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-Уоршелла поиска

кратчайших путей в графе”

Выполнил:

Студент группы 20ВВ3

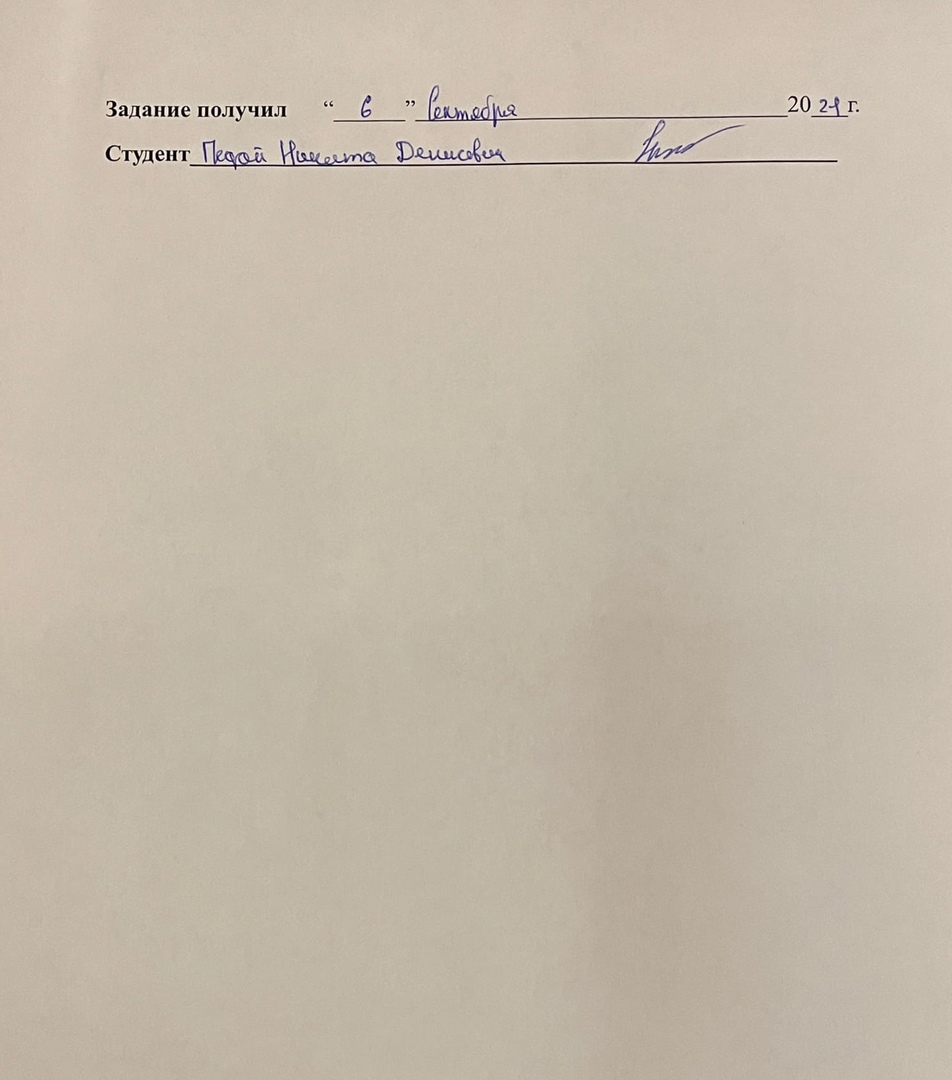
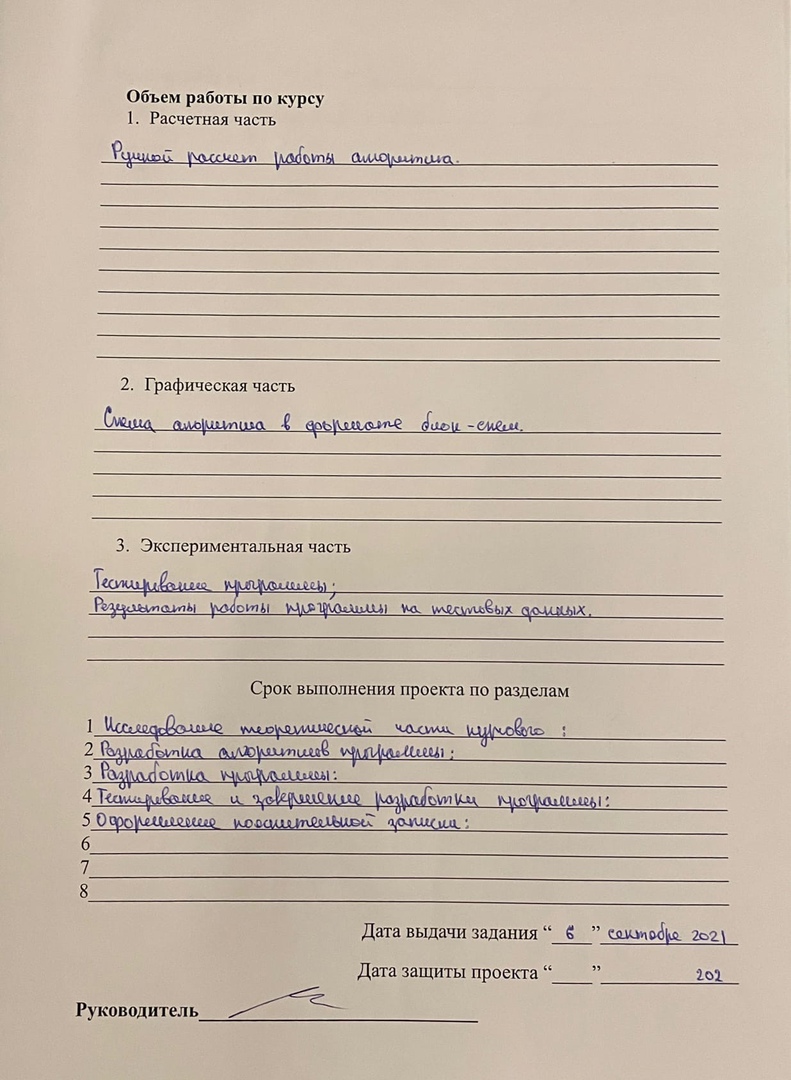
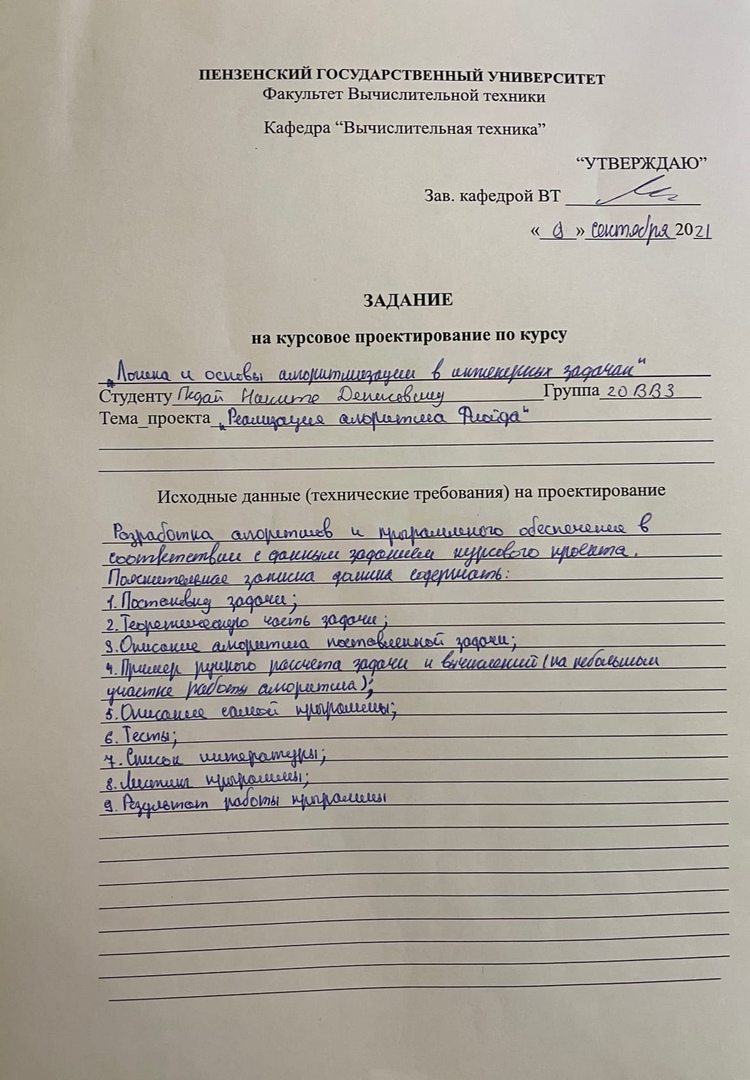
Педай Н.Д.

Приняли:

Митрохин М.А.

Юрова О.В.

Пенза 2021



# **Содержание**

[**Реферат** 6](#_Toc90282997)

[**Введение** 7](#_Toc90282998)

[**1.Постановка задачи** 8](#_Toc90282999)

[**2.Теоретическая часть** 9](#_Toc90283000)

[**3.Описание алгоритма программы** 11](#_Toc90283001)

[**4.Описание программы** 12](#_Toc90283002)

[**5.Тестирование программы** 19](#_Toc90283003)

[**6.Ручной расчет задачи** 22](#_Toc90283004)

[**Заключение** 24](#_Toc90283005)

[**Список литературы** 25](#_Toc90283006)

[**Приложение А. Листинг** 26](#_Toc90283007)

# **Реферат**

Отчет 31 страница ,22 рисунка

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 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 графе с помощью алгоритма Флойда-Уоршелла.

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**

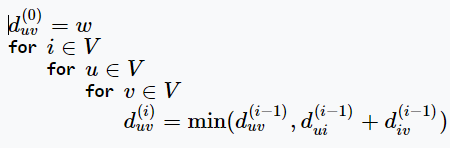
Обозначим длину кратчайшего пути между вершинами *u* и *v*, содержащего, помимо *u* и *v*, только вершины из множества:



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

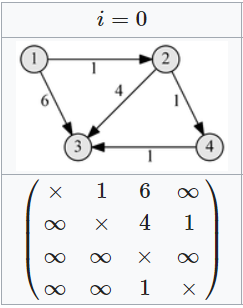
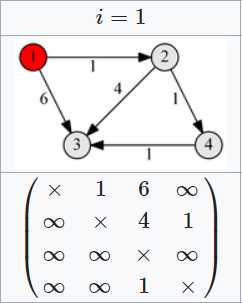


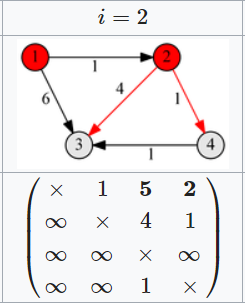
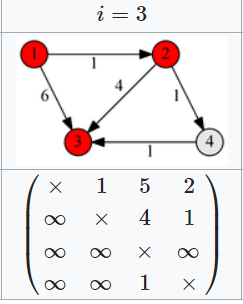
То есть, если кратчайший путь из *u* в *v*, содержащий только вершины из множества {1..i} , проходит через вершину *i*, то кратчайшим путем из *u* в *v* является кратчайший путь из u в *i*, объединенный с кратчайшим путем из *i* в *v*. В противном случае, когда этот путь не содержит вершины *i*, кратчайший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

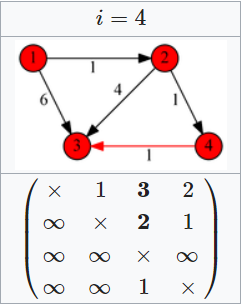


В итоге получаем, что матрица   является искомой матрицей кратчайших путей, поскольку содержит в себе длины кратчайших путей между всеми парами вершин, имеющих в качестве промежуточных вершин вершины из множества {1..n}, что есть попросту все вершины графа. Такая реализация работает за Θ(n3) времени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

Пример работы алгоритма:



# **3.Описание алгоритма программы**

При заполнении матрицы (ручном или автоматическом) используется динамический двумерный массив. Если между вершинами графа отсутствует прямой путь, то в массив записывается значение =10000.

После того как матрица выводится на экран начинает работать алгоритм Флойда-Уоршелла.

С помощью вложенных циклов производится поиск кратчайших путей. Если соблюдается условие mas[i][j] >mas[i][k] + mas[k][j] ,то в mas[i][j] записывается сумма mas[i][k] + mas[k][j].

После вышеперечисленных манипуляций измененная матрица выводится на экран и пользователю предлагается сохранить результат в файл формата .txt.

Ниже представлен псевдокод для данного алгоритма:

для k=0 пока k<N k=k+1

для i=0 пока i<N i=i+1

для j=0 пока j<N j=j+1

если (mas[i][j] больше mas[i][k] + mas[k][j]

то mas [i][j] = mas[i][k]+mas[k][j]

конец условия

# **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++).

При запуске программы пользователю представляется возможность выбрать один из двух видов заполнения матрицы: ручного или автоматического.

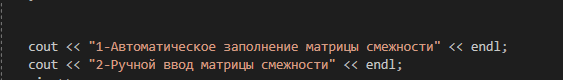


Рисунок 1-меню программы.

Далее, при выборе варианта “Автоматическое заполнение матрицы смежности” пользователь должен ввести размер заполняемой матрицы (NxN).

Далее происходит инициализация динамического массива и его заполнение псевдослучайными числами. (Если связи между вершинами графа нет, то пишется число 10000)

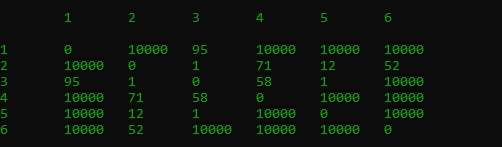


Рисунок 2-заполнение массива псевдослучайными числами

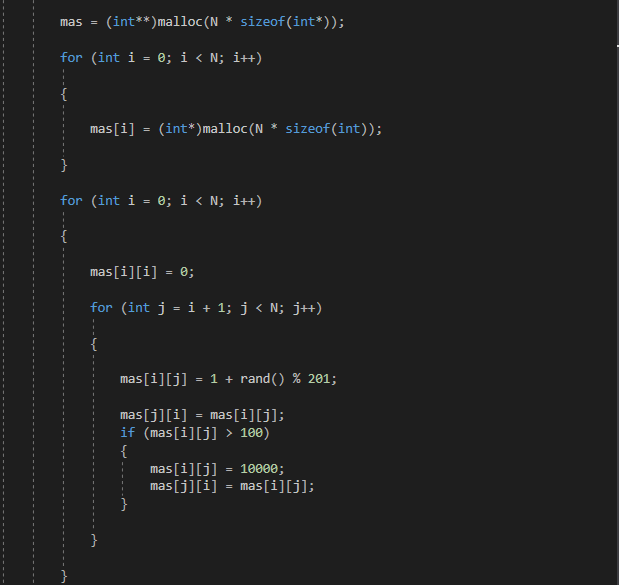


Рисунок 3-код автоматического заполнения массива

После заполнения производится вывод матрицы в консоль.

Затем происходит непосредственная реализация алгоритма Флойда-Уоршелла для поиска кратчайших путей между вершинами взвешенного графа.

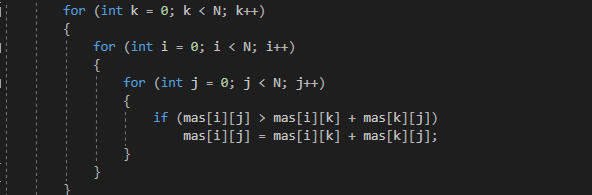


Рисунок 4-код алгоритма Флойда-Уоршелла

По завершении работы данного алгоритма результат выводится на экран и пользователю предлагается сохранить полученный результат в файл формата txt.

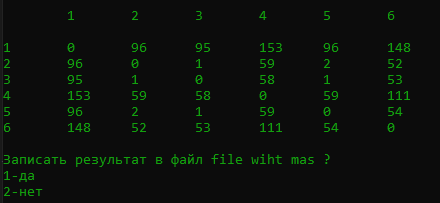


Рисунок 5-результат работы алгоритма

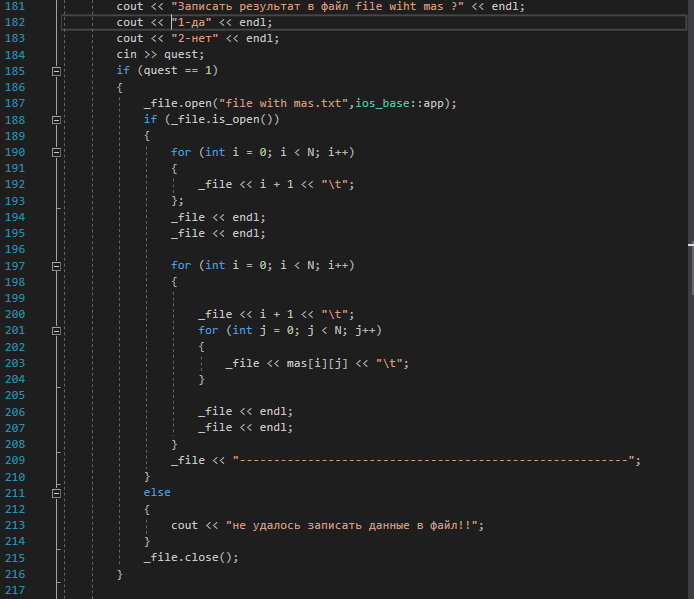


Рисунок 6-код для записи результата в файл

Для записи результата работы программы в файл следует ввести ‘1’.

Для просмотра содержимого файла нужно зайти в папку с проектом

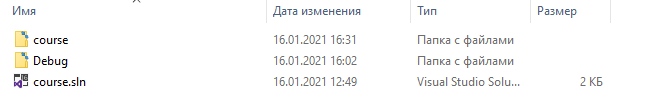


Рисунок 7 -путь в папку с сохраненным файлом

Далее в папку “course”

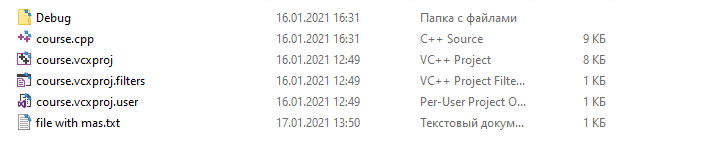


Рисунок 8- путь в папку с сохраненным файлом

Открыть файл “file with mas.txt”

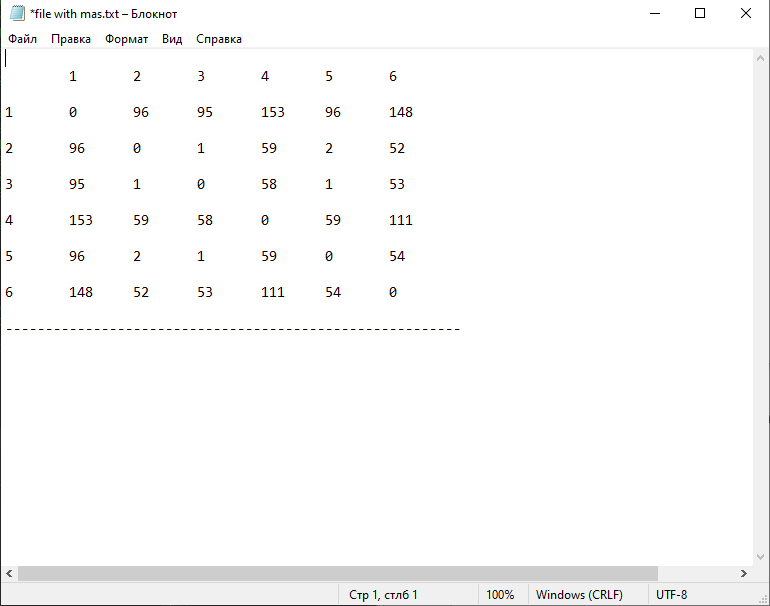


Рисунок 9-файл с результатом работы программы

При выборе в меню цифры “2-Ручной ввод матрицы смежности”



Рисунок 10-меню программы

Пользователь также должен ввести данные о размере массива(NxN) как и в случае автоматического заполнения:



Рисунок 11-установка размера матрицы при ручном вводе

Для наглядности составим небольшой граф вручную в тетради и перенесем данные в программу

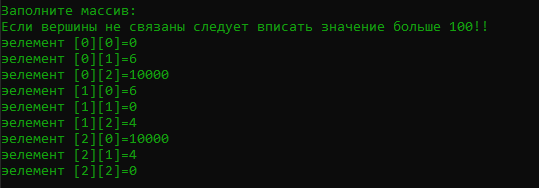


Рисунок 13-ручное заполнение графа в программе

Программа вывела сначала необработанный массив, затем измененный массив

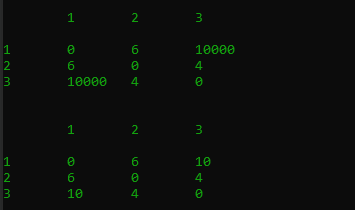


Рисунок 14-вывод матриц на экран

Пользователь, при ручном заполнении, также может сохранить в файл полученную в результате работы программы матрицу



Рисунок 15-меню сохранения в файл

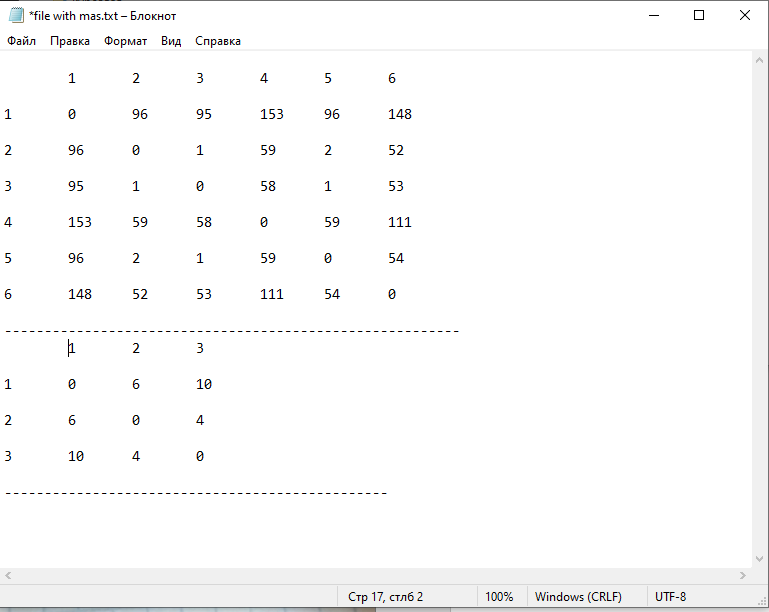


Рисунок 16-файл с сохраненным результатом

Запись производится в конец файла.

# **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 продемонстрирован результат тестирования программы:

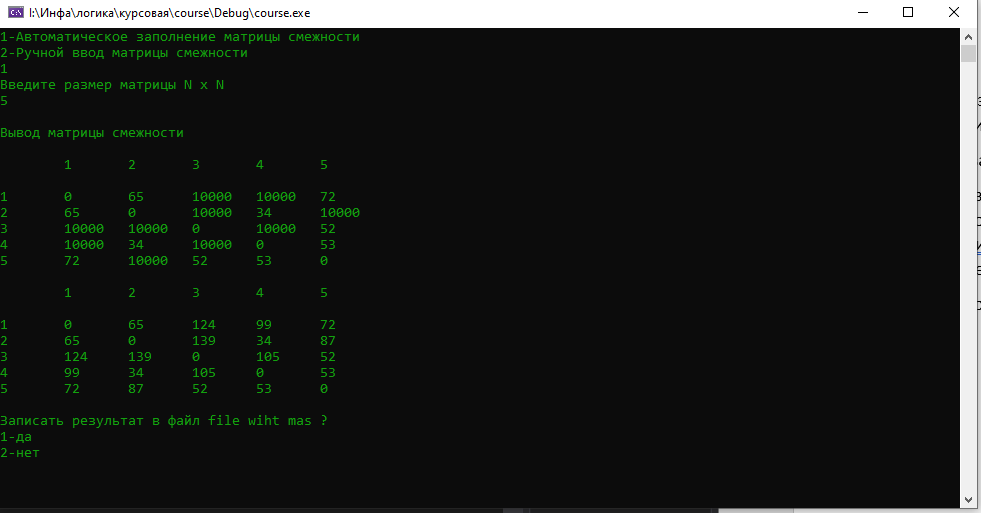


Рисунок 17-тестирование программы

Тестирование автоматического заполнения при N=5

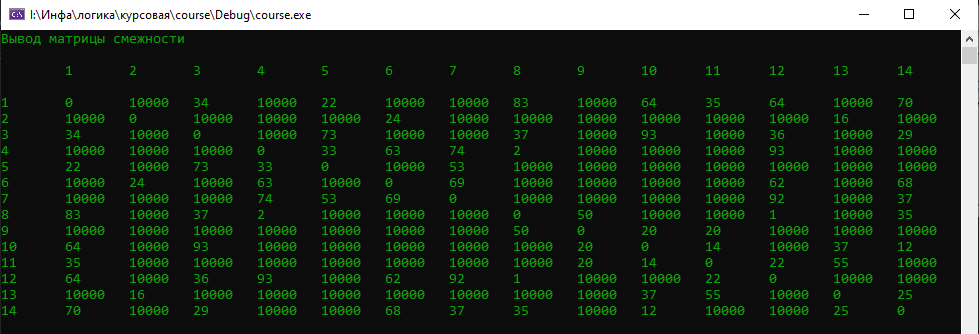


Рисунок 18-тестирование программы

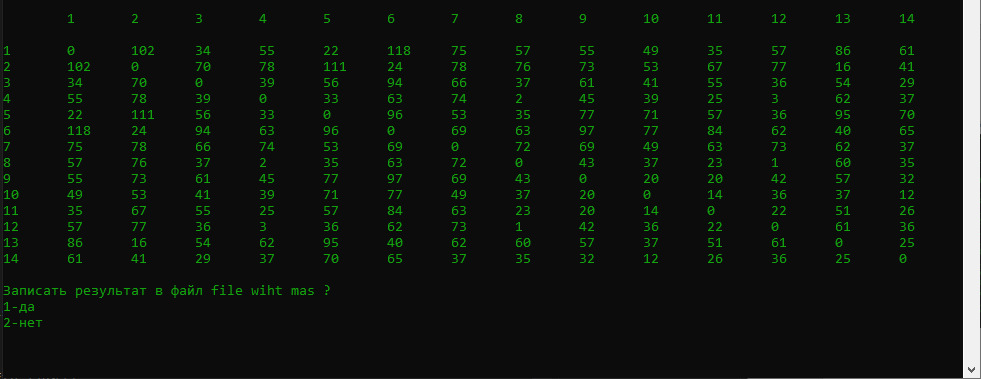


Рисунок 19 -тестирование программы

Тестирование программы при N=14

Произвели тестирование ручного заполнения программы

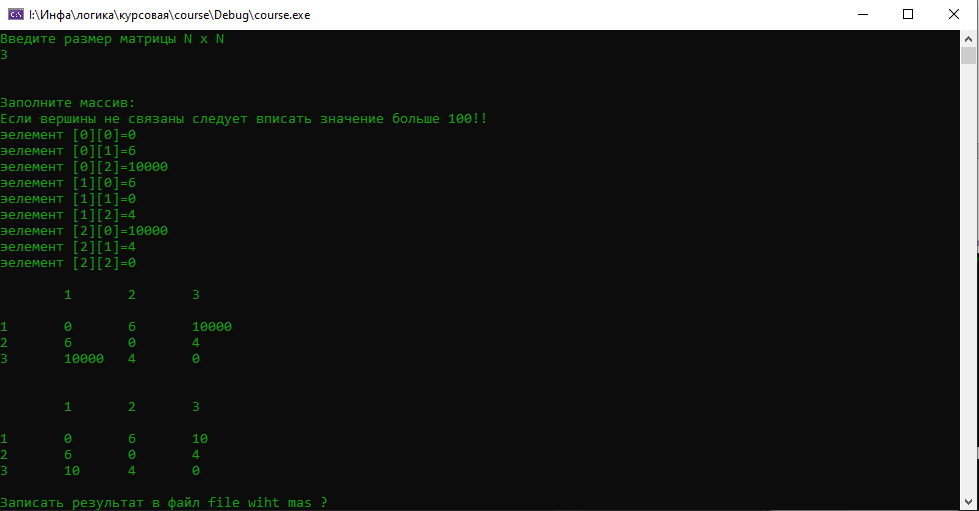


Рисунок 20-тестирование программы

Для удобства заполнения N=3. При любом способе ввода (ручном или автоматическом) программа работает корректно.

# **6.Ручной расчет задачи**

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

С помощью программы заполним матрицу случайными числами

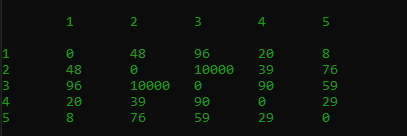


Рисунок 21-заполнение матрицы

Для наглядности нарисовали граф в тетради. для несоединенных вершин написали значение inf (infinite)

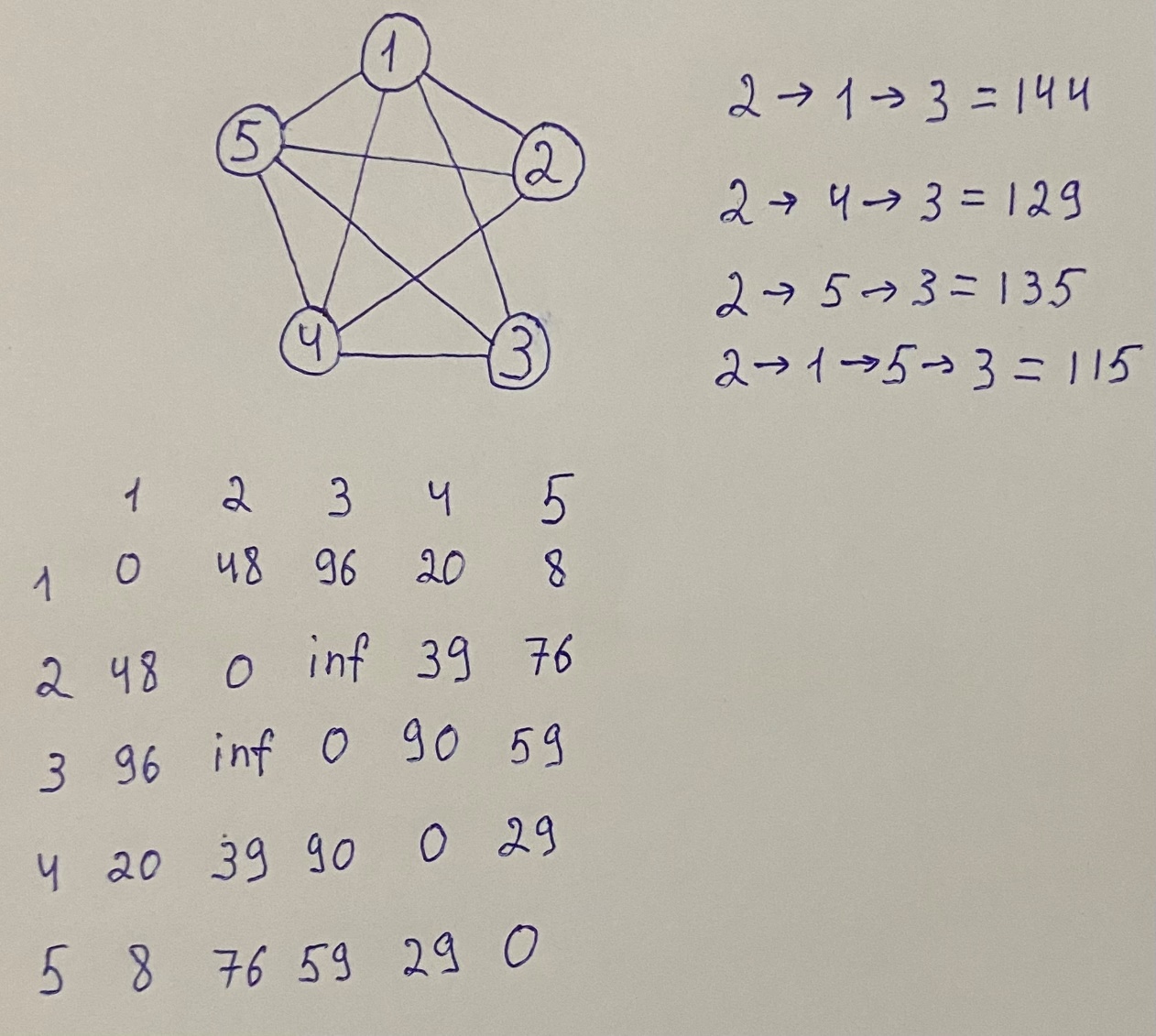


Рисунок 22-представление графа на бумаге

Отсюда видно, что вершины 2 и 3 не соединены между собой напрямую. Путем перебора различных вариантов самым коротким оказался следующий путь:

-Из вершины 2 в вершину 1

-Из вершины 1 в вершину 5

-Из вершины 5 в вершину 3

Полученный путь равен =115

# **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**

* Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд — М.: Издательский дом «Вильямс», 2009. — ISBN 978-5-8459-0857-5.
* Романовский И. В. Дискретный анализ: Учебное пособие для студентов, специализирующихся по прикладной математике и информатике. Изд. 3-е. — СПб.: Невский диалект, 2003. — 320 с. — ISBN 5-7940-0114-3.

# **Приложение А. Листинг**

#include <iostream>

#include <conio.h>

#include <time.h>

#include <fstream>

//Максимальное значение веса = 100

#define INF 101

using namespace std;

typedef struct Node {

int vertex;

struct Node\* next;

} Node;

int main()

{

system("color 2");

srand(time(NULL));

fstream \_file;

int\*\* mas, N;// N-количество вершин

int menu,quest;

\_cputws(L"1-Автоматическое заполнение матрицы смежности\n");

\_cputws(L"2-Ручной ввод матрицы смежности\n");

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

if (menu == 1) //Автоматическое заполнение

{

\_cputws(L"Введите размер матрицы N x N\n");

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

Node\*\* head = (Node \* \*)malloc(N \* sizeof(Node\*));

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

head[i] = NULL;

bool\* visited = (bool\*)malloc(N \* sizeof(bool));

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

visited[i] = false;

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

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

mas[i] = (int\*)malloc(N \* sizeof(int));

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

{

mas[i][i] = 0;

for (int j = i + 1; j < N; j++)

{

mas[i][j] = 1 + rand() % 201;

mas[j][i] = mas[i][j];

if (mas[i][j] > 100)

{

mas[i][j] = 10000;

mas[j][i] = mas[i][j];

}

}

}

\_cputws(L"\nВывод матрицы смежности\n\n\t ");

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

printf("%d\t", i + 1);

\_cputs("\n");

for (int i = 0; i < N; i++) // вывод матриц смежности

{

printf("\n%d\t", i + 1);

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

printf("%d\t", mas[i][j]);

}

}

\_cputs("\n\n");

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

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

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

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

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

\_cputs("\t");

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

printf("%d\t", i + 1);

\_cputs("\n\n");

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

{

printf("%d\t", i + 1);

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

printf("%d\t", mas[i][j]);

\_cputs("\n");

}

\_cputs("\n");

\_cputws(L"Записать результат в файл file wiht mas ?\n");

\_cputws(L"1-да\n");

\_cputws(L"2-нет\n");

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

if (quest == 1)

{

\_file.open("file with mas.txt",ios\_base::app);

if (\_file.is\_open())

{

\_file << "\n";

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

\_file << "\t" << i + 1;

\_file << "\n\n";

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

{

\_file << i + 1 << "\t";

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

{

\_file << mas[i][j] << "\t";

}

\_file << "\n\n";

}

\_file << "---------------------------------------------------------";

}

else

\_cputws(L"Не удалось записать данные в файл!!\n");

\_file.close();

}

}

if (menu == 2) // Ручное заполнение массива

{

\_cputws(L"Введите размер матрицы N x N\n");

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

\_cputs("\n\n");

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

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

mas[i] = (int\*)malloc(N \* sizeof(int));

\_cputws(L"Заполните массив:\n");

\_cputws(L"Если вершины не связаны следует вписать значение больше 100!!\n\n");

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

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

{

\_cputws(L"Элемент ");

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

cin >> mas[i][j];

}

\_cputs("\n\t");

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

printf("%d\t", i + 1);

\_cputs("\n\n");

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

{

printf("%d\t", i + 1);

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

printf("%d\t", mas[i][j]);

\_cputs("\n");

}

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

{

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

{

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

{

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

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

}

}

}

\_cputs("\n\n\t");

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

printf("%d\t", i + 1);

\_cputs("\n\n");

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

{

printf("%d\t", i + 1);

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

printf("%d\t", mas[i][j]);

\_cputs("\n");

}

\_cputs("\n");

\_cputws(L"Записать результат в файл file wiht mas ?\n");

\_cputws(L"1-да\n");

\_cputws(L"2-нет\n");

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

if (quest == 1)

{

\_file.open("file with mas.txt", ios\_base::app);

if (\_file.is\_open())

{

\_file << "\n";

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

\_file << "\t" << i + 1;

\_file << "\n\n";

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

{

\_file << i + 1 << "\t";

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

{

\_file << mas[i][j] << "\t";

}

\_file << "\n\n";

}

\_file << "---------------------------------------------------------";

}

else

\_cputws(L"Не удалось записать данные в файл!!\n");

\_file.close();

}

}

return 0;

}