%4d ", i);

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

printf("%4d ", arr[i][j]);

}

printf("\n");

}

printf("\n");

}

void fillMatrixRandomNeor(int matrix[SIZE][SIZE], int size) {

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

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

if (i == j) {

matrix[i][j] = 0;

}

else {

if (rand() % 5 == 0) {

matrix[i][j] = 0;

matrix[j][i] = 0;

}

else {

int randomValue = (rand() % 111) - 10;

matrix[i][j] = randomValue;

matrix[j][i] = randomValue;

}

}

}

}

}

void fillMatrixRandom(int matrix[SIZE][SIZE], int size) {

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

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

if (i == j) {

matrix[i][j] = 0; // На диагонали всегда 0

}

else {

// Выставляем вероятность для нулевого значения, чтобы сделать граф менее связанным

if (rand() % 5 == 0) {

matrix[i][j] = 0;

}

else {

matrix[i][j] = (rand() % 111) - 10;

}

}

}

}

}

void floydWarshall(int matrix[SIZE][SIZE], int size) {

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

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

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

if (matrix[i][k] + matrix[k][j] < matrix[i][j]) {

matrix[i][j] = matrix[i][k] + matrix[k][j];

}

}

}

}

printf("Следующая матрица показывает кратчайшие расстояния между каждой парой вершин \n");

printf("\n");

printf("\n");

printf(" ");

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

printf("%4d ", i);

}

printf("\n");

printf("\n");

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

printf("%4d ", i);

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

printf("%4d ", matrix[i][j]);

}

printf("\n");

}

printf("\n");

}

void saveGraphToFile(int matrix[SIZE][SIZE], int size) {

char vibor;

re:

printf("Хотите ли вы сохранить граф в файл? (y/n): ");

scanf(" %c", &vibor);

if (vibor == 'y' || vibor == 'Y') {

char filename[100]; // Создаем массив для хранения имени файла

printf("Введите имя файла для сохранения: ");

scanf("%s", filename);

FILE\* file = fopen(filename, "w");

if (file == NULL) {

perror("Не удалось открыть файл");

goto re;

}

// Сохранение размера графа в файл

fprintf(file, "%d\n", size);

fprintf(file, "\n");

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

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

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

}

fprintf(file, "\n");

}

fclose(file);

printf("Граф успешно сохранен в файле '%s'.\n", filename);

}

else if(vibor == 'n' || vibor == 'N') {

printf("Граф не был сохранен в файл.\n");

}

else {

printf("Неверный ответ. Введите y or n\n");

goto re;

}

}

void readGraphFromFile(int matrix[SIZE][SIZE], int\* size) {

char filename[100];

za:

printf("Введите имя файла для чтения графа: ");

scanf("%s", filename);

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Ошибка при открытии файла.\n");

goto za;

}

if (fscanf(file, "%d", size) != 1) {

printf("Ошибка: не удалось считать размер графа из файла\n");

fclose(file);

goto za;

}

if (\*size <= 0 || \*size > SIZE) {

printf("Ошибка: некорректный размер графа в файле\n");

fclose(file);

goto za;

}

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

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

if (fscanf(file, "%d", &matrix[i][j]) != 1) {

printf("Ошибка: не удалось считать матрицу смежности из файла\n");

fclose(file);

goto za;

}

}

}

fclose(file);

printf("Граф успешно прочитан из файла %s.\n", filename);

}

int main() {

setlocale(LC\_ALL, "RUS");

int nomer1;

int nomer2;

int nomer3;

int size;

printf("Курсовая работа\n");

printf("По курсу\"Логика и основы алгоритмизации в инженерных задачах\"\n");

printf("На тему\"Реализация алгоритма Флойда\"\n");

Sleep(5000);

system("cls");

printf("Выполнила: Широкова Ирина Дмитриевна\n");

printf("Группа: 22ВВП2\n");

Sleep(5000);

system("cls");

while (1) {

printf("Меню\n");

printf("1. Алгоритм Флойда\n");

printf("2. Выход\n");

printf("Введите номер вашего варианта:");

if (scanf("%d", &nomer1) != 1) {

printf("Ошибка: введите пожалуйста номер пункта меню\n");

while (getchar() != '\n'); // очищаем буфер ввода

continue;

}

if (nomer1 < 1 || nomer1 > 3) {

printf("Ошибка: выбран некорректный пункт меню\n");

while (getchar() != '\n'); // очищаем буфер ввода

continue;

}

system("cls");

switch(nomer1) {

case 1: {

printf("1.Сгенерировать автоматически\n");

printf("2.Ввести значения вручную\n");

printf("3.Загрузить из файла\n");

printf("Введите номер вашего варианта:");

if (scanf("%d", &nomer2) != 1) {

printf("Ошибка: введите пожалуйста номер пункта меню\n");

while (getchar() != '\n');

continue;

}

if (nomer2 < 1 || nomer2 > 3) {

printf("Ошибка: выбран некорректный пункт меню\n");

while (getchar() != '\n');

continue;

}

system("cls");

switch (nomer2) {

case 1: {

printf("1.Ориентированный граф\n");

printf("2.Неориентированный граф\n");

printf("Введите номер вашего варианта:");

if (scanf("%d", &nomer3) != 1) {

printf("Ошибка: введите пожалуйста номер пункта меню\n");

while (getchar() != '\n');

continue;

}

if (nomer3 < 1 || nomer3 > 2) {

printf("Ошибка: выбран некорректный пункт меню\n");

while (getchar() != '\n');

continue;

}

system("cls");

switch (nomer3) {

case 1: {

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

fillMatrixRandom(adjacencyMatrix, size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

case 2: {

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

fillMatrixRandomNeor(adjacencyMatrix, size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

}

break;

}

case 2: {

printf("Введите размер матрицы смежности от 1 до 100:");

scanf("%d", &size);

if (size <= 0 || size > SIZE) {

printf("Ошибка: количество вершин должно быть в пределах от 1 до 100\n");

continue;

}

system("cls");

int adjacencyMatrix[SIZE][SIZE];

create\_matrix(adjacencyMatrix, size);

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

printf("Введите элементы %d строки матрицы смежности:\n", i);

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

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

}

}

system("cls");

printf("Ваша матрица:\n ");

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

case 3: {

int adjacencyMatrix[SIZE][SIZE];

int size;

readGraphFromFile(adjacencyMatrix, &size);

print\_matrix(adjacencyMatrix, size);

floydWarshall(adjacencyMatrix, size);

saveGraphToFile(adjacencyMatrix, size);

break;

}

}

break;

}

case 2: {

break;

}

}

}

return 0;

}