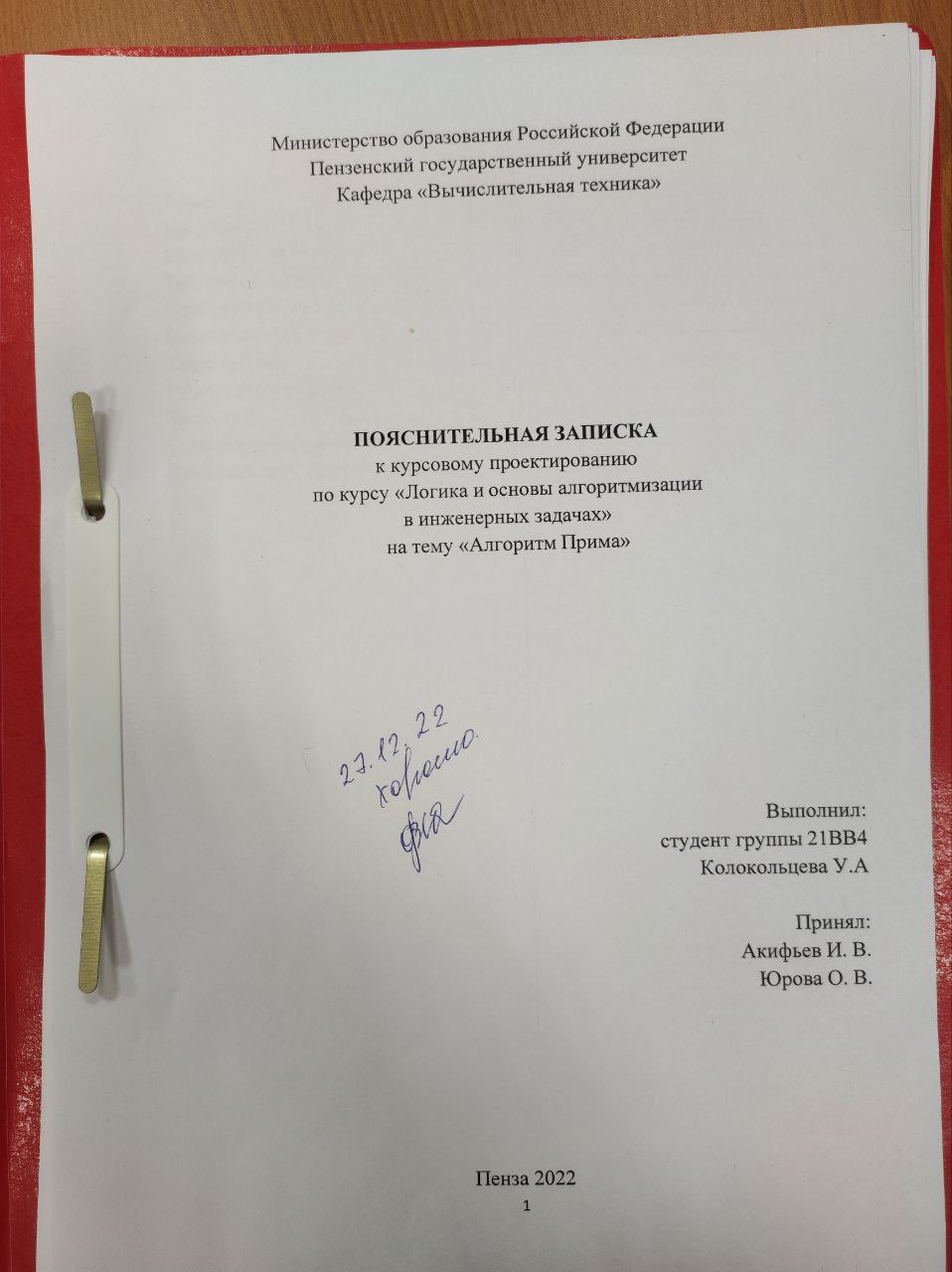
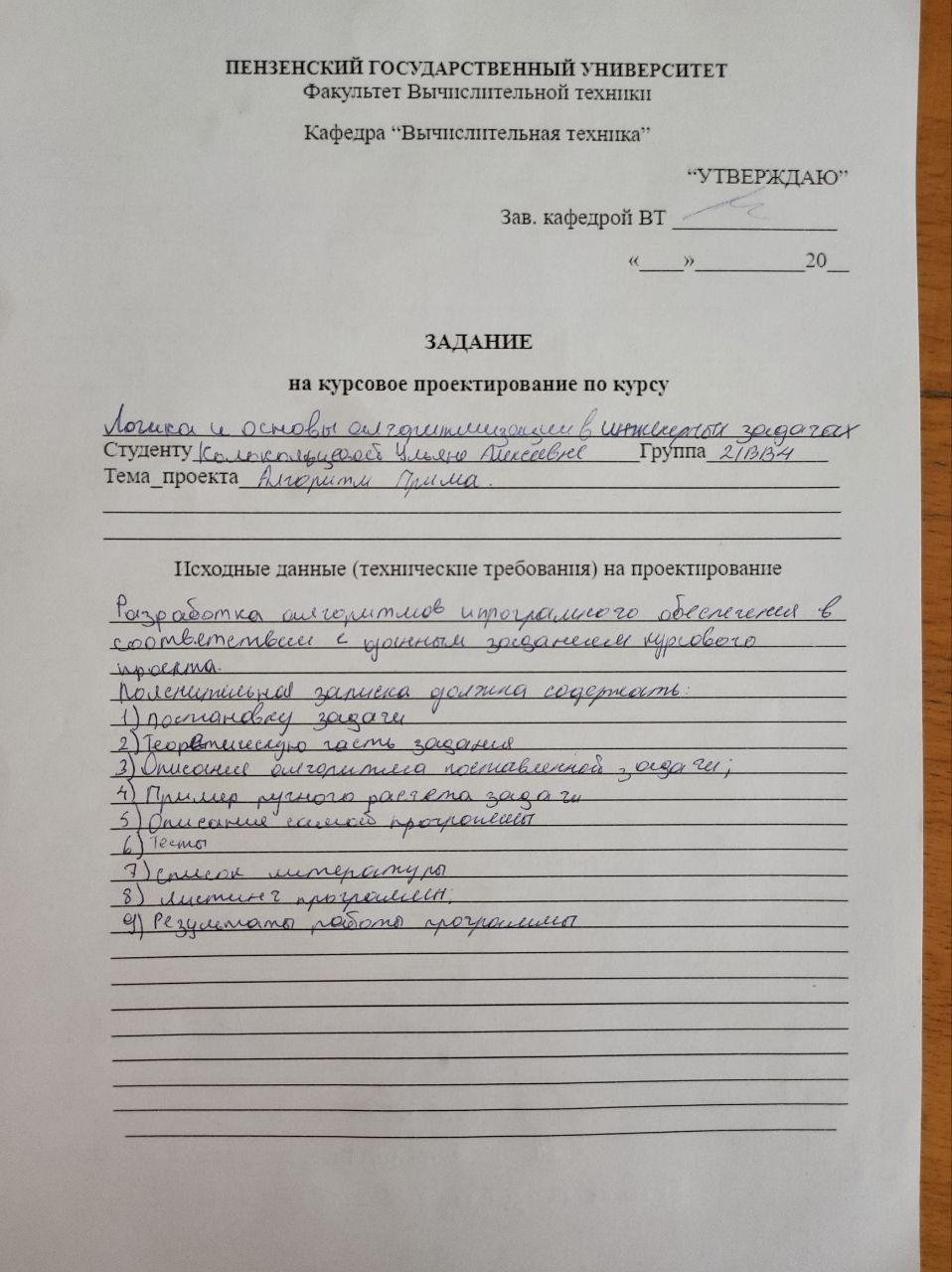
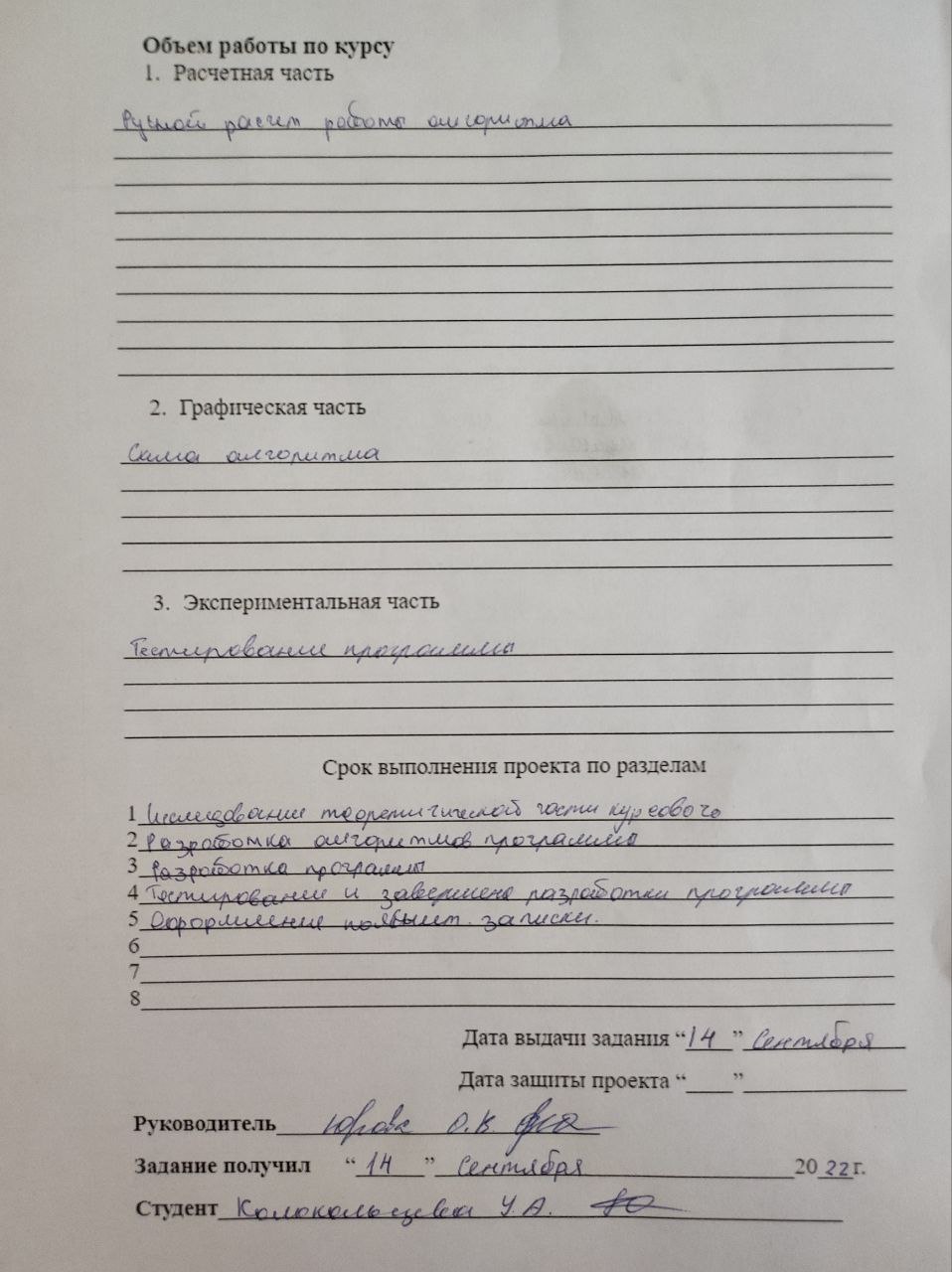
****

****

****

**Содержание**

Реферат………………………………………………………………………….5

Введение……………………………………………………………………… ..6

Постановка задачи……………………………………………………………...7

Теоретическая часть задания……………..……………………………………8

Описание алгоритма программы……………………………………………....9

Описание программы…………………………………………………...………11

Тестирование…………………………………………………...……………....17

Ручной расчет задачи……………………………………..……………............19

Заключение……………………………………………………………………..20

Список используемых источников……………………………………………21

Приложение А. Листнг программы…………………………………………...22

**Реферат**

Отчет 27 стр, 8 рисунков.

ГРАФ, ТЕОРИЯ ГРАФОВ, НЕОРИЕНТИРОВАННЫЙ ГРАФ, ВЗВЕШЕННЫЙ ГРАФ, ОСТОВНОЕ ДЕРЕВО.

Цель исследования – разработка программы, осуществляющей нахождение минимального остовного дерева.

В работе рассмотрен алгоритм Прима, на основе которого находятся подмножества ребер графа, который формирует дерево, включающее в себя каждую вершину, а также имеет минимальную сумму весов среди всех деревьев, которые могут быть сформированы из графа.

**Введение**

Одной из важнейших задач, возникающих при проектировании сетей связи – определение кратчайших путей между всеми парами узлов сети, например, в логистике это поможет сократить затраты на перевозки гру, электронике и схемотехнике. Дискретную математику сейчас трудно представить без оптимизационных алгоритмов Краскала и Прима на графах. Но это началось только в 60-е гг. XX века. Любопытно, что история этих алгоритмов началась до появления книги Д. Кёнига «Теория конечных и бесконечных графов», изданной в Лейпциге в 1936 году. Предполагается что существует несколько вариантов путей соединяющих пару вершин, а выбран один из них – кратчайший по длине.

**Постановка задачи**

Требуется разработать программу, осуществляющую нахождение минимального остовного дерева в заданном графе, используя алгоритм Прима.

Исходный граф в программе должен задаваться матрицей смежности. Программа должна работать так, что пользователь должен написать, в какой файл нужно записать результат, после чего он мог выбрать способ вывода: ввод элементов матрицы в консоле или рандомная генерация матрицы. После обработки данных пользователю нужно ввести количество вершин в матрице смежности. А затем программа должна находить минимальное остовое дерево.

Исходный граф взвешенный, неориентированный и не имеет петель.

Устройство ввода – клавиатура и мышь.

**Теоретическая часть задания**

Граф G называется деревом, если он является связным и не имеет циклов. Остовом связного графа G называется любой его подграф, содержащий все вершины графа G и являющийся деревом. Остовное дерево связного нагруженного графа G с минимальной суммой длин содержащихся в нем ребер, будем называть остовом минимального веса или минимальным остовным деревом (МОД).

Алгоритм Прима начинается с выбора произвольной вершины связного конечного неориентированного графа без петель, ребра которого имеют ненулевой вес. Далее выбирается инцидентное выбранной вершине ребро с минимальным весом. Это ребро является началом построения минимального остовного дерева. Последующие шаги сводятся к присоединению к построенному на предыдущем шаге дерева нового ребра, один из концов которого принадлежит уже построенному дереву, а другой не принадлежит, и при этом новое ребро имеет минимальный вес среди всех ребер, не принадлежащих уже построенному дереву.

**Описание алгоритма программы**

Для начала, пользователь должен вписать полное имя файла, в который будет помещен результат, даже если такого файла не существует, он будет создан. После чего ему нужно ознакомиться с предложениями по созданию матрицы смежности и ввести размер графа.

Затем, вызывается функция zero\_matrix, которая при выводе нулевой матрицы дает нам предупреждение «Отсутствуют ребра». Она считает количество нулей в матрице, и если оно равно числу всех элементов, то выходит соответствующее предупреждение.

Псевдокод:

1.нач

2. zero = 0;

3.цикл от i=0 до i<n [шаг i++]

4. цикл от j=0 до j<n [шаг j++]

5. если (matrix[i][j] == 0) , то zero++;

6. конец условия

7. конец цикла

8. если (zero == n \* n), то

9.Вывод "Ребра отсутствуют!;

10.выход

11. конец условия

12. конец цикла

**Описание программы**

Для написания данной программы использован язык программирования Си. Си - язык программирования общего назначения, который очень известен своей эффективностью, экономичностью ресурсов и переносимостью на других программные платформы. Все эти преимущества обеспечивают качество разработки программного обеспечения, их быстроту и сравнительно небольшой размер.

В качестве среды программирования был выбран программный продукт Microsoft Visual Studio 2019. Проект был создан в виде консольного приложения Win32 (Visual C++).

Работа программы начинается с написания файла, в который будет записан результат, после чего будет выведено небольшое меню, в котором предоставлен выбор метода вывода матрицы.

cout << " Алгоритм Прима\n\n";

cout << " В какой файл вы хотите занести результат?\n";

cin >> s;

cout << " Каким образом вы хотите создать матрицу смежности?\n";

cout << " 1. Ручной ввод\n" << " 2. Рандомная генерация\n";

cin >> k;

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

cout << " Ошибка!Выберите число из предоставленных 2 ";

return -1;

}

Если выбран 1 вариант, то мы создаем матрицу, вручную вводим каждый ее элемент в консоли и сохраняем в файле, указанном ранее

if (k == 1) {

cout << " Введите размер графа\n";

cin >> n;

if (n < 1) {

cout << " Ошибка! Выберите число больше 0\n";

return -1;

}

cout << " Введите матрицу смежности\n";

//инициализирует вершину для отслеживания

choosen = new bool[n];

for (int i = 1; i < n; i++)

choosen[i] = false;

choosen[0] = true; // выбирает 1 вершину

int x, y; // номера строк и столбцов

// создаем массив

array = new int\* [n];

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

array[i] = new int[n];

// вводим значения

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

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

{

cin >> tmp;

array[i][j] = tmp;

{

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << array[i][j] << " \n ";

}

}

}

Если же 2, то создаем матрицу с рандомными значениями и сохраняем в файле

if (k == 2) {

cout << " Введите размер графа\n";

cin >> n;

int\*\* matrix = new int\* [n]; // Генерируем матрицу M1

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

{

matrix[i] = new int[n];

}

int\*\* A = new int\* [n];

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

{

A[i] = new int[n];

}

srand(time(NULL));

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

{

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

{

matrix[i][j] = rand() %30;

if (i == j)

{

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] <= 5)

{

matrix[i][j] = 0;

}

}

}

cout << " Матрица смежности: \n";

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

{

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

{

matrix[i][j] = matrix[j][i];

printf(" %2d\t", matrix[j][i]); // Выводим элементы матрицы M1

{

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << matrix[i][j] << " \n ";

}

}

}

cout << endl;

}

Далее мы производим нахождение минимального остовного дерева, выводим результат в консоль и записываем результат в файл.

cout << " Результат\n";

while (edgeNum < n - 1) {

x = 0;

y = 0;

int min = INT\_MAX; // установим наибольшее число

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

if (choosen[i]) {

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

{

if (!choosen[j] && array[i][j]) {

if (min > array[i][j]) {

x = i;

y = j;

min = array[i][j];

}

}

}

}

}

cout << '(' << x + 1 << " -> " << y + 1 << ") " << array[x][y] << " | ";

choosen[y] = true;

++edgeNum;

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << '(' << x + 1 << " -> " << y + 1 << ") " << array[x][y] << " | ";

}

fout.close();

}

0;

Ниже можно увидеть оформление начального запроса и дальнейшие действия с ним.

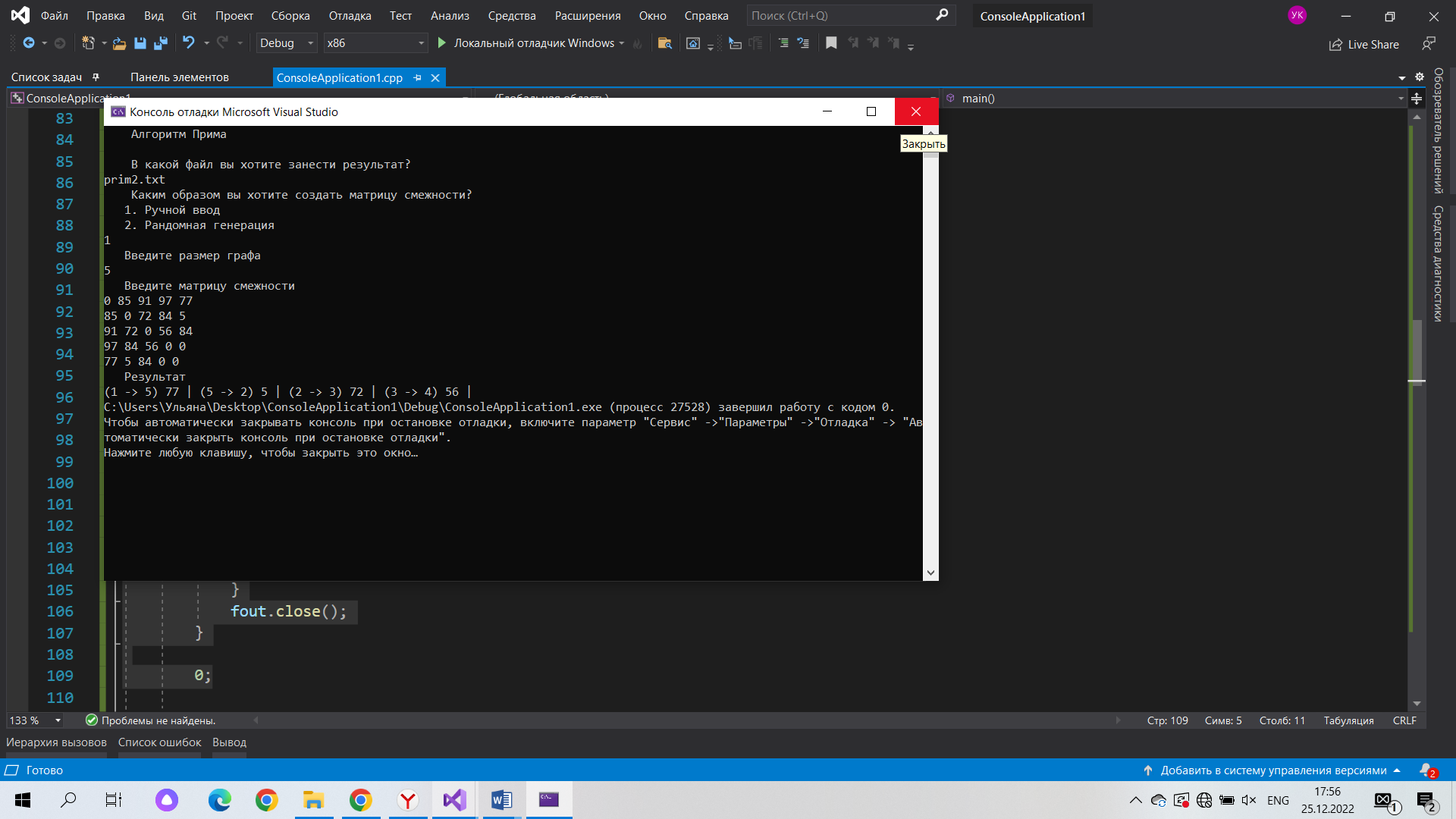
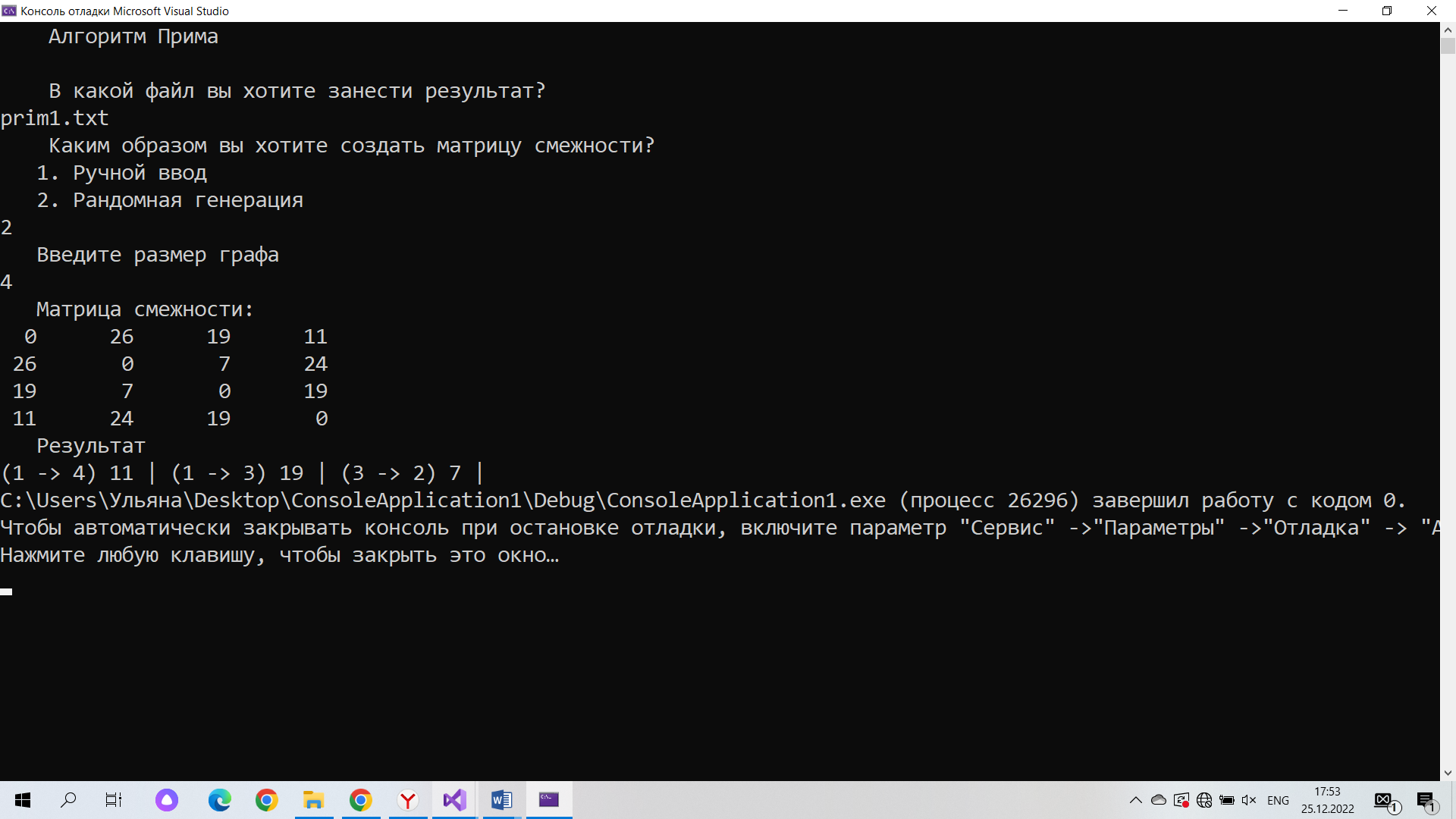


Рисунок 1 – Ручной ввод матрицы

  
Рисунок 2 – Случайная генерация матрицы

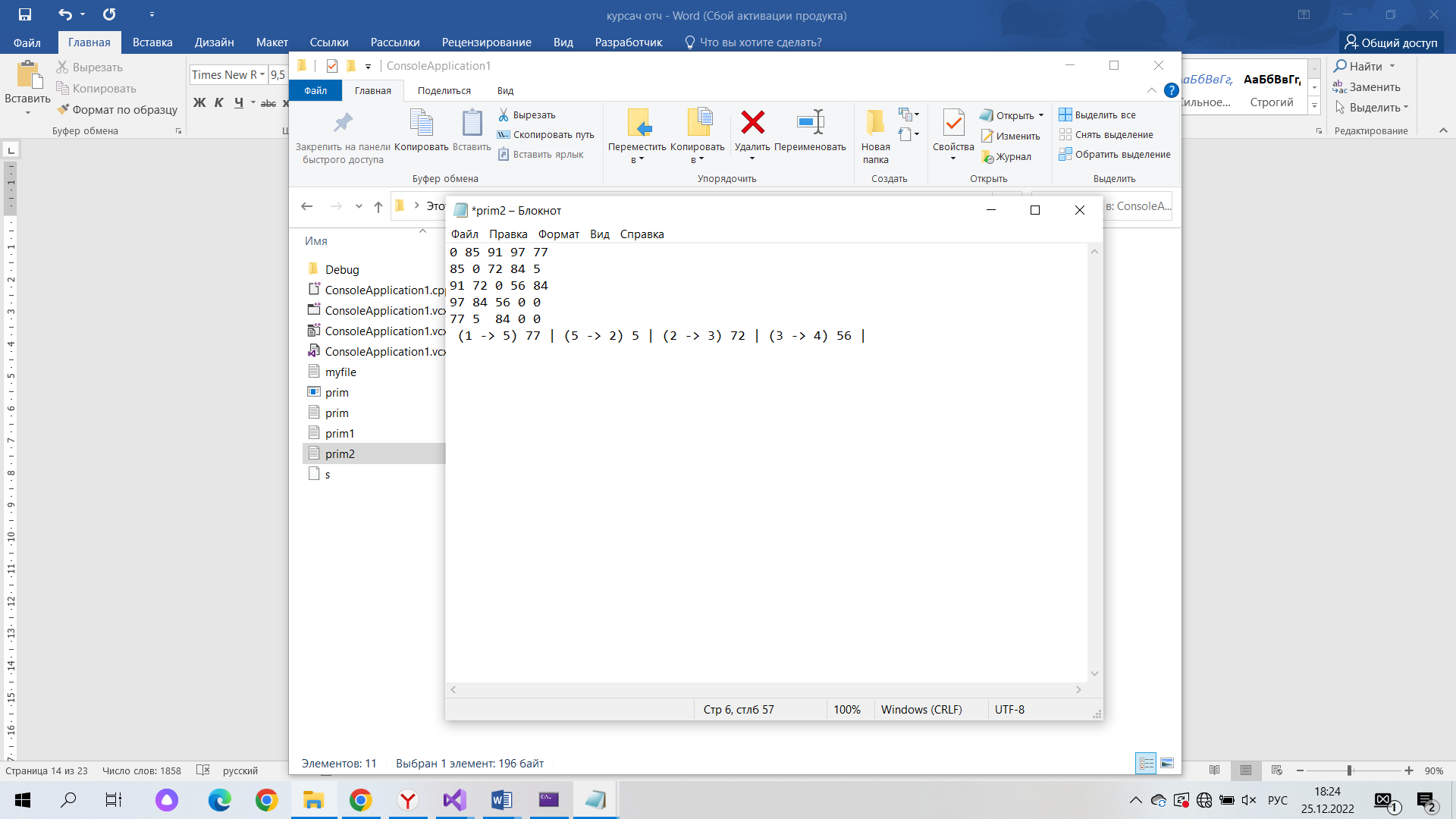
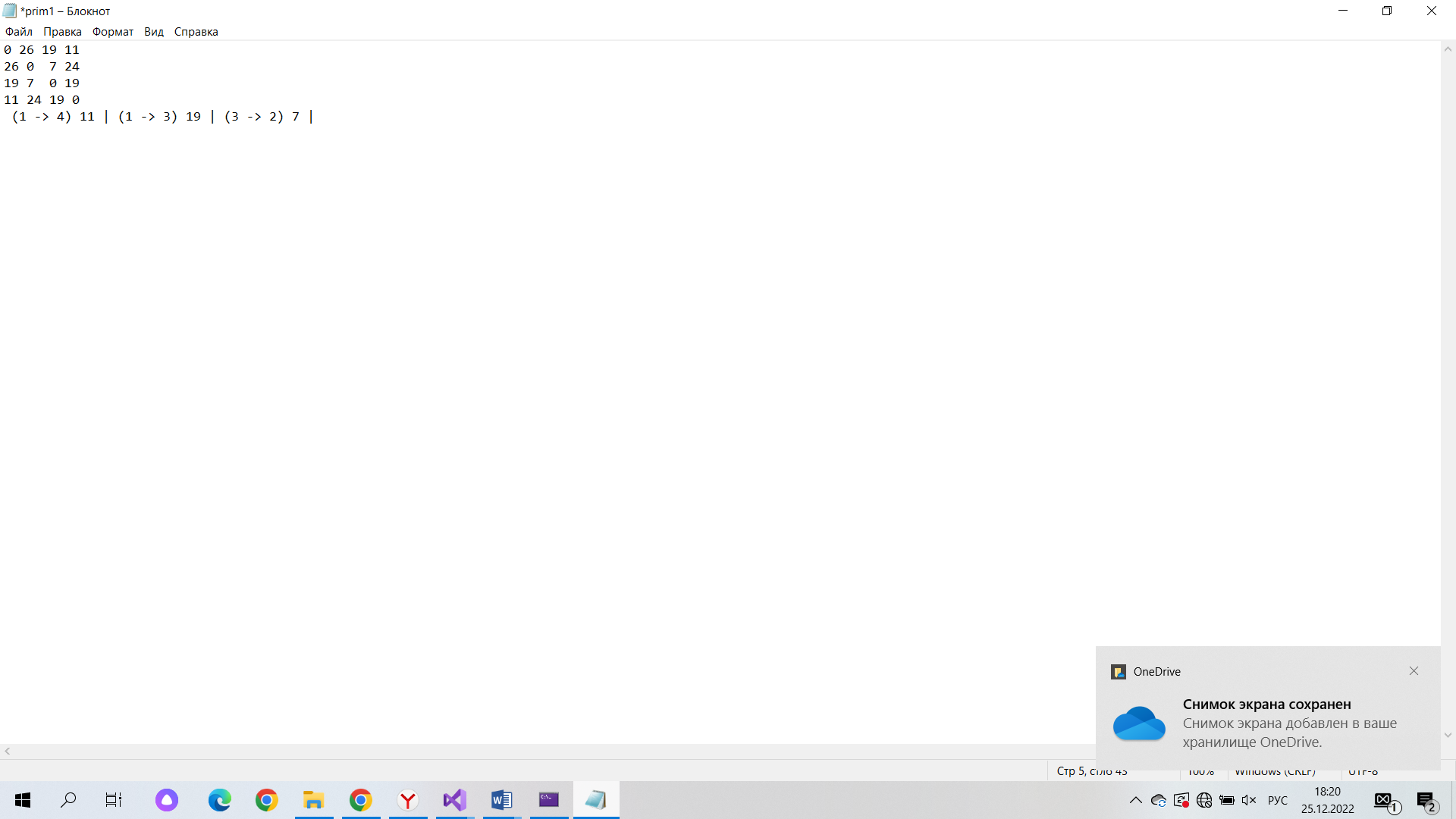


Рисунок 3 – Файл результатов ручного ввода

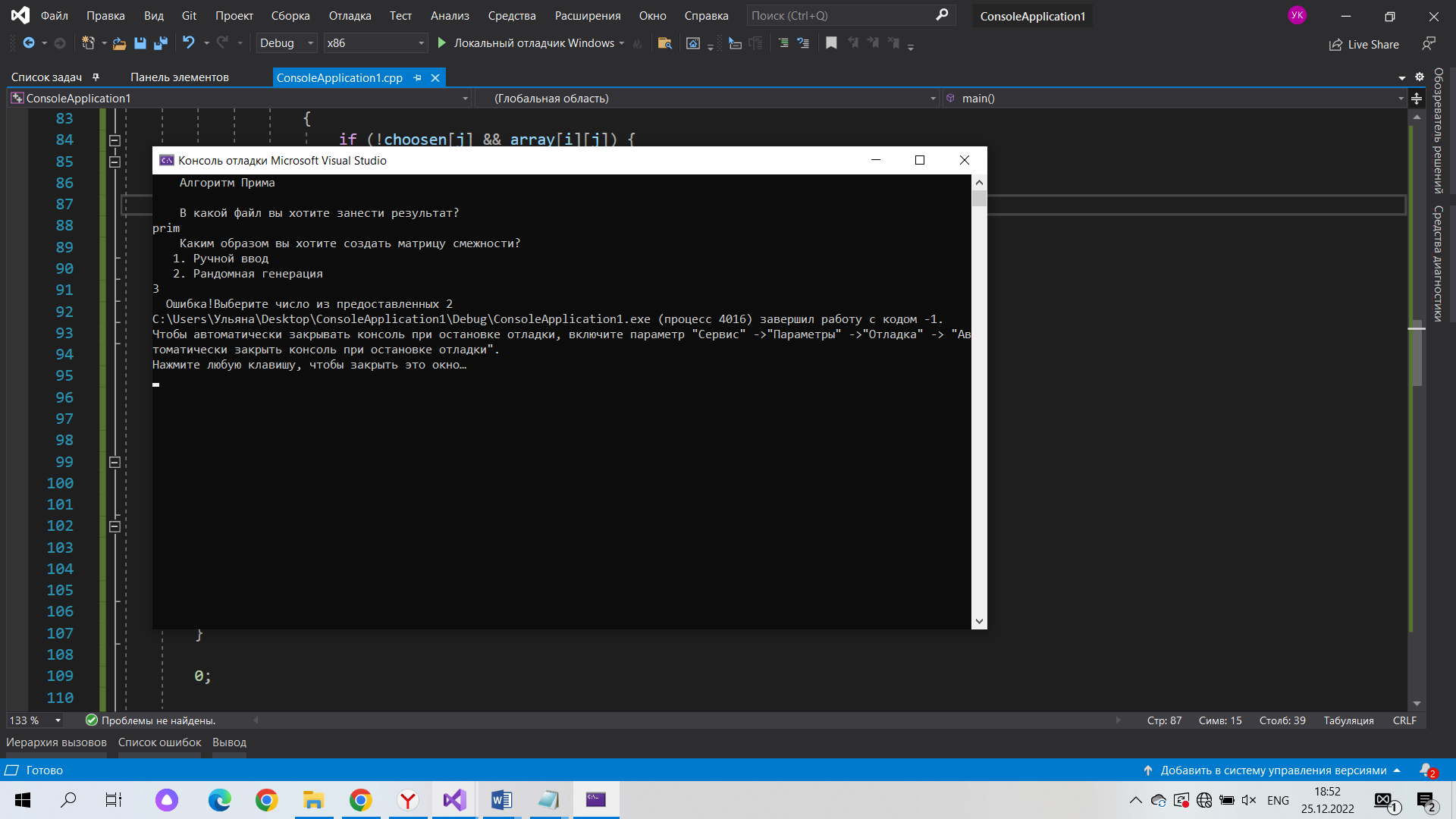
Рисунок 4 – Файл результатов рандомной генерации

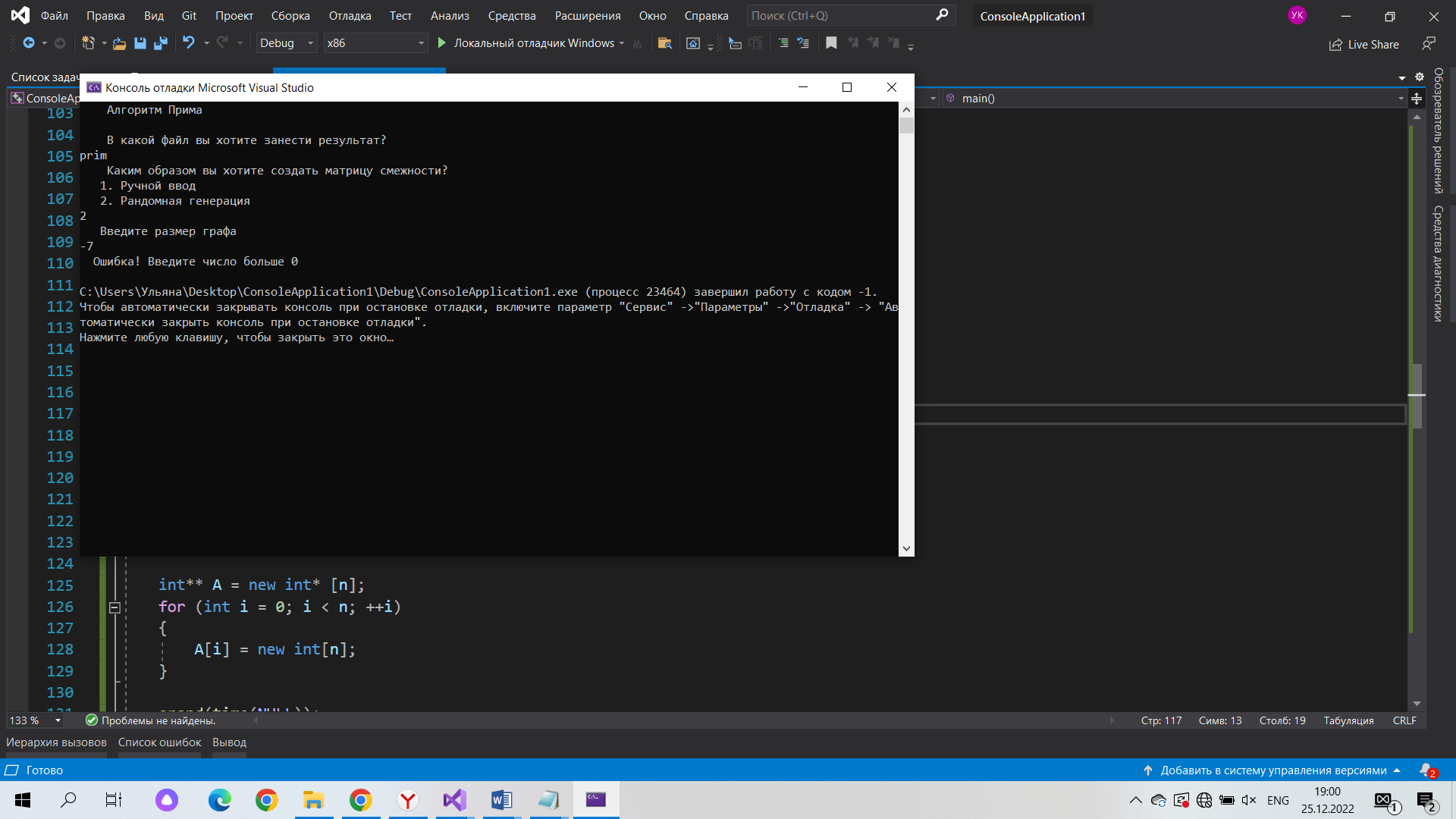
**Тестирование**

Среда разработки Microsoft Visual Studio 2019 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. Также были предусмотрены выводы предупреждений об ошибках при недопустимых действиях пользователя.

Ниже продемонстрирован результат тестирования программы.

  
Рисунок 5 – Тестирование при некорректном вводе пункта меню

  
Рисунок 6– Тестирование при некорректном вводе размера графа

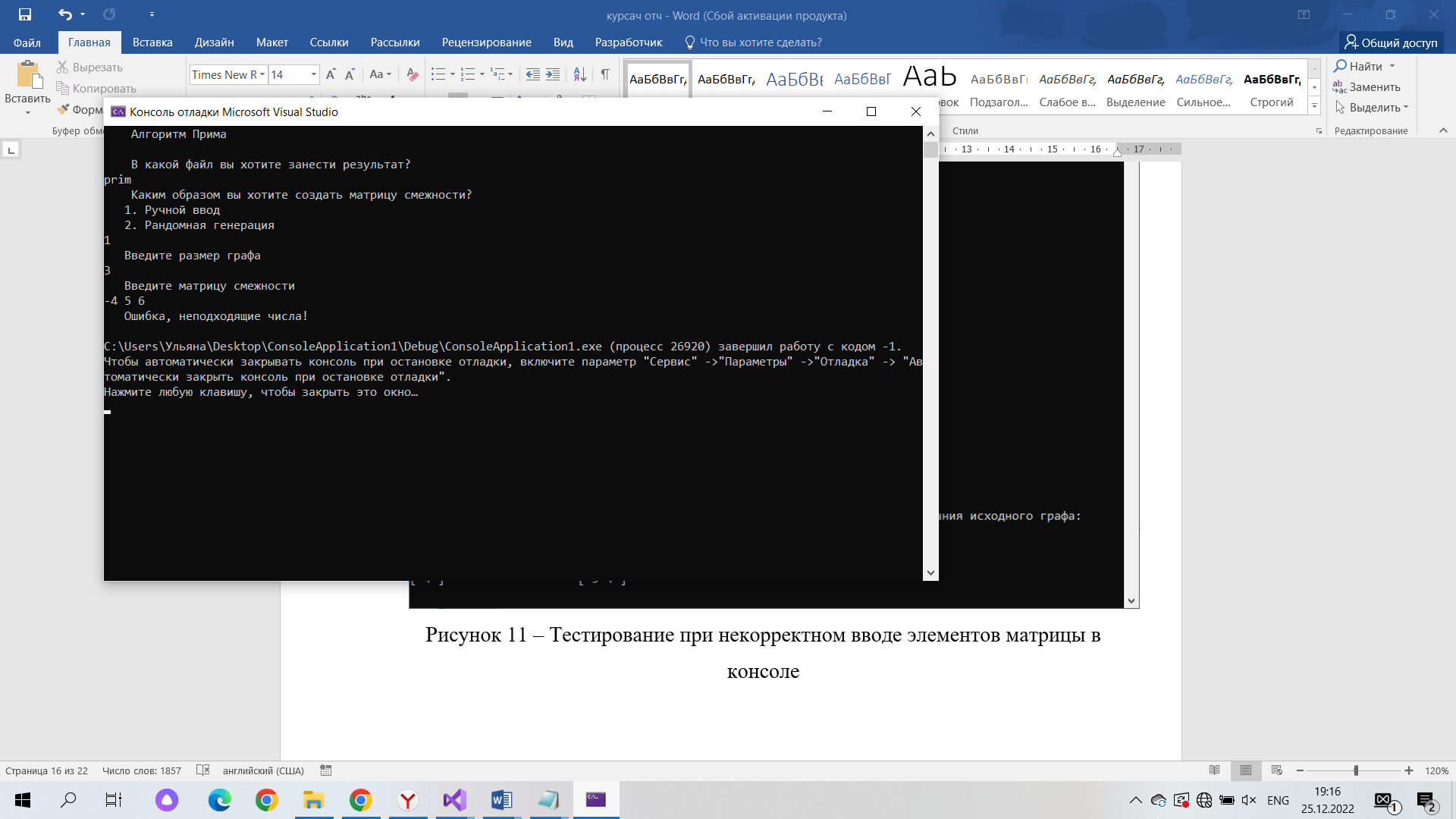
  
Рисунок 7 – Тестирование при некорректном вводе элементов матрицы в консоле

Таблица 1 – Описание поведения программы при тестировании

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Полученный результат |
| Некорректный ввод пункта меню | Сообщение об ошибке с просьбой попробовать заново | Верно |
| Некорректный ввод размера матрицы | Сообщение об ошибке с просьбой попробовать заново | Верно |
| Некорректный ввод элементов матрицы в консоле | Сообщение об ошибке, ввести еще раз | Верно |
| Некорректный вывод  Матрицы( нулевая матрица) | Сообщение о том, что ребра отсутствуют | Верно |

В результате тестирования было выявлено, что программа успешно проверяет на соответствие необходимым требованиям.

**Ручной расчет задачи**

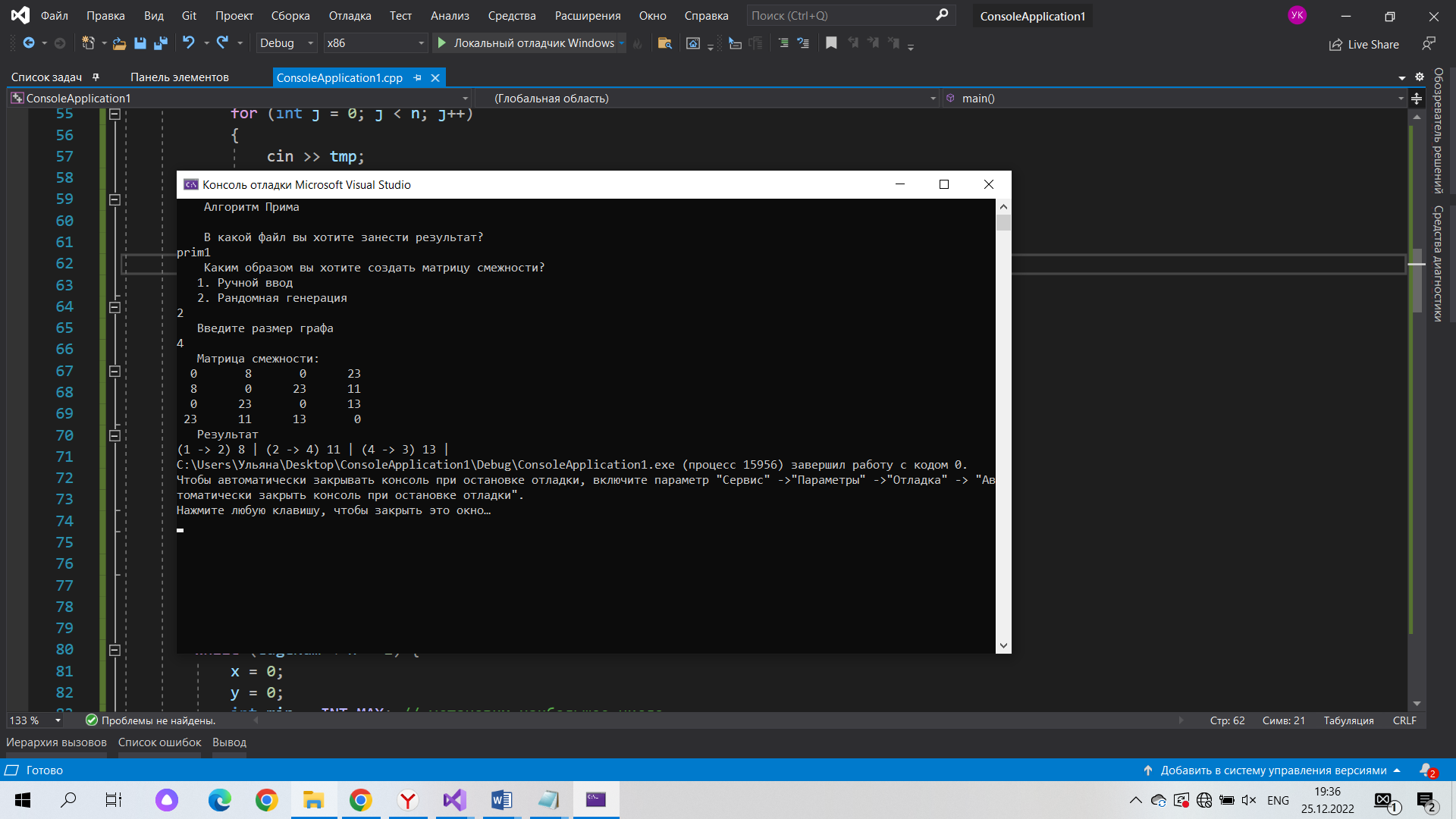


Рисунок 8 – Проверка на правильность работы программы

Матрица смежности вершин:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** |
| **1** | 0 | 8 | 1 | 23 |
| **2** | 8 | 0 | 23 | 11 |
| **3** | 0 | 23 | 0 | 13 |
| **4** | 23 | 11 | 13 | 0 |

1

Граф: 23

8

2 4

11

23 13

3

Найдем наименьший путь остовного дерева:

1-2 (8) 2-4 (11) 4-3 (13)

Таким образом, можно сделать вывод, что программа работает верно

**Заключение**

Таким образом, в процессе создания данного проекта разработана программа, реализующая алгоритм нахождения минимального остовного пути среди всех остальных в Microsoft Visual Studio 2019 C++ 6.0.

При выполнении данной курсовой работы были получены навыки разработки программ и освоены приемы создания матриц, основанных на теории графов. Углублены знания языка программирования C.

Недостатком разработанной программы является примитивный пользовательский интерфейс. Это связано с тем, что программа работает в консольном режиме, не добавляющем к сложности языка сложность программного оконного интерфейса.

**Список используемых источников**

1. Википедия, «Алгоритм Прима»
2. Методические указания по курсу «Логика и основы алгоритмизации в инженерных задачах»
3. Википедия, «Остовное дерево»

**Приложение А.**

**Листинг программы.**

#include <iostream>

#include <clocale>

#include<fstream>

#include<string>

using namespace std;

void zero\_matrix(int\*\* matrix, int n);

void zero\_matrix(int\*\* matrix, int n)

{

int zero = 0;

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

{

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

{

if (matrix[i][j] == 0) zero++;

}

}

if (zero == n \* n)

{

printf("\nРебра отсутствуют!\n");

exit(1);

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int n, tmp, k;

int\*\* array;

bool\* choosen;

int edgeNum = 0;

string s;

cout << " Алгоритм Прима\n\n";

cout << " В какой файл вы хотите занести результат?\n";

cout << " Имя файла: ";

cin >> s;

cout << " Каким образом вы хотите создать матрицу смежности?\n";

cout << " 1. Ручной ввод\n" << " 2. Рандомная генерация\n";

cout << " Ваш выбор: ";

cin >> k;

if ((k < 1) || (k > 2))

{

do

{

cout << "\nВведите один из 2 вариантов \n";

cin >> k;

if ((k < 1) || (k > 2))

{

cout<< "Невозможно считать данные! Попробуйте заново!\n";

}

} while ((k < 1) || (k > 2));

}

if (k == 1) {

cout << " Введите размер графа\n";

cout << " Ваш выбор: ";

cin >> n;

if (n < 2) {

do

{

cout << "\nВведите число больше 1\n";

cin >> n;

if (n < 2)

{

cout << "Невозможно считать данные! Попробуйте заново!\n";

}

} while (n < 2);

}

cout << " Введите матрицу смежности\n";

//инициализирует вершину для отслеживания

choosen = new bool[n]; //ребро, которое соединяет две вершины

for (int i = 1; i < n; i++)

choosen[i] = false;

choosen[0] = true; // выбирает 1 вершину

int x, y; // номера строк и столбцов

// создаем массив

array = new int\* [n];

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

array[i] = new int[n];

// вводим значения

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

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

{

cin >> tmp;

array[i][j] = tmp;

if (tmp < 0) {

do

{

cout << "\nВведите положительные числа \n";

cin >> tmp;

if (tmp < 0)

{

cout << "Невозможно считать данные! Попробуйте заново!\n";

}

} while (tmp < 0);

}

{

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << array[i][j] << " \n ";

}

}

}

cout << " Результат\n";

while (edgeNum < n - 1) { //edgeNum - выбираем вершину

x = 0; //вершина 1

y = 0; // веришны 2

int min = INT\_MAX; // установим наибольшее число для того, чтобы сравнить с наименьшим весом ребра

//определяются веса между выбранной вершиной edgeNum и остальными невыбранными вершинами

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

if (choosen[i]) {

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

{

if (!choosen[j] && array[i][j]) {

if (min > array[i][j]) { // выбираем выршину с наименьшим весом

//фиксируем выбранные ребра и вес

x = i;

y = j;

//устанавливаем минимальный вес

min = array[i][j];

}

}

}

}

}

cout << '(' << x + 1 << " -> " << y + 1 << ") " << array[x][y] << " | "; // выводим веришны на экран и вес ребра между этими вершинами

choosen[y] = true; // выбранную вершину исключаем из перечня

++edgeNum; // число выбранных вершин увеличиваем на 1

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << '(' << x + 1 << " -> " << y + 1 << ") " << array[x][y] << " | ";

}

fout.close();

}

0;

}

if (k == 2) {

cout << " Введите размер графа\n";

cout << " Ваш выбор: ";

cin >> n;

if (n < 2) {

do

{

cout << "\nВведите число больше 1\n";

cin >> n;

if (n < 2)

{

cout << "Невозможно считать данные! Попробуйте заново!\n";

}

} while (n < 2);

}

int\*\* matrix = new int\* [n]; // Генерируем матрицу M1

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

{

matrix[i] = new int[n];

}

int\*\* A = new int\* [n];

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

{

A[i] = new int[n];

}

srand(time(NULL));

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

{

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

{

matrix[i][j] = rand() % 30;

if (i == j)

{

matrix[i][j] = 0;

}

if (matrix[i][j] >= 0 && matrix[i][j] < 5)

{

matrix[i][j] = 0;

}

}

}

zero\_matrix(matrix,n);

cout << " Матрица смежности: \n";

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

fout << " Матрица смежности: \n";

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

{

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

{

matrix[i][j] = matrix[j][i];

printf(" %2d\t", matrix[j][i]); // Выводим элементы матрицы M1

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

}

fout << " \n ";

cout << endl;

}

choosen = new bool[n];

for (int i = 1; i < n; i++)

choosen[i] = false;

choosen[0] = true;

int x, y;

cout << " Результат\n";

while (edgeNum < n - 1) {

x = 0;

y = 0;

int min = INT\_MAX;

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

if (choosen[i]) {

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

{

if (!choosen[j] && matrix[i][j]) {

if (min > matrix[i][j]) {

x = i;

y = j;

min = matrix[i][j];

}

}

}

}

}

cout << '(' << x + 1 << " -> " << y + 1 << ") " << matrix[x][y] << " | ";

choosen[y] = true;

++edgeNum;

ofstream fout;

fout.open(s, ofstream::app);

if (!fout.is\_open()) {

cout << " Ошибка открытия файла!" << endl;

}

else

{

fout << '(' << x + 1 << " -> " << y + 1 << ") " << matrix[x][y] << " | ";

}

fout.close();

}

}

}