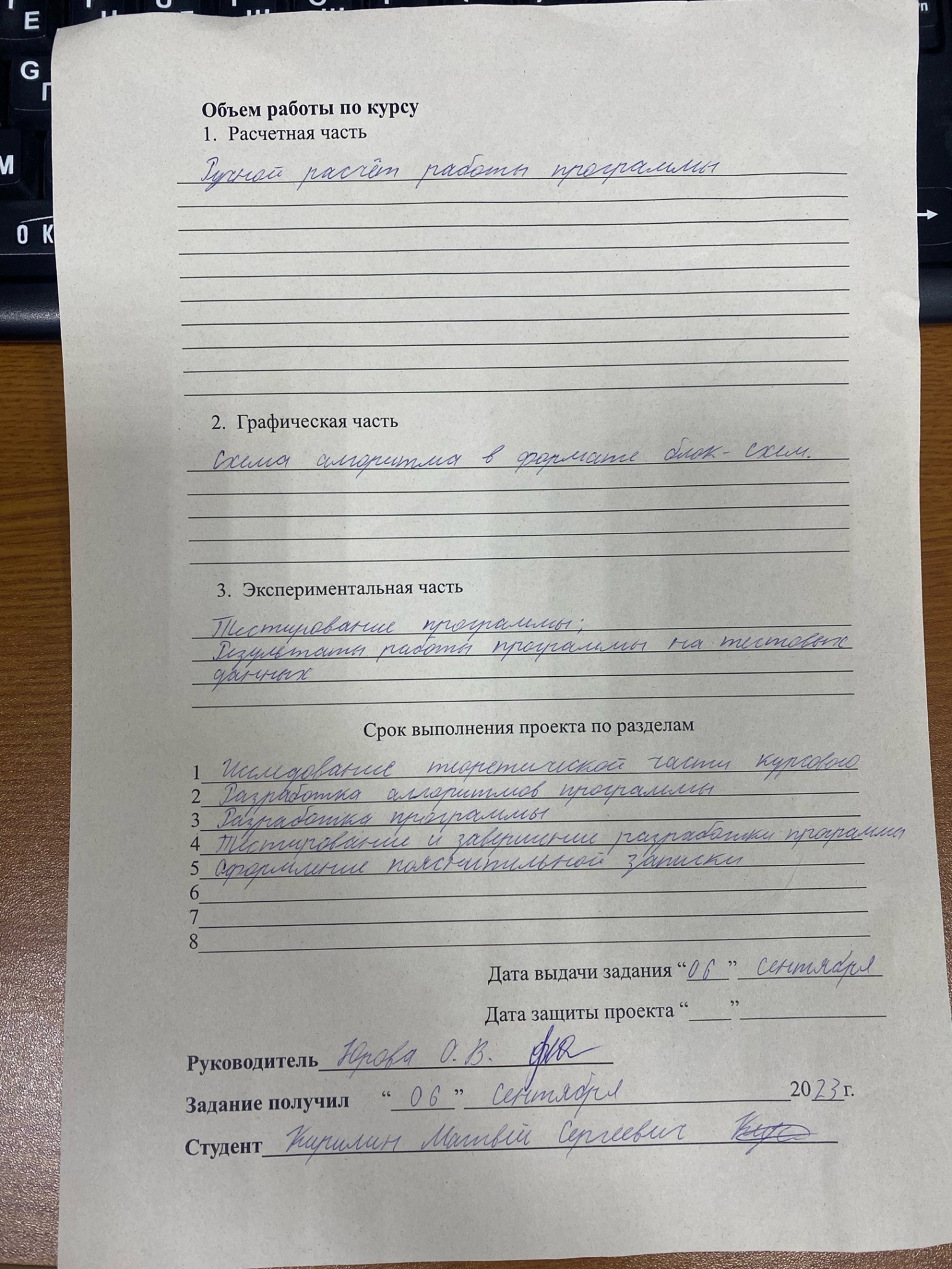
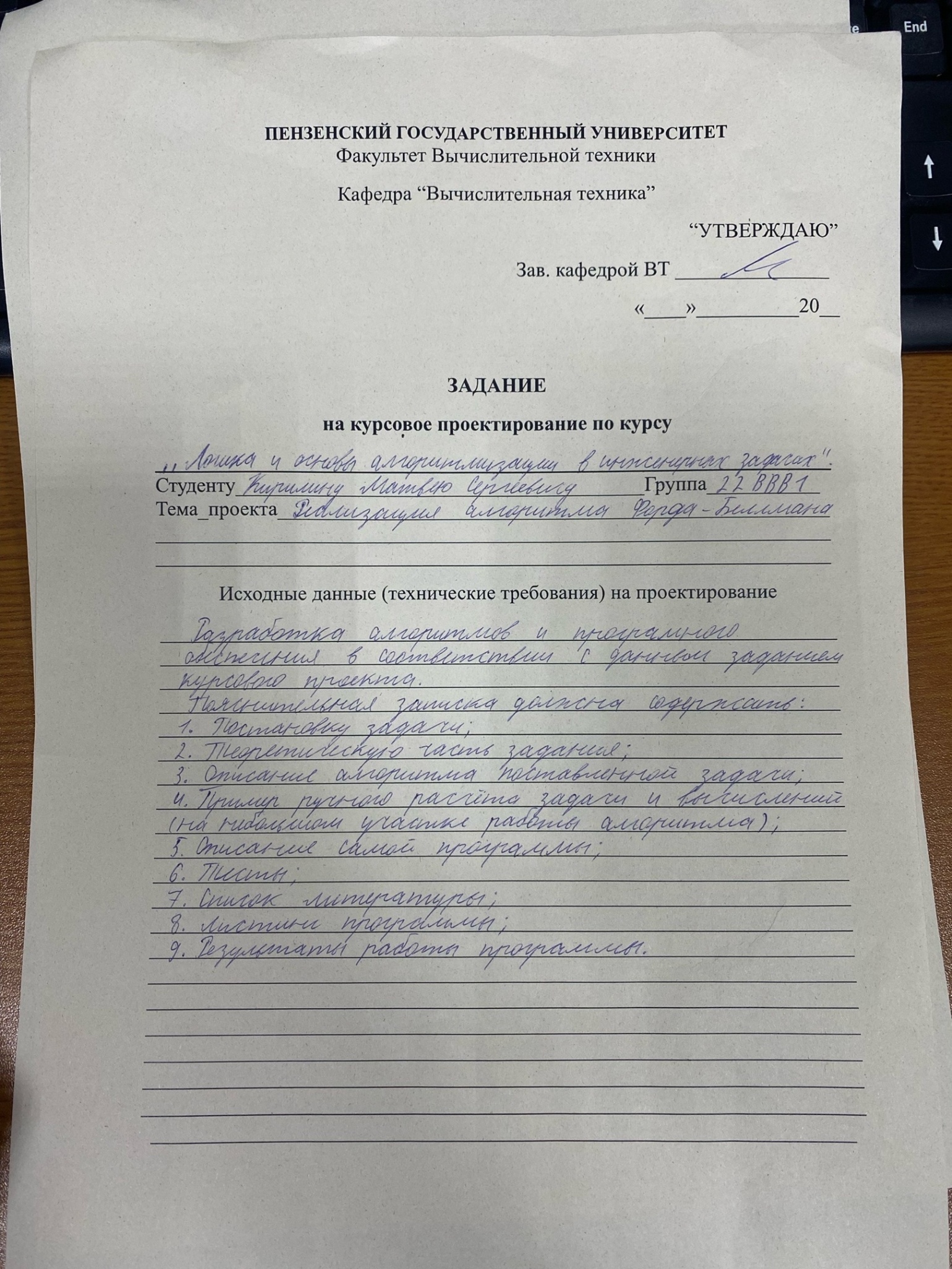
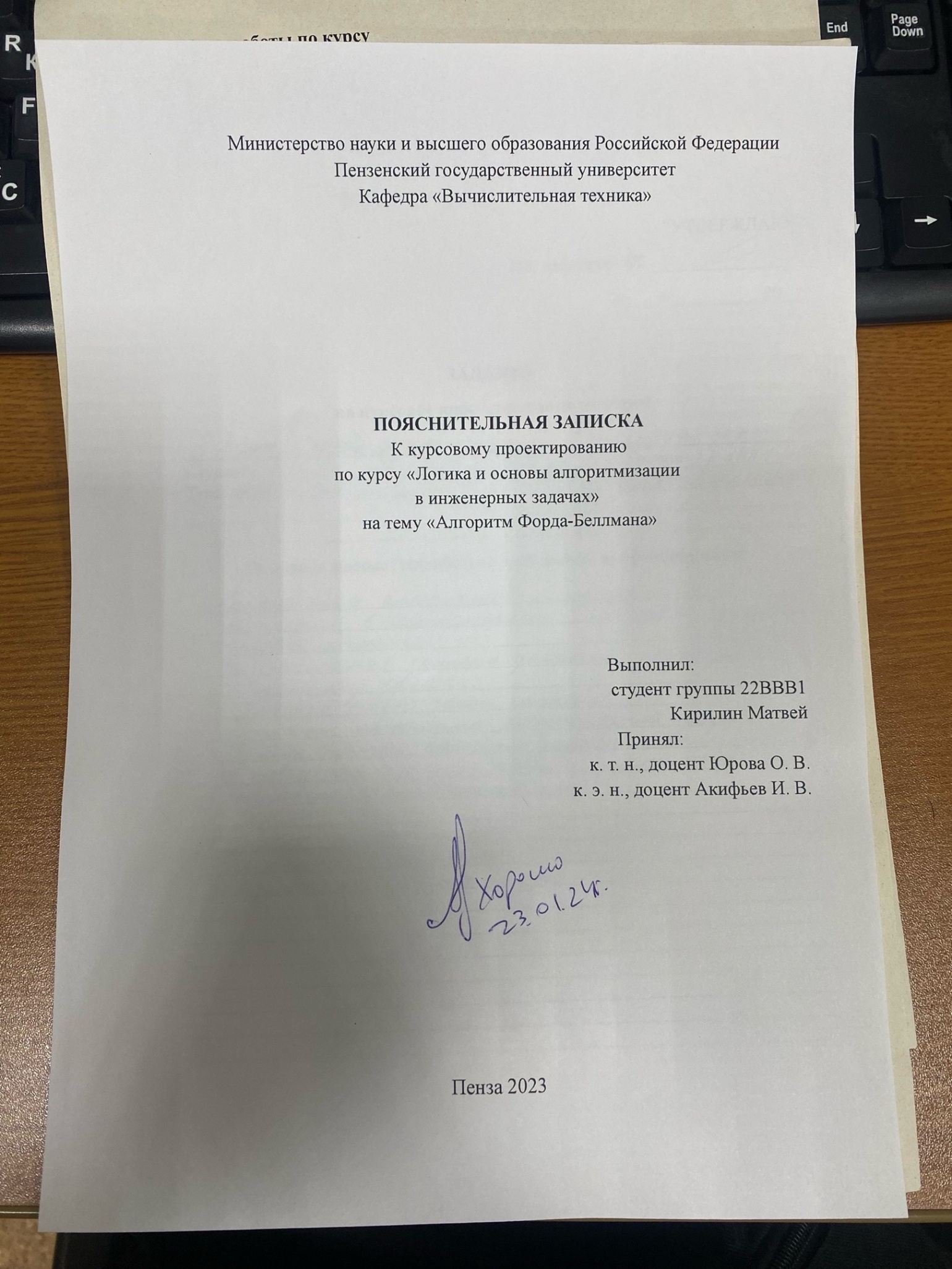
****

# 

# Содержание

[Содержание 1](#_Toc154432850)

[Реферат 2](#_Toc154432851)

[Введение 3](#_Toc154432852)

[1. Постановка задачи 4](#_Toc154432853)

[2. Теоретическая основа задания 5](#_Toc154432854)

[3. Описание программного алгоритма 6](#_Toc154432855)

[Описание программы 9](#_Toc154432856)

[4. Тестирование 11](#_Toc154432857)

[5. Ручной расчёт задачи 14](#_Toc154432858)

[Заключение 16](#_Toc154432859)

[Список литературы 17](#_Toc154432860)

[Приложение А. 18](#_Toc154432861)

# Реферат

Отчет 23 стр, 8 рисунков.

Алгоритм Форда-Беллмана

Цель исследования – разработка программы алгоритма Форда-Беллмана.

В работе рассмотрена реализация алгоритма Форда-Беллмана поиска кратчайшего пути во взвешенном графе. Установлено, что с помощью данного алгоритма можно найти кротчайший путь во взвешенном графе.

**Введение**

Алгоритм Форда-Беллмана - алгоритм поиска кратчайшего пути во взвешенном графе.

История алгоритма связана сразу с тремя независимыми математиками: Лестером Фордом, Ричардом Беллманом и Эдвардом Муром. Форд и Беллман опубликовали алгоритм в 1956 и 1958 годах соответственно, а Мур сделал это в 1957 году. И иногда его называют алгоритмом Беллмана – Форда – Мура. Метод используется в некоторых протоколах дистанционно-векторной маршрутизации, например в RIP (Routing Information Protocol – Протокол маршрутной информации).

Также как и алгоритм Дейкстры, алгоритм Беллмана — Форда вычисляет во взвешенном графе кратчайшие пути от одной вершины до всех остальных. Он подходит для работы с графами, имеющими ребра с отрицательным весом. Но спектр применимости алгоритма затрагивает не все такие графы, ввиду того, что каждый очередной проход по пути, составленному из ребер, сумма весов которых отрицательна (т. е. по отрицательному циклу), лишь улучшает требуемое значение. Бесконечное число улучшений делает невозможным определение одного конкретного значения, являющегося оптимальным.

В качестве среды разработки мною была выбрана среда MicrosoftVisualStudio 2017, язык программирования – С++.

В ходе выполнения данной курсовой работы были приобретены навыки работы с формами и их элементами в среде MicrosoftVisualStudio 2017, навыки работы с проектами и многомодульными программами.

1. **Постановка задачи**

Требуется разработать программу, которая находит кратчайший путь во взвешенном графе, используя алгоритм Форда-Беллмана.

Разработать эффективную программу, способную находить кратчайшие пути во взвешенном графе с использованием алгоритма Форда-Беллмана, обеспечивая удобный ввод, наглядный вывод результатов и обработку возможных ошибочных сценариев.

1. **Теоретическая основа задания**

Алгоритм Форда-Беллмана:

Алгоритм основан на принципе динамического программирования и используется для поиска кратчайших путей в графе от одной заданной вершины до всех остальных.

Он способен работать с графами, содержащими рёбра с отрицательными весами, но обнаруживает наличие отрицательных циклов в графе.

Структуры данных:

В рамках программы используются динамические массивы для представления матрицы смежности, массива расстояний (dist), массива предков (parent) и временной матрицы (temp).

За исключением матрицы смежности, все остальные структуры данных выделяются в динамической памяти.

Оптимизации:

При реализации алгоритма вводится возможность сохранения результатов в файл (result.txt), что позволяет пользователям сохранять результаты работы программы для последующего анализа.

Результаты:

Программа предоставляет пользователю удобный интерфейс для работы с алгоритмом Форда-Беллмана. Ввод данных позволяет пользователю создать взвешенный граф, а алгоритм находит кратчайшие пути от заданной вершины до всех остальных. Возможность сохранения результатов в файл обеспечивает удобство использования программы для различных вариантов анализа графов.

1. **Описание программного алгоритма**

Для реализации алгоритма создаются три массива dist, parent, и temp для хранения расстояний, родительских вершин и временного массива весов рёбер.(Память выделяется для каждого массива с использованием функции malloc.)

Далее temp заполняется значениями из исходного массива arr.

dist и parent инициализируются: расстояние до всех вершин устанавливается в бесконечность (INT\_MAX), а родительские вершины - в -1.Пользователь вводит вершину-источник (src), от которой начинается поиск кратчайших путей.

Внешний цикл (k) выполняется rows - 1 раз (где rows - количество вершин в графе), а внутренние циклы обходят все рёбра графа.

Для каждого ребра проверяется, можно ли улучшить текущее расстояние до вершины j через вершину i. Если это возможно, обновляются значения dist[j] и parent[j]. Следующим шагом выводятся кратчайшие расстояния и соответствующие им пути от вершины-источника (src) до остальных вершин графа.

Ниже представлен псевдокод части программы, ответственной за поиск кратчайшего пути в граф.

1.choice: целое

2.save: логическое

3. dist: массив целых размером rows

4. parent: массив целых размером rows

5. temp: матрица целых размером rows x rows

6. src: целое

7. Вывести("Хотите записать результат в файл:")

8. Вывести("1. ДА")

9. Вывести("2. НЕТ")

10. Вывести("Выберите действие: ")

11. Ввести(choice)

12. Если choice равно 1

13. save установить в Истина

14. dist выделить память размером rows \* размер целого

15. parent выделить память размером rows \* размер целого

16. temp выделить память размером rows x rows \* размер целого

17. Если temp равно NULL

18. Вывести("Не удалось выделить память!")

19. Вернуть

20. Для каждого i от 0 до rows

21. temp[i] выделить память размером rows \* размер целого

22. Для каждого i от 0 до rows

23. Для каждого j от 0 до rows

24. temp[i][j] установить равным arr[i][j] // на главной диагонали нули

25. Для каждого i от 0 до rows

26. dist[i] установить равным INT\_MAX

27. parent[i] установить равным -1

28. Вывести("Введите вершину источник:")

29. Ввести(src)

30. dist[src] установить равным 0

31. // Реализация алгоритма Форда-Беллмана

32. Для каждого k от 0 до rows - 1

33. Для каждого i от 0 до rows

34. Для каждого j от 0 до rows

35. temp[i][j] установитьравным (temp[i][j] < 0) ? -temp[i][j] : temp[i][j]

36. Если temp[i][j] неравно 0 и dist[i] неравно INT\_MAX и dist[i] + temp[i][j] меньше dist[j]

37. dist[j] установитьравным dist[i] + temp[i][j]

38. parent[j] установить равным i

39. // Вывод кратчайших путей

40. Для каждого i от 0 до rows

41. Если i не равно src

42. Вывести("Кратчайший путь от вершины", src, "до вершины", i, ":", dist[i])

43. Вывести("Путь:", i)

44. p установить равным parent[i]

45. Пока p не равно -1

46. Вывести(" <-", p)

47. p установить равным parent[p]

48. Вывести()

49. Если save равно Истина

50. Открыть файл "result.txt" для записи

51. Если файл открыт успешно

52. Записать "Матрица смежности" в файл

53. Для каждого i от 0 до rows

54. Для каждого j от 0 до rows

55. Записать arr[i][j] в файл, пробел

56. Записать новую строку в файл

57. Для каждого i от 0 до rows

58. Если i не равно src

59. Записать "Кратчайший путь от вершины", src, "до вершины", i, ":", dist[i] в файл

60. Записать "Путь:", i в файл

61. p установить равным parent[i]

62. Пока p не равно -1

63. Записать " <-", p в файл

64. p установить равным parent[p]

65. Записать новую строку в файл

66. Закрыть файл

67. Освободитьпамять: dist, parent, temp

Полный код программы представлен в Приложении А.

## **Описание программы**

Для написания данной программы использован язык программирования С++. Язык программирования С++ - 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++).

Программа имеет основные функции:

**Создание случайного графа**:

Пользователь вводит количество вершин графа.

Матрица смежности заполняется случайными весами для рёбер, и она выводится на экран.

1. Алгоритм Форда-Беллмана:
   * Программа запускает алгоритм Форда-Беллмана для нахождения кратчайших путей от заданной вершины до всех остальных.
   * Результаты алгоритма, включая кратчайшие пути и расстояния, выводятся на экран.
   * Возможность сохранения результатов в файл result.txt.
2. Просмотр матрицы смежности:
   * Пользователь может просмотреть матрицу смежности графа.
3. Обработка пользовательского ввода:
   * Программа предоставляет меню с опциями для выбора действий.
   * Пользователь может выбрать создание графа, запуск алгоритма, просмотр матрицы и выход из программы.

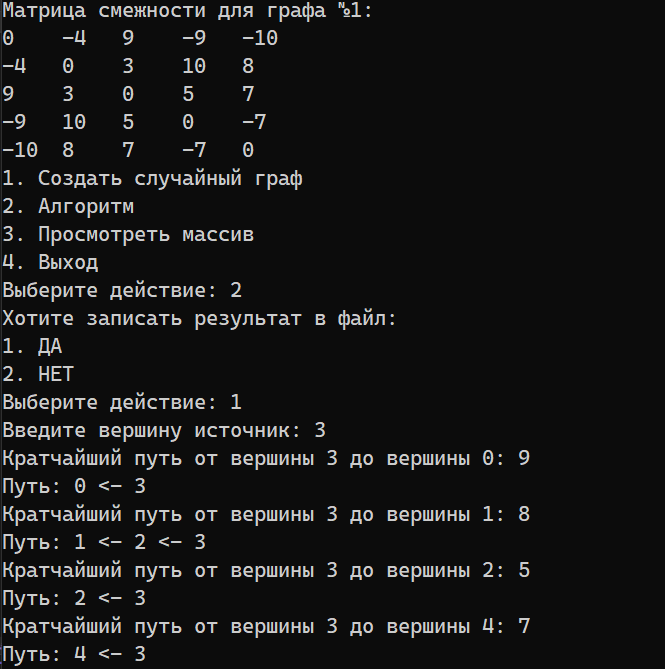


Рисунок 1–Результат работы программы

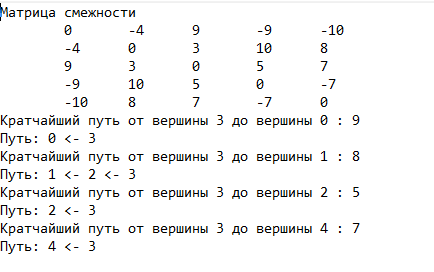


Рисунок 2- Cохранённый в файлерезультат

# Тестирование

Среда разработки MicrosoftVisualStudio 2022 предоставляет все средства, необходимые при разработке и отладке многомодульнойпрограммы.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования было выявлено и исправлено множество проблем, изменением дизайна выводимых данных, алгоритмом программы, взаимодействием функций.Задали размеры графа – N= 5 и вершину источника– M = 3.

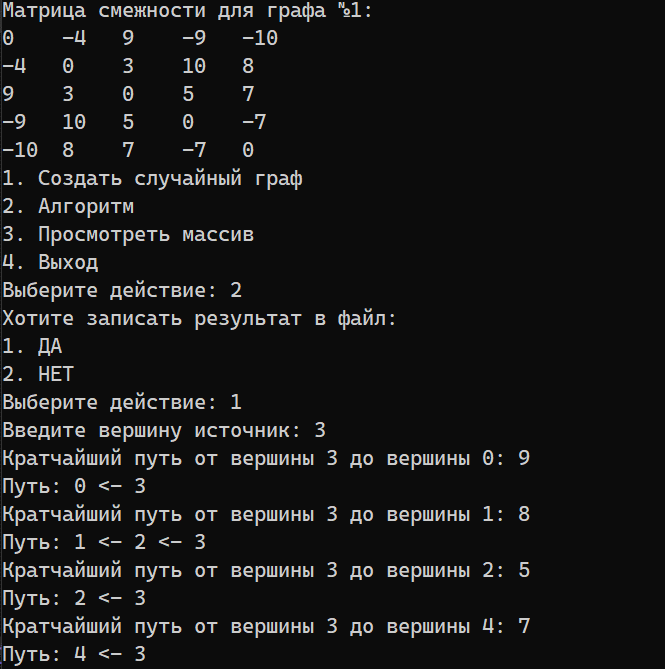


Рисунок 3- Результат тестирования приM = 3

На рисунке 3 продемонстрирован результат тестирования при размерах M = 3. Наш результат записался в файл.

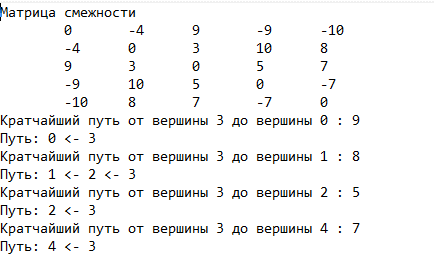


Рисунок 4- Результат, записанный в файл

Задали вершину источника – M = 1.

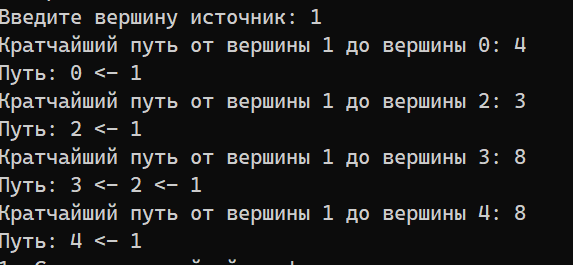


Рисунок 5- результат тестирования при размерах M = 1

На рисунке 5продемонстрирован результат тестирования при размерахM = 1. Наш результат записался в файл.

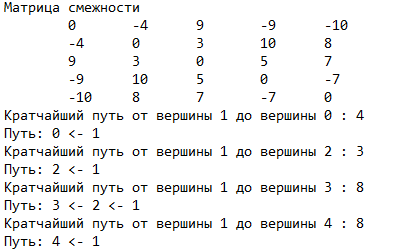


Рисунок 6 - Результат M = 1, записанный в файл

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Запуск программы | Запуск окна | + |
| Работа алгоритма | Находит кратчайший путь | + |
| Запись в файл | Записывает результат в файл | + |

Таблица 1 – Описание поведения программы при тестировании

1. **Ручной расчёт задачи**

Проведем проверку программы посредством ручных вычислений на примере матрицы размером 4x4.

Начинаем поиск кратчайшего пути. Проверяем каждый возможный путь и ищем минимальный.

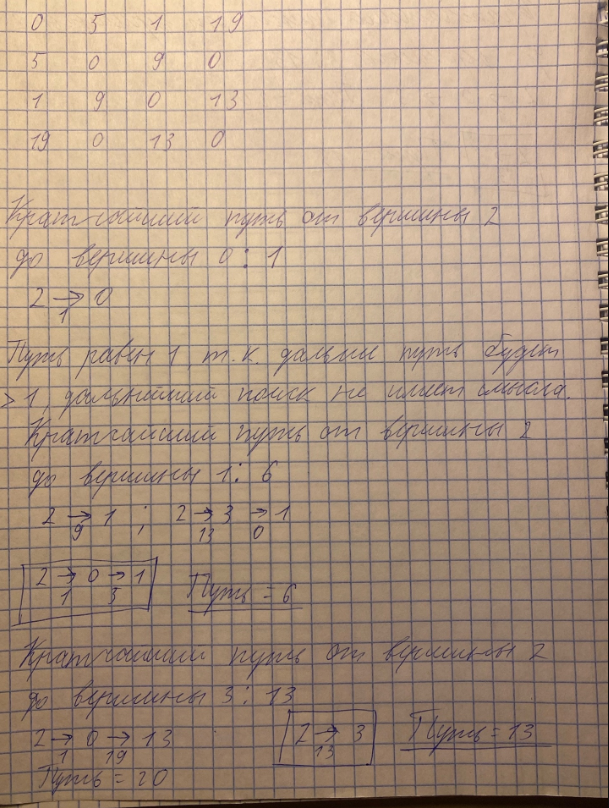


Рисунок 7- ручной расчёт результата

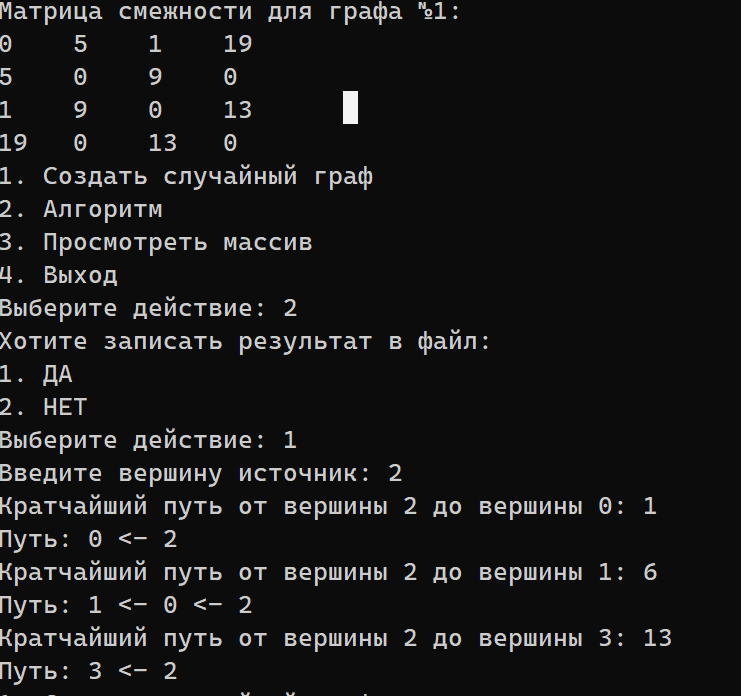


Рисунок 8- результат работы программы

Сравнив результат работы программ и ручной расчёт, можно сделать вывод, что программа работает верно.

# Заключение

В ходе данного курсового проекта по разработке программы для поиска кратчайших путей во взвешенном графе с использованием алгоритма Форда-Беллмана были достигнуты следующие результаты.

Программа предоставляет удобный интерфейс для ввода взвешенного графа, осуществляет эффективную реализацию алгоритма Форда-Беллмана и выводит результаты в виде кратчайших путей от заданной вершины до всех остальных. Также предоставляется опция сохранения результатов в файл, что обеспечивает удобство дальнейшего анализа.

Программа взаимодействует с пользователем, предоставляя выбор записи результатов в файл, что повышает ее гибкость в использовании. Реализованы проверки на ошибочные сценарии, такие как некорректный ввод данных или невозможность выделения памяти.

Основной принцип работы алгоритма Форда-Беллмана успешно реализован, и программа демонстрирует корректное определение кратчайших путей в графе. Это подтверждается как тестовыми примерами, так и тестированием на различных случаях взвешенных графов.

Таким образом, разработанная программа предоставляет эффективный и удобный инструмент для нахождения кратчайших путей во взвешенных графах с использованием алгоритма Форда-Беллмана, и ее функциональность полностью соответствует поставленной задаче

**Список литературы**

1. Седжвик Р., Уэйн К. "Алгоритмы. Построение и анализ." М.: Вильямс, 2018. - Рекомендуемая глава: "Графы и алгоритмы на графах."

2. Хоровиц Д. "Алгоритмы. Справочник с примерами на C++." Питер, 2017. - Особенно главы, посвященные алгоритмам поиска путей.

3. Кормен Т., Лейзерсон Ч., Ривест Р. "Алгоритмы: Построение и анализ." М.: МЦНМО, 2001. - Главы, связанные с алгоритмами поиска.

4. Шилдт Г. "Полный справочник по C++." Вильямс, 2006. - Разделы, охватывающие структуры данных и алгоритмы на C++.

5. Дейтел Х., Дейтел П. "Как программировать на C/C++." Питер, 2013. - Главы, касающиеся разработки алгоритмов на C++.

6. Лафоре Р. "Объектно-ориентированное программирование в C++." Питер, 2002. - Практические примеры реализации алгоритмов на C++.

7. Макконнелл С. "Совершенный код. Мастер-класс." Питер, 2014. - Рекомендации по эффективной реализации алгоритмов.

# 

# Приложение А.

Листинг программы.

// test.cpp: определяет точку входа для консольного приложения.

//

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <fstream>

using namespace std;

void ford(int\*\* arr, int rows) {

int choice;

bool save = false;

printf("Хотите записать результат в файл: \n");

printf("1. ДА \n");

printf("2. НЕТ \n");

printf("Выберитедействие: ");

for (;;) {

cin >> choice;

if (cin.fail() || choice < 1 || choice > 2) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный номер! Повторите попытку: ";

}

else { break; }

}

if (choice == 1)

{

save = true;

}

int\* dist = (int\*)malloc(rows \* sizeof(int));

int\* parent = (int\*)malloc(rows \* sizeof(int));

int\*\* temp = 0;

temp = (int\*\*)malloc(rows \* sizeof(int\*));

if (temp == NULL) {

printf("Не удалось выделить память!\n");

return;

}

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

temp[i] = (int\*)malloc(rows \* sizeof(int));

}

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

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

temp[i][j] = arr[i][j]; // на главной диагонали нули

}

}

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

dist[i] = INT\_MAX;

parent[i] = -1;

}

int src;

printf("Введите вершину источник: ");

for (;;) {

cin >> src;

if (cin.fail() || src > rows - 1 || src < 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный номер вершины графа! Повторите попытку: ";

}

else { break; }

}

dist[src] = 0;

// Реализация алгоритма Форда-Беллмана

for (int k = 0; k < rows - 1; k++) {

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

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

temp[i][j] = (temp[i][j] < 0) ? -temp[i][j] : temp[i][j];

if (temp[i][j] != 0 && dist[i] != INT\_MAX && dist[i] + temp[i][j] < dist[j]) {

dist[j] = dist[i] + temp[i][j];

parent[j] = i;

}

}

}

}

// Выводкратчайшихпутей

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

if (i != src) {

printf("Кратчайший путь от вершины %d до вершины %d: %d\n", src, i, dist[i]);

printf("Путь: %d", i);

int p = parent[i];

while (p != -1) {

printf(" <- %d", p);

p = parent[p];

}

printf("\n");

}

}

if (save == true)

{

ofstream file("result.txt", ios\_base::out);

if (file.is\_open())

{

file << "Матрицасмежности " << '\n';

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

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

file << "\t" << arr[i][j] << " ";

}

file << endl;

}

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

{

if (i != src)

{

file << "Кратчайшийпутьотвершины " << src << " довершины " << i << " : " << dist[i] << endl;

file << "Путь: " << i;

int p = parent[i];

while (p != -1)

{

file << " <- " << p;

p = parent[p];

}

file << "\n";

}

}

file.close();

}

}

free(dist);

free(parent);

}

void main() {

setlocale(LC\_ALL, "RUS");

srand(time(NULL));

int\*\* arr = 0; // матрицасмежности

int i, j, rows = 0;

FILE\* file;

char name[255];

int choice = 0, rows1, rows2, what;

printf("Курсовая работа\nПо дисциплине: \"Логика и основы алгоритмизации в инженерных задачах\"\nНа тему: \"Реализация алгоритма Форда-Беллмана\"\n\nВыполнил студент группы 22ВВВ1: Кирилин Матвей\nПринял: к.т.н. доцент Юрова О.В.\n\n");

system("PAUSE");

system("cls");

do {

printf("1. Создать случайный граф\n");

printf("2. Алгоритм\n");

printf("3. Просмотреть массив\n");

printf("4. Выход\n");

printf("Выберите действие: ");

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

switch (choice) {

case 1: // Создать случайный граф

printf("Введите количество вершин графа: ");

for (;;) {

cin >> rows;

if (cin.fail() || rows < 2) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Некорректный номер количество вершин графа! Повторитепопытку: ";

}

else { break; }

}

cout << endl;

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

if (arr == NULL) {

printf("Не удалось выделить память!\n");

break;;

}

for (i = 0; i < rows; i++) {

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

}

// генерировать случайные значения для матрицы смежности

for (i = 0; i < rows; i++) {

for (j = i; j < rows; j++) {

if (i == j) {

arr[i][j] = 0; // на главной диагонали нули

}

else {

arr[i][j] = - 0 + rand() % 21; // случайные значения 0 или 1

arr[j][i] = arr[i][j]; // симметрично заполнять значения для неориентированного графа

}

}

}

// выводить матрицу смежности на экран

printf("Матрица смежности для графа №1:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

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

}

printf("\n");

}

break;

case 2: // alg

if (arr == NULL) {

printf("Сначала создайте граф!\n");

break;

}

ford(arr, rows);

break;

case 3: // Просмотреть массив

if (arr == NULL) {

printf("Сначала создайте граф!\n");

break;

}

// выводить матрицу смежности на экран

printf("Матрица смежности для графа:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < rows; j++) {

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

}

printf("\n");

}

break;

case 4: // Выход

system("cls");

printf("До свидания!\n");

break;

default:

printf("Некорректный выбор!\n");

break;

}

} while (choice != 4);

}