Министерство 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 Флойда»

Выполнил:

Студент группы 22ВВВ2

Кондратьева В.И.

Принял:

Митрохин М.А.

Пенза 2023

**Содержание**

[Реферат 5](#_Toc153301077)

[Введение 6](#_Toc153301078)

[1 Постановка задачи 7](#_Toc153301079)

[2 Теоретическая часть 8](#_Toc153301080)

[3 Описание алгоритма программы 9](#_Toc153301081)

[4 Описание программы 10](#_Toc153301082)

[5.Тестирование программы 20](#_Toc153301083)

[6 Ручной расчет задачи 26](#_Toc153301084)

[Заключение 28](#_Toc153301085)

[Список литературы 29](#_Toc153301086)

[Приложение А. Листинг 30](#_Toc153301087)

# **Реферат**

Отчет 35 страниц ,17 рисунков

ГРАФ, ТЕОРИЯ ГРАФОВ, АЛГОРИТМ ФЛОЙДА-УОРШЕЛЛА, АЛГОРИТМ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ

Цель исследования – разработка программы, способная рассчитать кратчайший путь между вершинами графа.

В работе используется алгоритм Флойда-Уоршелла. С его помощью можно найти кратчайший путь между вершинами графа.

# **Введение**

В математической теории графов и информатике граф — это совокупность непустого множества вершин и множества пар вершин. Объекты представляются как вершины, или узлы графа, а связи — как дуги, или рёбра. Для разных областей применения виды графов могут различаться направленностью, ограничениями на количество связей и дополнительными данными о вершинах или рёбрах.

Многие структуры, представляющие практический интерес в математике и информатике, могут быть представлены графами.

Математические модели в виде графов широко используются при моделировании разнообразных явлений, процессов и систем. Многие теоретические и реальные прикладные задачи могут быть решены при помощи тех или иных процедур анализа графовых моделей. В качестве примера можно привести задачу составления маршрута движения транспорта между различными городами при заданном расстоянии между населенными пунктами.

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

В данной работе рассматривается алгоритм Флойда — Уоршелла — алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом.

# **1 Постановка задачи**

Требуется разработать программу, которая реализует поиск кратчайших путей во взвешенном ориентированном графе с помощью алгоритма Флойда-Уоршелла. Исходный граф в программе должен задаваться матрицей смежности.

Программа должна содержать в себе:

- текстовое или графическое меню;

- возможность задания пользователем размера графа;

- возможность выбора автоматического или ручного задания графа;

- возможность сохранения результатов работы программы.

# **2 Теоретическая часть**

Обозначим длину кратчайшего пути между вершинами *u* и *v*, содержащего, помимо *u* и *v*, только вершины из множества:

На каждом шаге алгоритма, мы будем брать очередную вершину (пусть её номер — i) и для всех пар вершин u и v вычислять:

То есть, если кратчайший путь из u в v, содержащий только вершины из множества {1..i} , проходит через вершину *i*, то кратчайшим путем из *u* в *v* является кратчайший путь из u в *i*, объединенный с кратчайшим путем из *i* в *v*. В противном случае, когда этот путь не содержит вершины *i*, кратчайший путь из *u* в *v*, содержащий только вершины из множества {1..*i*} является кратчайшим путем из *u* в *v*, содержащим только вершины из множества {1..*i*-1}

В итоге получаем, что матрица и является искомой матрицей кратчайших путей, поскольку содержит в себе длины кратчайших путей между всеми парами вершин, имеющих в качестве промежуточных вершин вершины из множества {1...n}, что есть попросту все вершины графа. Такая реализация работает за Q(n3) времени и использует Q(n3) памяти.

# **3 Описание алгоритма программы**

Для реализации алгоритма используется функция alg (int si, int\*\* a), в которую передаётся массив случайных значений или значений, полученных с помощью ручного ввода.

В функции alg (int si, int\*\* a) с помощью вложенных циклов производится поиск кратчайших путей. Если соблюдается условие a[i][j] > a[i][k] + a[k][j], то в a[i][j] записывается сумма a[i][k] + a[k][j]. Здесь переменная k соответствует промежуточной вершине. Например, если необходимо попасть из вершины «2» в вершину «3», то это можно сделать как напрямую, так и через некую вершину «k». После осуществления алгоритма получается матрица, в которой каждое значение a[i][j] обозначает кратчайшее расстояние из пункта i в пункт j. Сложность алгоритма: О(n3).

Ниже представлен псевдокод функции alg (int si, int\*\* a):

1.Для k=0 пока k < si делать k++;

2.Для i=0 пока i < si делать i++;

3.Для j=0 пока j < si делать j++;

4. Если a[i][j] > a[i][k] + a[k][j]

5. a[i][j] = a[i][k] + a[k][j]

Полный код программы можно увидеть в Приложении А.

# **4 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: alg, shapka.

При запуске программы на экран выводится меню. При помощи функции \_getch() библиотеки conio.h вводится значение.

1 - Неориентированный граф (ручной ввод), 2 - Неориентированный граф (случайные значения), 3 - Ориентированный граф (ручной ввод), 4 - Ориентированный граф (случайные значения), ecs – выход.

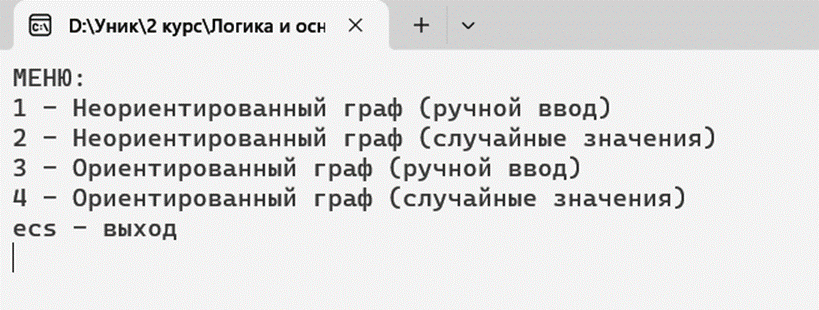
****

Рисунок 1 – Меню программы

При нажатии клавиши «1» пользователю предлагается ввести размер матрицы. Далее происходит генерация матрицы. Пользователю предлагается заполнить значения выше главной диагонали. Если введённое значение не попадает в заданный диапазон, то пользователю предлагается ввести другое значение. Значения ниже главной диагонали автоматически заполняются симметрично введенным, главная диагональ заполняется нулями. После чего полученная матрица выводится на экран и записывается в файл «графы.txt». При этом все введенные значения, равные 10000 выводятся как «-». Ниже представлен кусок кода для задания неориентированного графа (ручной ввод).

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

if (i < j) { //если выше главной диагонали

int a1 = i + 1, a2 = j + 1;

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

while ((chi < 1) || (chi >= 1000)) {

if (chi == 10000) {

break;

}

printf("Ваше число не входит в диапазон от 1 до 999\n\n");

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

}

arr[i][j] = chi;

}

if (i > j) { //если ниже главной диагонали

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

}

}

}

Затем выполняется алгоритм Флойда-Уоршелла – функция alg, в которую передаётся размер матрицы size и массив arr. Об алгоритме работы функции alg подробно написано в разделе 3.

После этого выполняется функция shapka, в которую передаётся размер массива, матрица значений, массив для вывода шапки значений, файл. В данной функции задаётся и выводится шапка таблицы с номерами вершин, а также сама матрица. При этом все введенные значения, равные 10000 выводятся как «-». Ниже представлен кусок кода для функции shapka.

void shapka(int si, int\*\* a, int\* m, FILE\* F) {

//printf("%d", si);

for (int i = 0; i < si; i++) { //задание шапки таблицы

m[i] = i + 1;

}

fprintf(F, " "); //вывод шапки таблицы

printf(" ");

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

fprintf(F, "%7d", m[i]);

printf("%7d", m[i]);

}

fprintf(F, "\n");

printf("\n");

for (int i = 0; i < si \* 7 + 6; i++) {

fprintf(F, "-");

printf("-");

}

fprintf(F, "\n");

printf("\n");

for (int i = 0; i < si; i++) { //цикл для вывода массива

fprintf(F, "%d|", m[i]); //вывод первого столбца

printf("%d|", m[i]);

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

if (a[i][j] >= 10000) {

fprintf(F, " -");

printf(" -");

}

else {

fprintf(F, "%7d", a[i][j]);

printf("%7d", a[i][j]);

}

}

fprintf(F, "\n");

printf("\n");

}

fprintf(F, "\n\n");

printf("\n\n");

}

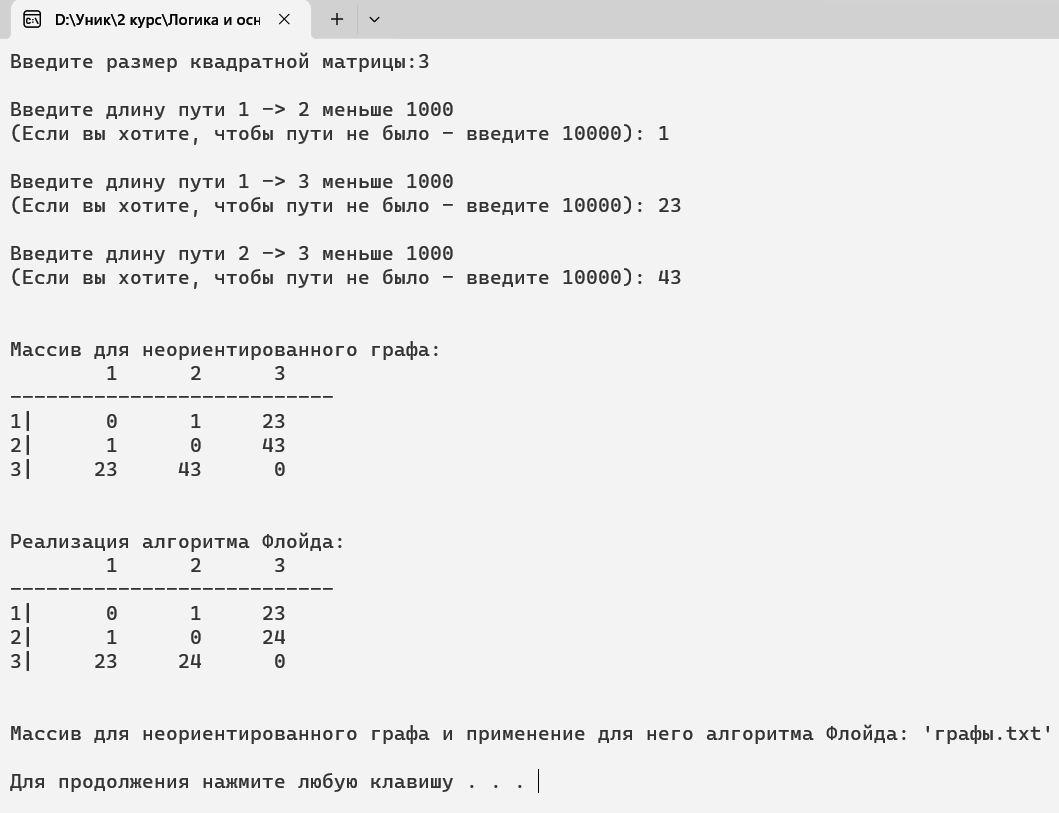


Рисунок 2 – Задание неориентированного графа (ручной ввод)

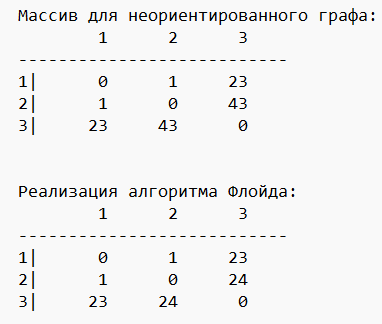


Рисунок 3 – Массивы для неориентированного графа (ручной ввод) до и после осуществление алгоритма

При нажатии клавиши «2» пользователю предлагается ввести размер матрицы. Далее происходит генерация матрицы. Программа генерирует случайные значения выше главной диагонали. Значения ниже главной диагонали автоматически заполняются симметрично введенным. Для генерации случайных значений берется остаток от деления на 120 случайного числа rand (). К числу прибавляется единица, чтобы не было нулей. Если получившееся число больше сотни, то данному элементу матрицы присваивается 10000. Главная диагональ заполняется нулями. После чего полученная матрица выводится на экран и записывается в файл «графы.txt». При этом все введенные значения, равные 10000 выводятся как «-». Ниже представлен кусок кода для задания неориентированного графа (ручной ввод).

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

if (i < j) { //если выше главной диагонали

arr[i][j] = rand() % 120 + 1;

if (arr[i][j] >= 100){

arr[i][j] = 10000;

}

}

if (i > j) { //если ниже главной диагонали

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

}

}

}

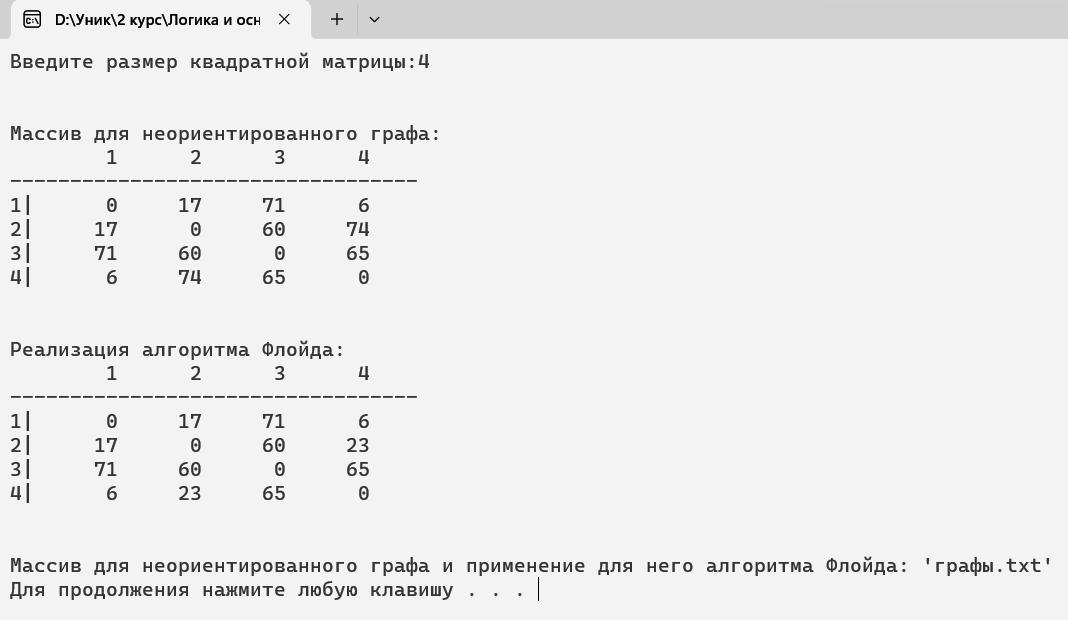


Рисунок 4 – Задание неориентированного графа (случайные значения)

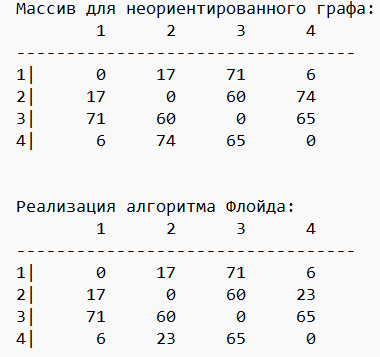


Рисунок 5 – Массивы для неориентированного графа (случайные значения) до и после осуществление алгоритма

При нажатии клавиши «3» пользователю предлагается ввести размер матрицы. Далее происходит генерация матрицы. Главная диагональ заполняется нулями, остальные значения вводятся пользователем. Если введённое значение не попадает в заданный диапазон, то пользователю предлагается ввести другое значение. После чего полученная матрица выводится на экран и записывается в файл «графы.txt». При этом все введенные значения, равные 10000 выводятся как «-». Ниже представлен кусок кода для задания ориентированного графа (ручной ввод).

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

else{

int a1 = i + 1, a2 = j + 1;

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

while ((chi < 1) || (chi >= 1000)) {

printf("Ваше число не входит в диапазон от 1 до 999\n\n");

if (chi == 10000) {

break;

}

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

}

arr[i][j] = chi;

}

}

}

Затем выполняется алгоритм Флойда-Уоршелла – функция alg. После этого выполняется функция shapka.

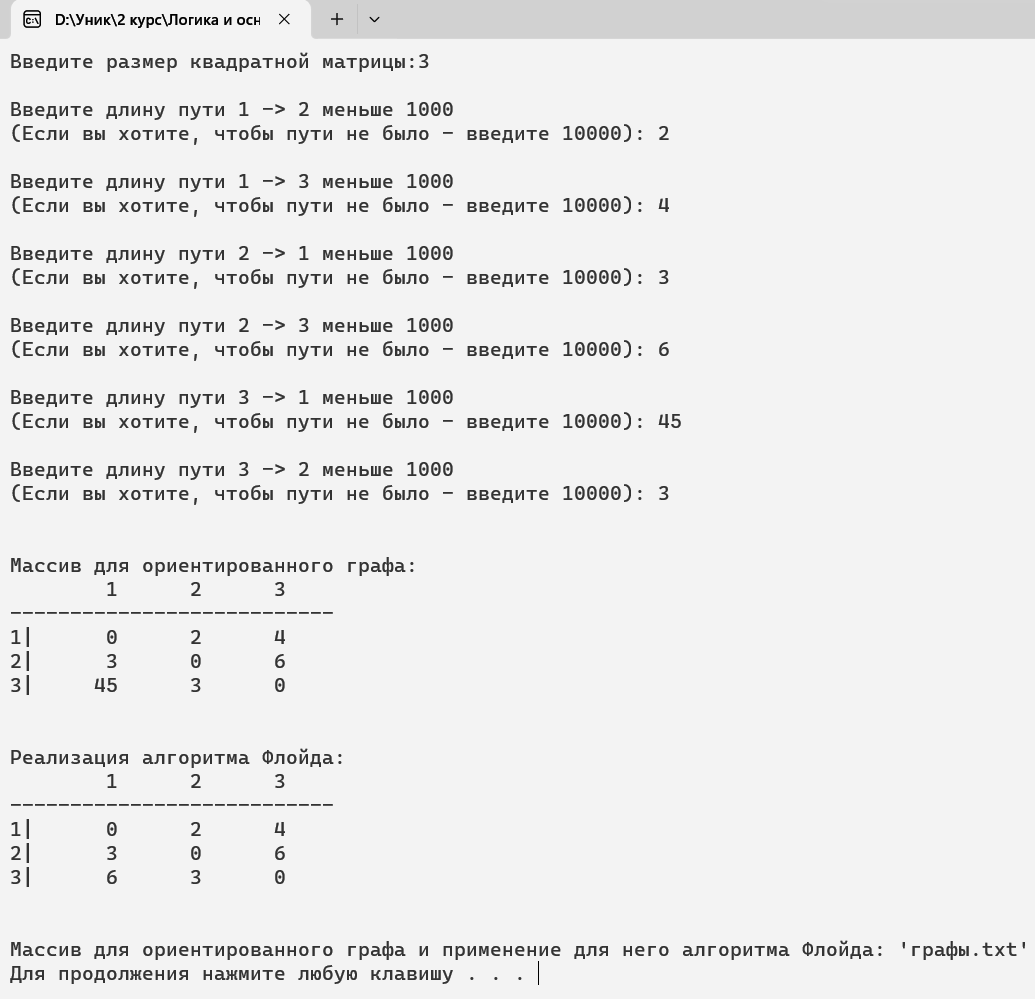


Рисунок 6 – Задание ориентированного графа (ручной ввод)

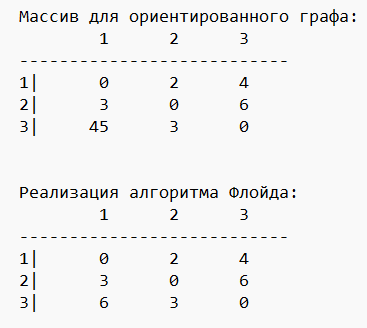


Рисунок 7 – Массивы для ориентированного графа (ручной ввод) до и после осуществление алгоритма

При нажатии клавиши «4» пользователю предлагается ввести размер матрицы. Далее происходит генерация матрицы. Для генерации случайных значений берется остаток от деления на 120 случайного числа rand (). К числу прибавляется единица, чтобы не было нулей. Если получившееся число больше сотни, то данному элементу матрицы присваивается 10000. Главная диагональ заполняется нулями. После чего полученная матрица выводится на экран и записывается в файл «графы.txt». При этом все введенные значения, равные 10000 выводятся как «-». Ниже представлен кусок кода для задания неориентированного графа (ручной ввод).

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

else{ //если выше главной диагонали

arr[i][j] = rand() % 120 + 1;

if (arr[i][j] >= 100) {

arr[i][j] = 10000;

}

}

}

}

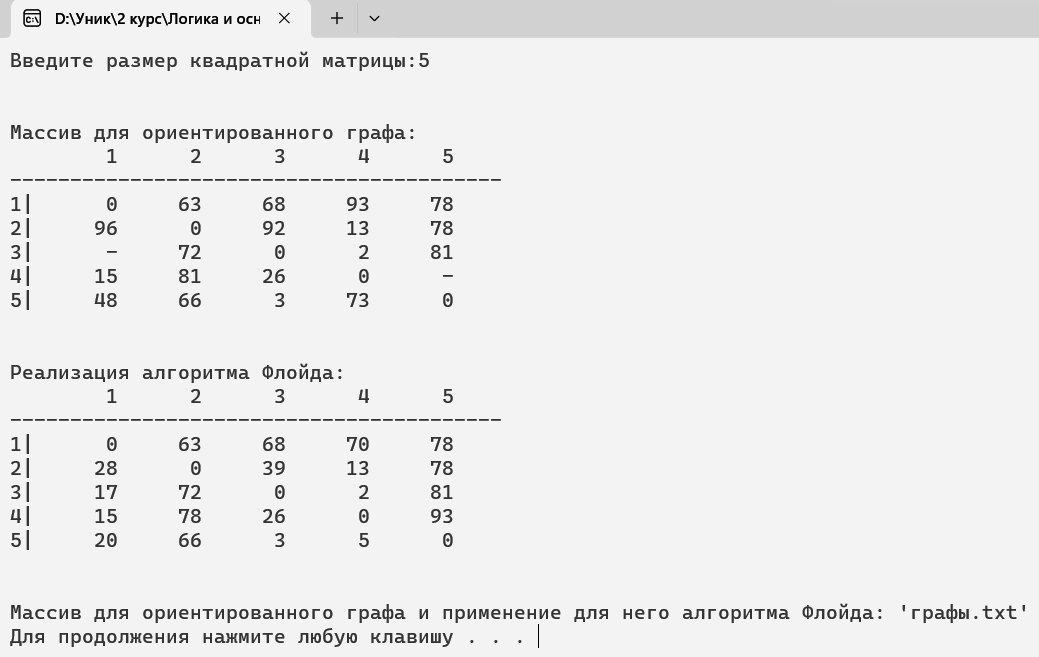


Рисунок 8 – Задание ориентированного графа (случайные значения)



Рисунок 9 – Массивы для ориентированного графа (случайные значения) до и после осуществление алгоритма

# **5 Тестирование программы**

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.

Проверка на попадание в диапазон значений при ручном вводе:

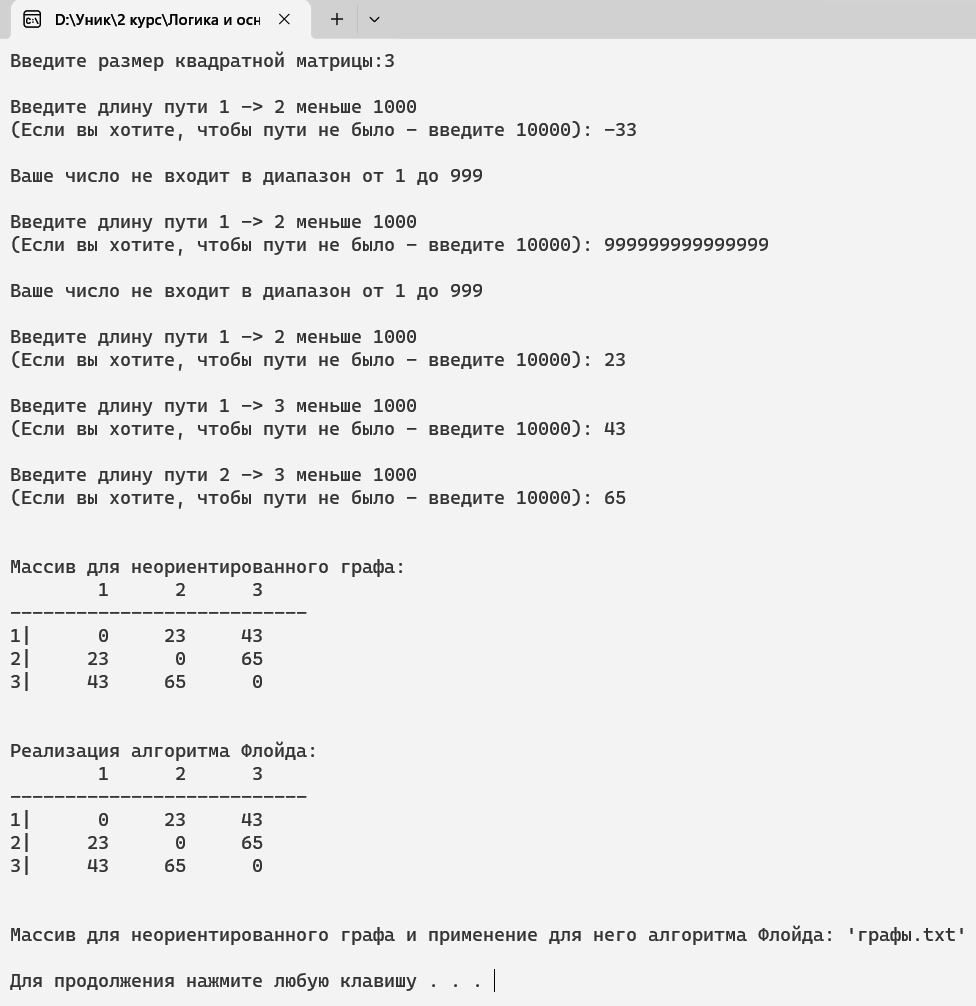


Рисунок 10 – Проверка диапазона значений

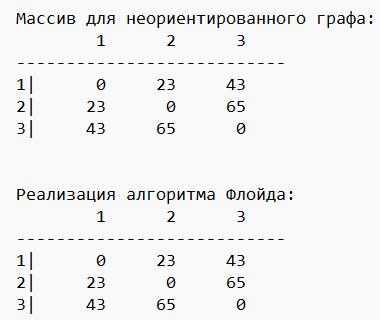


Рисунок 11 – Проверка диапазона значений

Проверка замены «10000» на «-»:

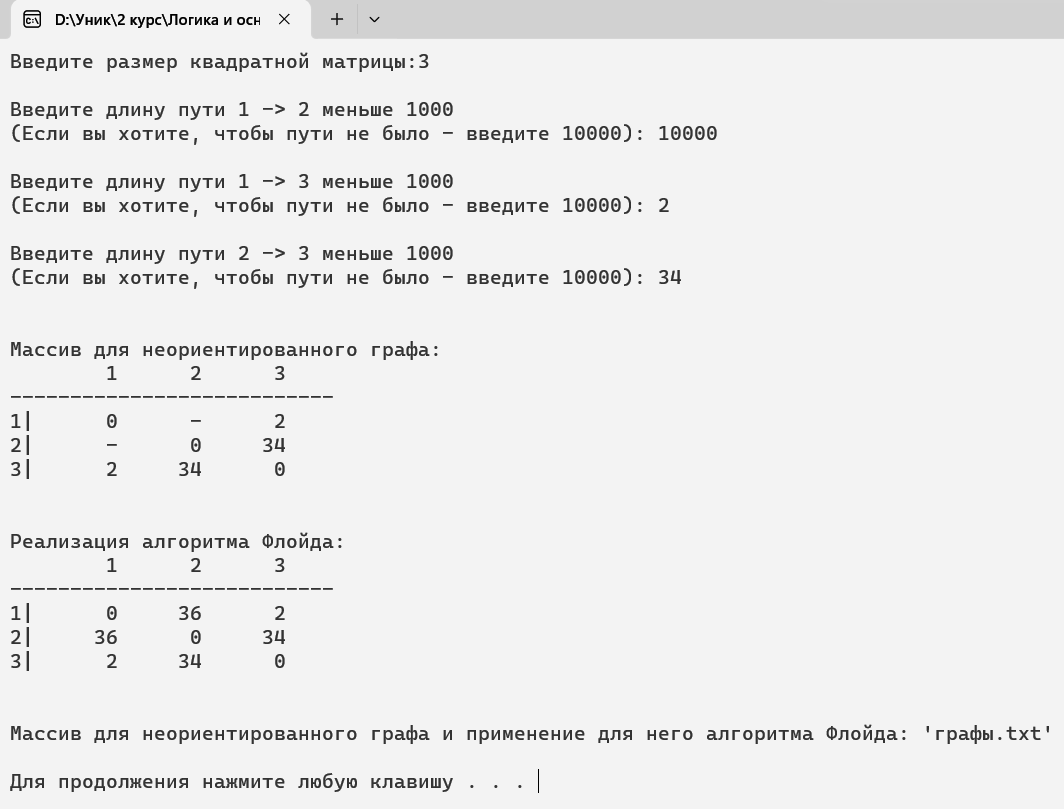


Рисунок 12 – Пользователь вводит в матрицу число «10000»

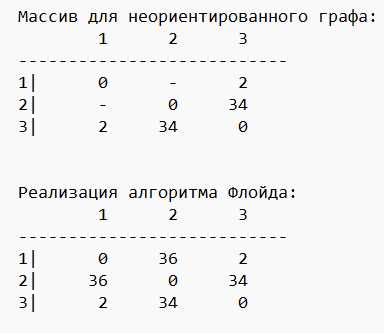


Рисунок 13 – Замена «10000» на «-» при выводе

Тестирование на заполнение массива случайными значениями:

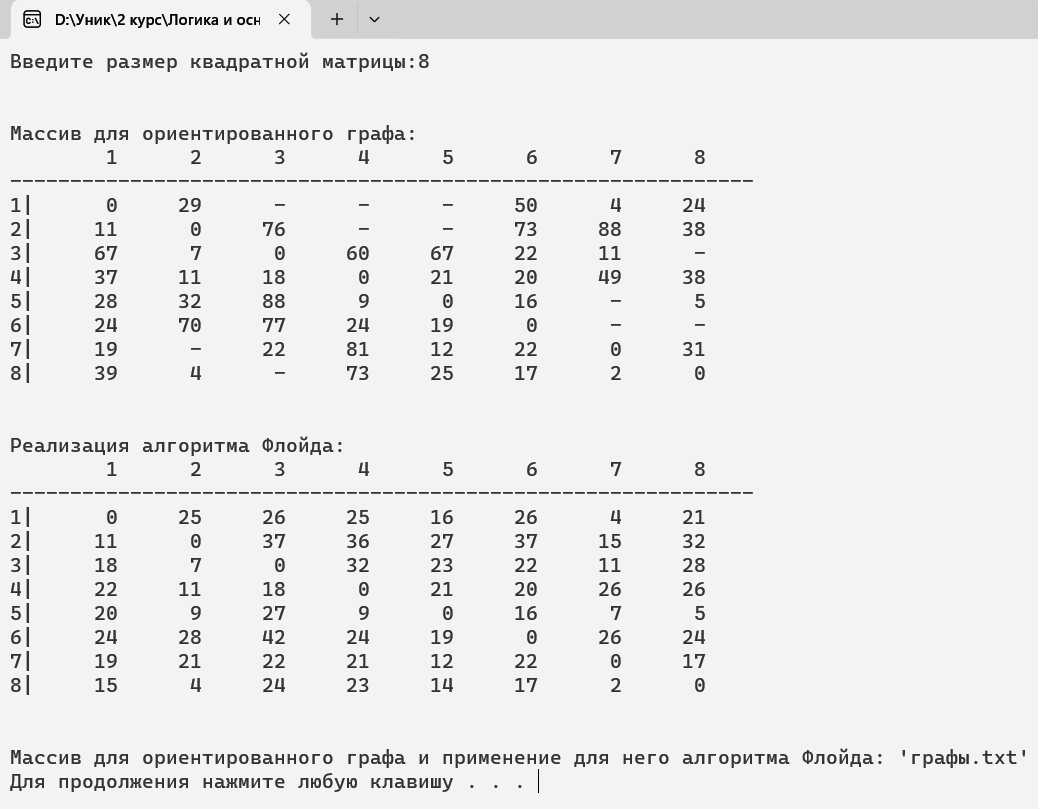


Рисунок 14 – Создание массива случайных значений

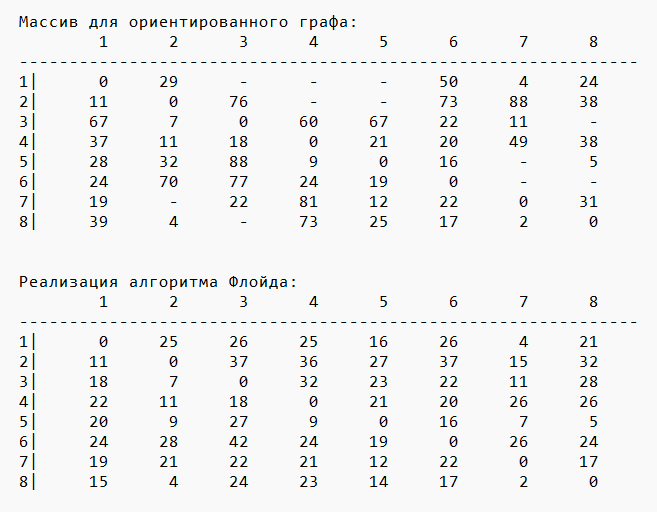


Рисунок 15 – Массива случайных значений до и после осуществления алгоритма

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод меню | Верно (Рисунок 1) |
| Вывод генерации матрицы | Вывод сообщения о количестве вершин | Верно (Рисунок 10) |
| Проверка диапазона значений | Вывод сообщения о том, что число не входит в диапазон. Требование ввести число, входящее в диапазон | Верно (Рисунок 10, Рисунок 11) |
| Проверка вывода «-» при значении, равном 10000 | Если пользователь вводит в матрицу число 10000, то на экран во время вывода матрицы выводится «-» | Верно (Рисунок 12, Рисунок 13) |
| Заполнение массива случайными значениями | При выводе матрицы все значения разные, есть вершины, между которыми нет пути. | Верно (Рисунок 14, Рисунок 15) |

# **6 Ручной расчет задачи**

Для удобства ручного расчета возьмем граф с количеством вершин равным 5.

Заполняем матрицу случайными значениями, после чего ее считывает программа и выводит результат.

****

Рисунок 16 – Результат работы программы

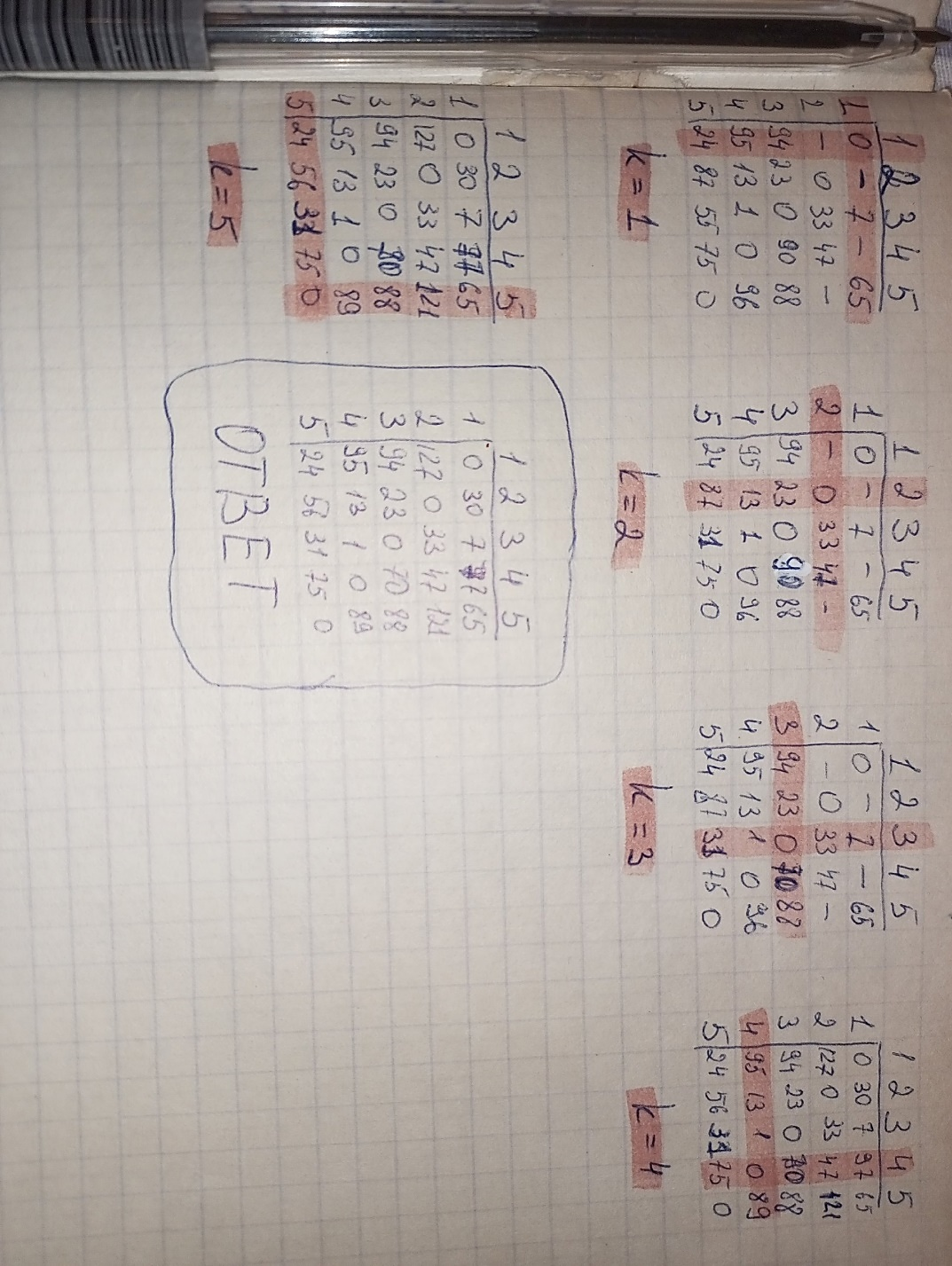


Рисунок 17 – Ручной просчёт программы

Результаты программы и ручного просчёта совпали, следовательно программа работает правильно.

# **Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Флойда-Уоршелла в MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей, основанных на теории орграфов. Приобретены навыки по осуществлению алгоритма поиска кратчайших путей. Углублены знания языка программирования Cи.

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. Уилсон Р. Введение в теорию графов. Пер. с анг. 1977. 208 с.

2. Харви Дейтел, Пол Дейтел. Как программировать на C/C++. 2009 г.

3. Оре О. Графы и их применение: Пер. с англ. 1965. 176 с.

4. - <https://habr.com/ru/post/105825/>

5. - <http://mindhalls.ru/floyd-warshall-algorithm/>

# **Приложение А. Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <windows.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

void alg(int si, int\*\* a) {

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

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

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

if (a[i][j] > a[i][k] + a[k][j]) {

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

}

}

}

}

}

void shapka(int si, int\*\* a, int\* m, FILE\* F) {

//printf("%d", si);

for (int i = 0; i < si; i++) { //задание шапки таблицы

m[i] = i + 1;

}

fprintf(F, " "); //вывод шапки таблицы

printf(" ");

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

fprintf(F, "%7d", m[i]);

printf("%7d", m[i]);

}

fprintf(F, "\n");

printf("\n");

for (int i = 0; i < si \* 7 + 6; i++) {

fprintf(F, "-");

printf("-");

}

fprintf(F, "\n");

printf("\n");

for (int i = 0; i < si; i++) { //цикл для вывода массива

fprintf(F, "%d|", m[i]); //вывод первого столбца

printf("%d|", m[i]);

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

if (a[i][j] >= 10000) {

fprintf(F, " -");

printf(" -");

}

else {

fprintf(F, "%7d", a[i][j]);

printf("%7d", a[i][j]);

}

}

fprintf(F, "\n");

printf("\n");

}

fprintf(F, "\n\n");

printf("\n\n");

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

FILE\* f;

char menu;

int size;

int\*\* arr;

int\* arr1;

do {

system("cls");

printf("МЕНЮ:\n");

printf("1 - Неориентированный граф (ручной ввод)\n");

printf("2 - Неориентированный граф (случайные значения)\n");

printf("3 - Ориентированный граф (ручной ввод)\n");

printf("4 - Ориентированный граф (случайные значения)\n");

printf("ecs - выход\n");

menu = \_getch();

switch (menu) {

case '1':

system("cls");

printf("Введите размер квадратной матрицы:");

scanf("%d", &size);

printf("\n");

arr1 = (int\*)malloc(sizeof(int) \* size);

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

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

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

}

int chi;

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

if (i < j) { //если выше главной диагонали

int a1 = i + 1, a2 = j + 1;

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

while ((chi < 1) || (chi >= 1000)) {

if (chi == 10000) {

break;

}

printf("Ваше число не входит в диапазон от 1 до 999\n\n");

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

}

arr[i][j] = chi;

}

if (i > j) { //если ниже главной диагонали

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

}

}

}

printf("\n");

f = fopen("графы.txt", "a");

fprintf(f, "Массив для неориентированного графа:\n");

printf("Массив для неориентированного графа:\n");

shapka(size, arr, arr1, f);

alg(size, arr);

fprintf(f, "Реализация алгоритма Флойда:\n");

printf("Реализация алгоритма Флойда:\n");

shapka(size, arr, arr1, f);

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

fclose(f);

printf("Массив для неориентированного графа и применение для него алгоритма Флойда: 'графы.txt'\n\n");

system("pause");

for(int i = 0; i < size; ++i) free(arr[i]);

free(arr);

free(arr1);

break;

}

switch (menu) {

case '2':

system("cls");

printf("Введите размер квадратной матрицы:");

scanf("%d", &size);

printf("\n");

arr1 = (int\*)malloc(sizeof(int) \* size);

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

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

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

}

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

if (i < j) { //если выше главной диагонали

arr[i][j] = rand() % 120 + 1;

if (arr[i][j] >= 100){

arr[i][j] = 10000;

}

}

if (i > j) { //если ниже главной диагонали

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

}

}

}

printf("\n");

f = fopen("графы.txt", "a");

fprintf(f, "Массив для неориентированного графа:\n");

printf("Массив для неориентированного графа:\n");

shapka(size, arr, arr1, f);

alg(size, arr);

fprintf(f, "Реализация алгоритма Флойда:\n");

printf("Реализация алгоритма Флойда:\n");

shapka(size, arr, arr1, f);

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

fclose(f);

printf("Массив для неориентированного графа и применение для него алгоритма Флойда: 'графы.txt'\n");

system("pause");

for (int i = 0; i < size; ++i) free(arr[i]);

free(arr);

free(arr1);

break;

}

switch (menu) {

case '3':

system("cls");

printf("Введите размер квадратной матрицы:");

scanf("%d", &size);

printf("\n");

arr1 = (int\*)malloc(sizeof(int) \* size);

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

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

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

}

int chi;

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

else{

int a1 = i + 1, a2 = j + 1;

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

while ((chi < 1) || (chi >= 1000)) {

printf("Ваше число не входит в диапазон от 1 до 999\n\n");

if (chi == 10000) {

break;

}

printf("Введите длину пути %d -> %d меньше 1000\n(Если вы хотите, чтобы пути не было - введите 10000): ", a1, a2);

scanf("%d", &chi);

printf("\n");

}

arr[i][j] = chi;

}

}

}

printf("\n");

f = fopen("графы.txt", "a");

fprintf(f, "Массив для ориентированного графа:\n");

printf("Массив для ориентированного графа:\n");

shapka(size, arr, arr1, f);

alg(size, arr);

fprintf(f, "Реализация алгоритма Флойда:\n");

printf("Реализация алгоритма Флойда:\n");

shapka(size, arr, arr1, f);

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

fclose(f);

printf("Массив для ориентированного графа и применение для него алгоритма Флойда: 'графы.txt'\n");

system("pause");

for (int i = 0; i < size; ++i) free(arr[i]);

free(arr);

free(arr1);

break;

}

switch (menu) {

case '4':

system("cls");

printf("Введите размер квадратной матрицы:");

scanf("%d", &size);

printf("\n");

arr1 = (int\*)malloc(sizeof(int) \* size);

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

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

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

}

for (int i = 0; i < size; i++) { //цикл для задания массива

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

if (i == j) { //если главная диагональ

arr[i][j] = 0;

}

else{ //если выше главной диагонали

arr[i][j] = rand() % 120 + 1;

if (arr[i][j] >= 100) {

arr[i][j] = 10000;

}

}

}

}

printf("\n");

f = fopen("графы.txt", "a");

fprintf(f, "Массив для ориентированного графа:\n");

printf("Массив для ориентированного графа:\n");

shapka(size, arr, arr1, f);

alg(size, arr);

fprintf(f, "Реализация алгоритма Флойда:\n");

printf("Реализация алгоритма Флойда:\n");

shapka(size, arr, arr1, f);

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

fclose(f);

printf("Массив для ориентированного графа и применение для него алгоритма Флойда: 'графы.txt'\n");

system("pause");

for (int i = 0; i < size; ++i) free(arr[i]);

free(arr);

free(arr1);

break;

}

} while (menu != 27);

}