Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовой работе

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Реализация алгоритма Флойда — Уоршелла поиска кратчайших путей в графе»

Выполнил:

Студент группы 19ВВ3

Костин Д.А.

Принял:

Митрохин М.А.

Пенза 2022

**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**Факультет Вычислительной техники

Кафедра “Вычислительная техника”

“УТВЕРЖДАЮ”

Зав. кафедрой ВТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Митрохин

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г

**ЗАДАНИЕ**

**на курсовое проектирование по курсу**

**Логика и основы алгоритмизации в инженерных задачах**

**Студенту** Костин Д.А. **Группа** 19ВВ3  
**Тема проекта**: реализация алгоритма Флойда — Уоршелла поиска кратчайших путей в графе ***­***

Исходные данные (технические требования) на проектирование

Разработка алгоритмов и программного обеспечения в соответствии с данными задания курсового проекта

Пояснительная записка должна содержать:

1. Постановку задачи
2. Теоретическую часть задания
3. Описание алгоритма поставленной задачи
4. Пример ручного просчета задачи и вычислений (на небольшом участке работы алгоритма)
5. Описание самой программы
6. Тесты
7. Список литературы
8. Листинг программы
9. Результат работы программы

**Объем работы по курсу**

1. **Расчетная часть**

Ручной расчет работы алгоритма

1. **Графическая часть**

Схема алгоритма в формате блок-схемы

1. **Экспериментальная часть**

Тестирование программы

Результат работы программы на тестовых данных

**Срок выполнения проекта по разделам**

1. Исследование теоретической части
2. Разработка алгоритмов программы
3. Разработка программы
4. Тестирование и завершение разработки программы
5. Оформление пояснительной записки

Дата выдачи задания “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты проекта “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Задание получил** “\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Студент**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Костин Д.А.

**Реферат** 5

**Введение** 6

**Описание алгоритма** 7

**Описание программы** 10

**Тестирование** 16

**Заключение** 18

**Список литературы** 19

**Приложение А** 20

**Приложение В** 23

# **Реферат**

**Отчет 22 стр, 13 рисунков.**

Реализация алгоритма Флойда-Уоршелла.

Цель исследования – разработка программы, выполняющей алгоритм Флойда-Уоршелла.

В работе рассмотрен способ нахождения кратчайших расстояний между всеми вершинами взвешенного графа без циклов с отрицательными весами.

# **Введение**

Алгоритм Флойда (алгоритм Флойда–Уоршелла) — алгоритм нахождения длин кратчайших путей между всеми парами вершин во взвешенном ориентированном графе. Работает корректно, если в графе нет циклов отрицательной величины, а в случае, когда такой цикл есть, позволяет найти хотя бы один такой цикл.

Алгоритм работает за O(n3) времени и использует O(n2) памяти.

Разработан в 1962 году.

Постановка задачи

Требуется разработать программу, которая реализует алгоритм Флойда.

Условный граф должен задаваться матрицей смежности. Программа должна работать так, чтобы у пользователя был выбор самому вводить данные, загрузить из файла или случайно сгенерировать, после обработки данных на экран должна выводиться матрица смежности и результат работы алгоритма Флойда.

Устройство ввода- клавиатура.

# **Описание алгоритма**

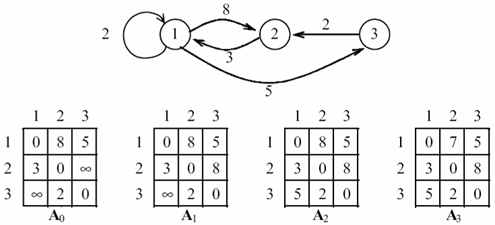
В алгоритме Флойда используется матрица A размером nxn, в которой вычисляются длины кратчайших путей. Элемент A[i,j] равен расстоянию от вершины i к вершине j, которое имеет конечное значение, если существует ребро (i,j), и равен бесконечности в противном случае.

Основная идея алгоритма. Пусть есть три вершины i, j, k и заданы расстояния между ними. Если выполняется неравенство A[i,k]+A[k,j]<A[i,j], то целесообразно заменить путь i->j путем i->k->j. Такая замена выполняется систематически в процессе выполнения данного алгоритма.

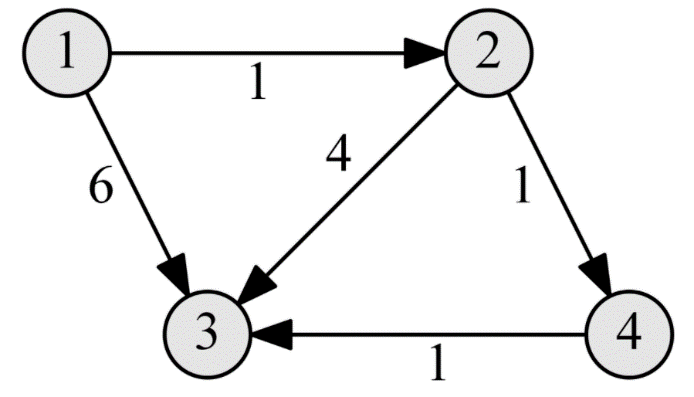
Шаг 0. Определяем начальную матрицу расстояния A0 и матрицу последовательности вершин S0. Каждый диагональный элемент обеих матриц равен 0, таким образом, показывая, что эти элементы в вычислениях не участвуют. Полагаем k = 1.

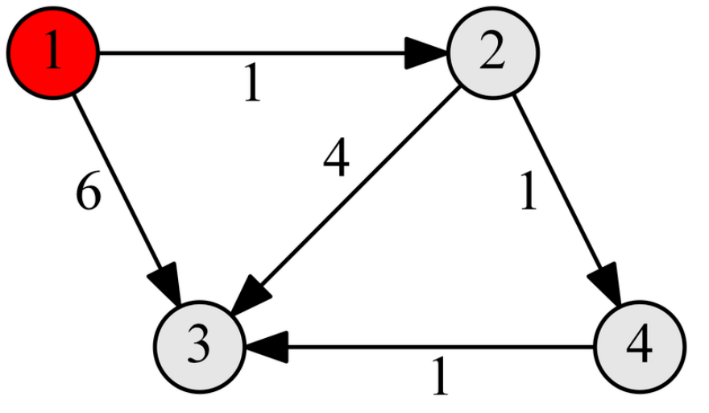
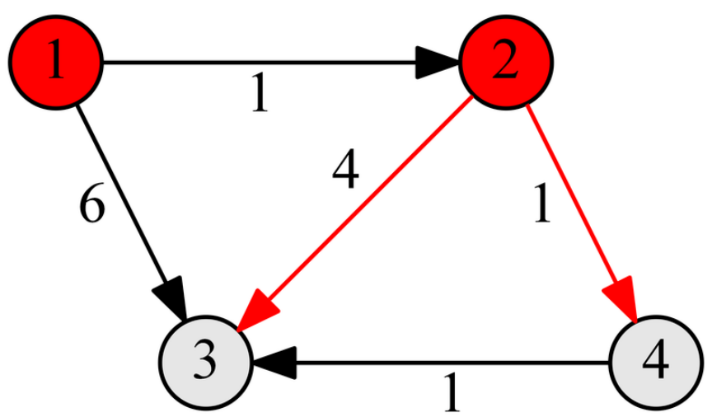
Основной шаг k. Задаем строку k и столбец k как ведущую строку и ведущий столбец. Рассматриваем возможность применения замены описанной выше, ко всем элементам A[i,j] матрицы Ak-1. Если выполняется неравенство A[i,k]+A[k,j]<A[i,j], (i\ne k, j\ne k, i\ne j), тогда выполняем следующие действия:

1. создаем матрицу Ak путем замены в матрице Ak-1 элемента A[i,j] на сумму A[i,k]+A[k,j] ;
2. создаем матрицу Sk путем замены в матрице Sk-1 элемента S[i,j] на k. Полагаем k = k + 1 и повторяем шаг k.

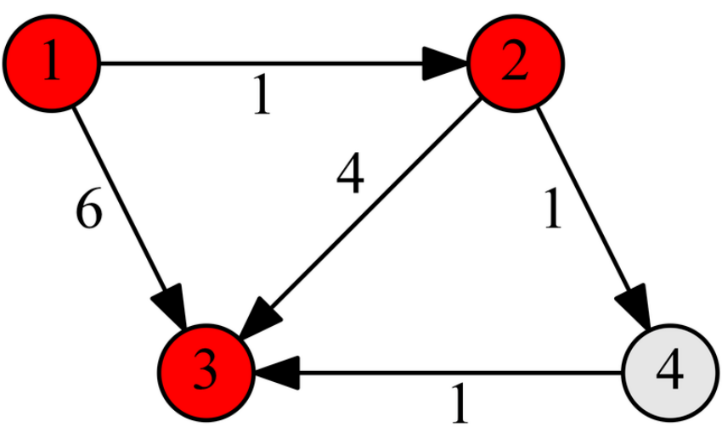
Таким образом, алгоритм Флойда делает n итераций, после i -й итерации матрица А будет содержать длины кратчайших путей между любыми двумя парами вершин при условии, что эти пути проходят через вершины от первой до i -й. На каждой итерации перебираются все пары вершин и путь между ними сокращается при помощи i -й вершины.

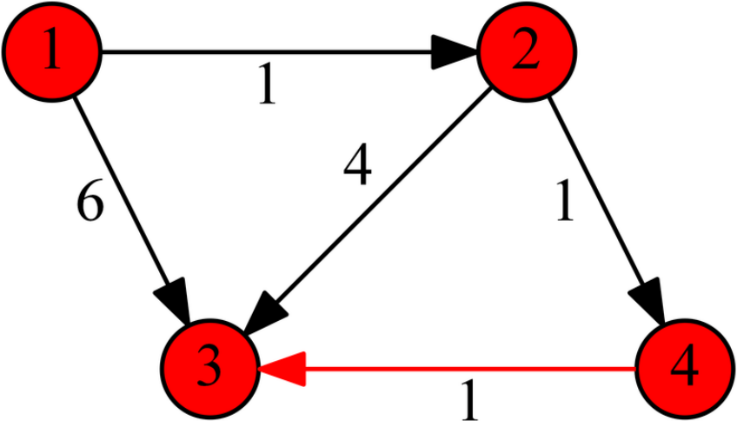
**Пример работы**

1 Дан граф

2 Выберем вершину

3 Выберем потенциально более короткий путь

4 Выберем потенциально более короткий путь с новой вершины

5 Самый короткий путь найден

# **Описание программы**

Для написания данной программы был использован язык программирование СИ++.

Проект был создан в виде консольного приложения в программе VS 2019.

Функция генерация матрицы.

int\*\* Gen\_matrix(const int SIZE) на вход получает размер матрицы

{

int\*\* a = new int\* [SIZE]; создание двумерного динамического массива

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

{

a[i] = new int[SIZE];

}

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

{

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

{

if (!(i == j))

{

a[i][j] = rand() % 100; заполнение матрицы случайными числами

}

else {

a[i][j] = 0;

}

}

}

return a; функция возвращает указатель на матрицу

}

Функция чтения матрицы из файла.

int\*\* Get\_matrix(string path,int &SIZE) получает имя файла и размер массива

{

int counter = 0;

ifstream fin;

fin.open(path);

if (!fin.is\_open())

{

cout << "\nОшибка открытия файла" << endl;

}

else

{

cout << "\nФайл открыт" << endl;

int temp;

while (fin >> temp)

{

counter++; вычисление размера матрицы

}

}

fin.close();

SIZE = sqrt(counter);

int\*\* A;

A = new int\* [SIZE];

for (size\_t i = 0; i < SIZE; i++)

{

A[i] = new int[SIZE];

}

fin.open(path);

if (!fin.is\_open())

{

cout << "\nОшибка отрытия файла" << endl;

}

else

{

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

{

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

{

fin >> A[i][j]; запись из файла в матрицу

}

}

}

return A; функция возвращает указатель на матрицу

}

Функция ввода матрицы с клавиатуры.

int\*\* Write\_matrix(const int SIZE) получает размер матрицы

{

int\*\* a = new int\* [SIZE]; создание двумерного массива

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

{

a[i] = new int[SIZE];

}

for (int i = 0; i < SIZE; i++) ввод матрицы с клавиатуры

{

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

{

if (!(i == j)) {

cout << "Введите строку " << i + 1 << " столбец " << j + 1 << ": "; cin >> a[i][j];

}

else {

a[i][j] = 0;

}

}

}

return a;

}

Функция, реализующая алгоритм Флойда.

int\*\* Floyd(const int SIZE, int\*\* a)-SIZE размер матрицы int\*\* a указатель на матрицу

{

int s, t,fiter=0,counter=0;

int\* find;

int\*\* base = new int\* [SIZE];

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

{

base[i] = new int[SIZE];

}

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

{

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

{

base[i][j] = a[i][j];

}

}

int\*\* next= new int\* [SIZE]; заполнение массива для восстановления путей

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

{

next[i] = new int[SIZE];

}

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

{

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

{

if (a[i][j] != INF)

{

next[i][j] = j;

}

else

{

next[i][j] = -1;

}

}

}

//================================================ алгоритм Флойда

for (int k = 0; k < SIZE; k++)

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

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

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

{

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

next[i][j] = next[i][k]; запись следующей вершины в новом пути

}

//=================================================

cout << "Матрица кратчайших путей в графе" << endl;

cout << "\t";

cout << endl;

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

{

cout << i + 1 << "|\t";

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

{

if (a[i][j] != INF)

{

cout << a[i][j] << " ";

}

else

{

cout << "- ";

}

}

cout << endl;

}

//===================================== подсчет количества новых более коротких путей

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

{

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

{

if (a[i][j] != base[i][j])

{

counter++;

}

}

}

find = new int[counter \* 2];

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

{

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

{

if (a[i][j] != base[i][j])

{

find[fiter] = i;

find[fiter + 1] = j;

fiter += 2;

}

}

}

cout << "\n\n";

cout << "\nНайденны более короткие пути\n"; - восстановление коротких путей

for (int i = 0; i < counter \* 2; i += 2)

{

s = find[i];

t = find[i + 1];

cout << "\nПуть " << s+1 << " --> " << t+1 << ": ";

while (s != t)

{

cout << s << " ";

s = next[s][t];

}

cout << t << endl;

}

return a;

}

Функция вывода матрицы в консоль.

void Output(const int SIZE, int\*\* a) получает размер матрицы и указатель на массив

{

//вывод матрицы в консоль

cout << endl;

for (size\_t i = 0; i < SIZE; i++)

{

cout << "R" << i + 1 << "| ";

for (size\_t j = 0; j < SIZE; j++)

{

cout << a[i][j] << " ";

}

cout << endl;

}

cout << "=========================================================" << "\n\n";

}

Функции вывода результата в файл.

void File\_output(string path, const int SIZE, int\*\* a) получает имя файла, размер матрицы

{ и указатель на матрицу

ofstream fout;

fout.open(path);

if (!fout.is\_open())

{

cout << "Ошибка отрытия файла" << endl;

}

else

{

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

{

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

{

fout << a[i][j] << " ";

}

fout << "\n";

}

}

fout.close();

}

Работа программы начинается с запроса генерации матрицы. Если пользователь выберет пункт под номером 1, то ему предложат ввести путь к файлу из которого будет записываться матрица. Если выбрать пункт 2, то пользователь введёт размер матрицы и запишет матрицу с клавиатуры. Если выбрав пункт 3, то ему предложат ввести размер матрицы, затем сгенерируют матрицу со случайными значениями заданного размера. Пункт 0 завершает работу программы.

Функция main

int main()

{

setlocale(LC\_ALL, "rus");

int SIZE;

int b;

int\*\* A;

string path;

cout << "--->1. Загрузить из файла" << endl;

cout << "--->2. Записать через консоль" << endl;

cout << "--->3. Сгенерировать случайно" << endl;

cout << "--->0. Выход" << endl;

b = \_getch();

if (b =='0' )

{

return 0;

}

if (b == '1')

{

cout << "--->Введите путь к файлу: "; cin >> path; cout << endl;

A=Get\_matrix(path, SIZE);

Output(SIZE,A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

}

}

if (b == '2')

{

cout << "\n--->Введите кол-во вершин: "; cin >> SIZE; cout << endl;

A = Write\_matrix(SIZE);

Output(SIZE, A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

}

}

if (b == '3')

{

cout << "\n--->Введите кол-во вершин: "; cin >> SIZE; cout << endl;

A = Gen\_matrix(SIZE);

Output(SIZE, A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

}

}

return 0;

}

# **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.

Таблица 1

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Вывод меню в консоль | Верно |
| Генерация матрицы случайным образом и ручным ввод размера | Вывод сообщения о вводе размера матрицы  И выводе результатов программы в консоль | Верно |
| Генерация матрицы из файла | Вывод сообщения о выборе генерации и выводе результатов программы | Верно |
| Сохранение в файл | Создание файла, содержащего результаты программы | Верно |
| Выход из программы | Завершение работы программы | Верно |

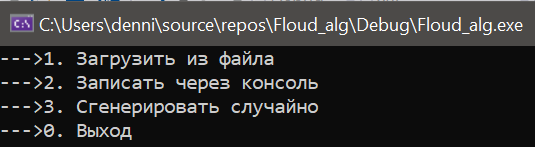


Рисунок Вывод меню в консоль

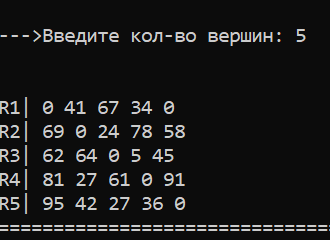


Рисунок Случайная генерация матрицы

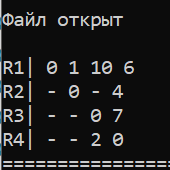


Рисунок Генерация матрицы из файла

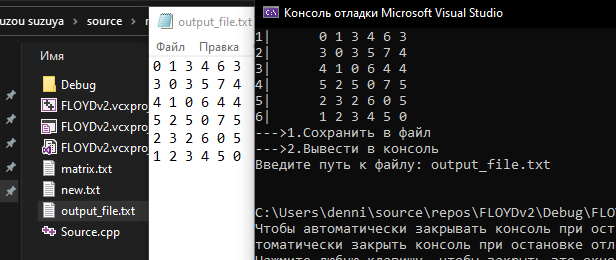


Рисунок Сохранение результата в файл

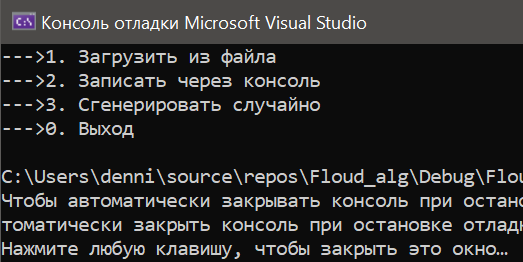


Рисунок Выход из программы

# **Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм Флойда для поиска кратчайшего пути в графе в среде MicrosoftVisualStudio 2019.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц смежностей. Приобретены навыки по осуществлению алгоритма Флойда. Углублены знания языка программирования C++.

# **Список литературы**

1. Керниган Б. Ритчи Д. Язык программирования С. 1985 г.
2. К. Джамса. Учимся программировать на языке C++. 1997 г
3. В. Г. Давыдов. Программирование и основы алгоритмизации. 2003
4. В.В. Подбельский, С.С.Фомин. Программирование на языке Си. М .: ФиС, 1999, 600 с.

# **Приложение А**

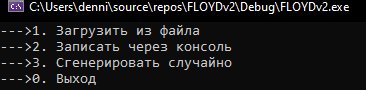


Рисунок Меню

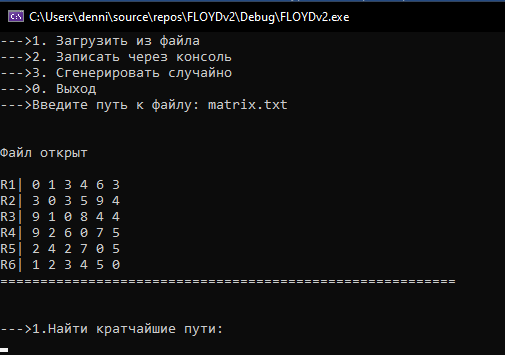


Рисунок Загрузка матрицы из файла

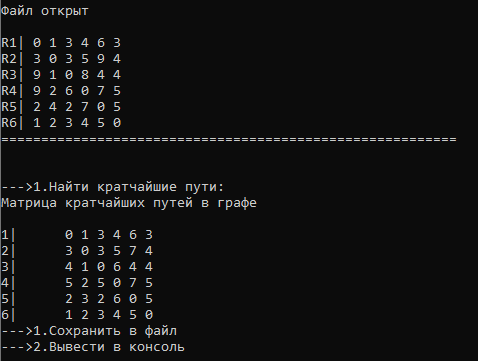


Рисунок Результат работы алгоритма

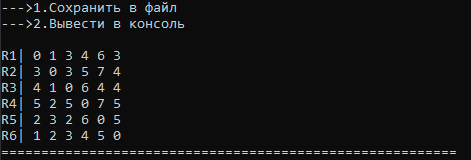


Рисунок Вывод результата работы в консоль

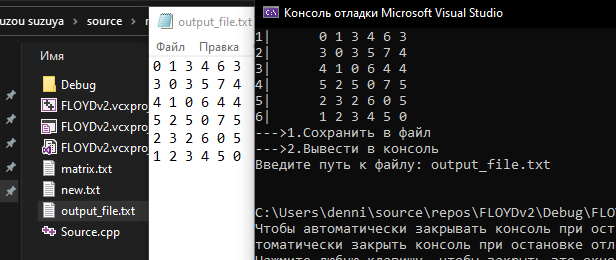


Рисунок Вывод результата в файл

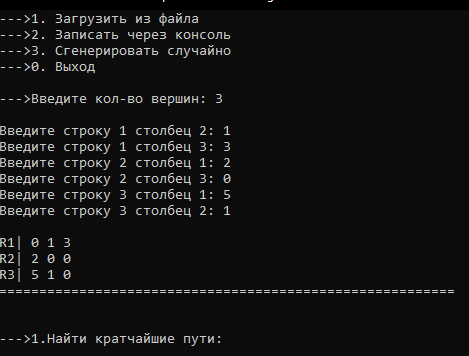


Рисунок Запись матрицы через консоль

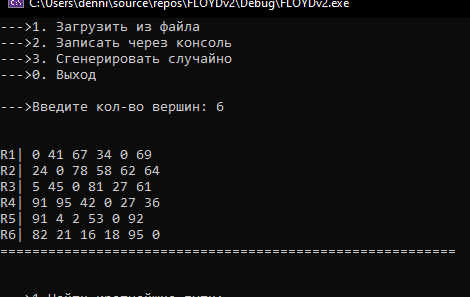


Рисунок Случайная генерация матрицы

# **Приложение В**

Листинг программы

#include <iostream>

#include <locale>

#include <time.h>

#include <fstream>

#include <conio.h>

using namespace std;

int\*\* Floyd(const int SIZE, int\*\* a)

{

//================================================

for (int k = 0; k < SIZE; k++)

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

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

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

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

//=================================================

cout << "Матрица кратчайших путей в графе" << endl;

cout << "\t";

cout << endl;

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

{

cout << i + 1 << "|\t";

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

cout << a[i][j] << " ";

}

cout << endl;

}

return a;

}

int\*\* Get\_matrix(string path,int &SIZE)

{

int counter = 0;

ifstream fin;

fin.open(path);

if (!fin.is\_open())

{

cout << "\nОшибка открытия файла" << endl;

}

else

{

cout << "\nФайл открыт" << endl;

int temp;

while (fin >> temp)

{

counter++;

}

}

fin.close();

SIZE = sqrt(counter);

int\*\* A;

A = new int\* [SIZE];

for (size\_t i = 0; i < SIZE; i++)

{

A[i] = new int[SIZE];

}

fin.open(path);

if (!fin.is\_open())

{

cout << "\nОшибка отрытия файла" << endl;

}

else

{

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

{

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

{

fin >> A[i][j];

}

}

}

return A;

}

int\*\* Gen\_matrix(const int SIZE)

{

int\*\* a = new int\* [SIZE];

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

{

a[i] = new int[SIZE];

}

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

{

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

{

if (!(i == j))

{

a[i][j] = rand() % 100;

}

else {

a[i][j] = 0;

}

}

}

return a;

}

int\*\* Write\_matrix(const int SIZE)

{

int\*\* a = new int\* [SIZE];

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

{

a[i] = new int[SIZE];

}

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

{

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

{

if (!(i == j)) {

cout << "Введите строку " << i + 1 << " столбец " << j + 1 << ": "; cin >> a[i][j];

}

else {

a[i][j] = 0;

}

}

}

return a;

}

void Output(const int SIZE, int\*\* a)

{

//вывод матрицы в консоль

cout << endl;

for (size\_t i = 0; i < SIZE; i++)

{

cout << "R" << i + 1 << "| ";

for (size\_t j = 0; j < SIZE; j++)

{

cout << a[i][j] << " ";

}

cout << endl;

}

cout << "=========================================================" << "\n\n";

}

void File\_output(string path, const int SIZE, int\*\* a)

{

ofstream fout;

fout.open(path);

if (!fout.is\_open())

{

cout << "Ошибка отрытия файла" << endl;

}

else

{

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

{

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

{

fout << a[i][j] << " ";

}

fout << "\n";

}

}

fout.close();

}

int main()

{

setlocale(LC\_ALL, "rus");

int SIZE;

int b;

int\*\* A;

string path;

cout << "--->1. Загрузить из файла" << endl;

cout << "--->2. Записать через консоль" << endl;

cout << "--->3. Сгенерировать случайно" << endl;

cout << "--->0. Выход" << endl;

b = \_getch();

if (b =='0' )

{

return 0;

}

if (b == '1')

{

cout << "--->Введите путь к файлу: "; cin >> path; cout << endl;

A=Get\_matrix(path, SIZE);

Output(SIZE,A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

return 0;

}

}

if (b == '2')

{

cout << "\n--->Введите кол-во вершин: "; cin >> SIZE; cout << endl;

A = Write\_matrix(SIZE);

Output(SIZE, A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

}

return 0;

}

if (b == '3')

{

cout << "\n--->Введите кол-во вершин: "; cin >> SIZE; cout << endl;

A = Gen\_matrix(SIZE);

Output(SIZE, A);

cout << "\n--->1.Найти кратчайшие пути: " << endl;

b = \_getch();

if (b == '1')

{

Floyd(SIZE, A);

cout << "--->1.Сохранить в файл" << endl;

cout << "--->2.Вывести в консоль" << endl;

b = \_getch();

if (b == '1')

{

cout << "Введите путь к файлу: "; cin >> path; cout << endl;

File\_output(path, SIZE, A);

}

if (b == '2')

{

Output(SIZE, A);

}

}

return 0;

}

return 0;

}