МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 2

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Доцент                                                               С.Л. Козенко

должность, уч. Степень, звание           подпись, дата                     инициалы, фамилия

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2

РЕШЕНИЕ СЛАУ

по курсу: ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                               Бобрович Н. С.

                                                                        подпись, дата                      инициалы, фамилия

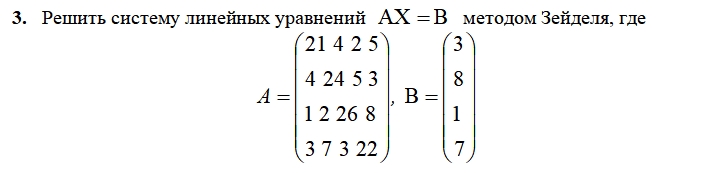
Санкт-Петербург 2023

**Цель работы:**

а) освоение основных методов решения систем линейных алгебраических уравнений (СЛАУ);

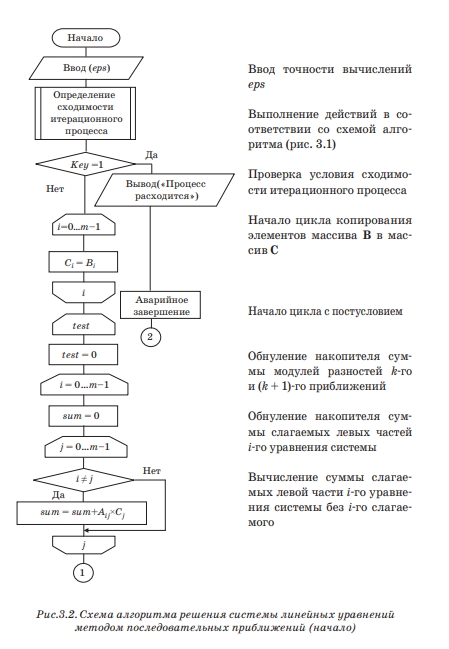
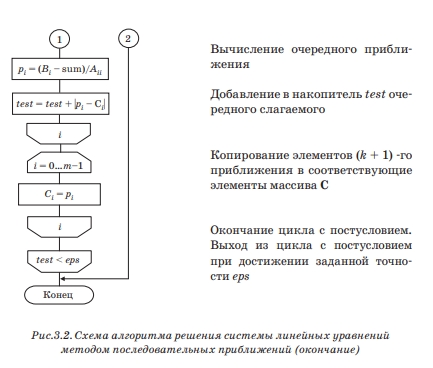
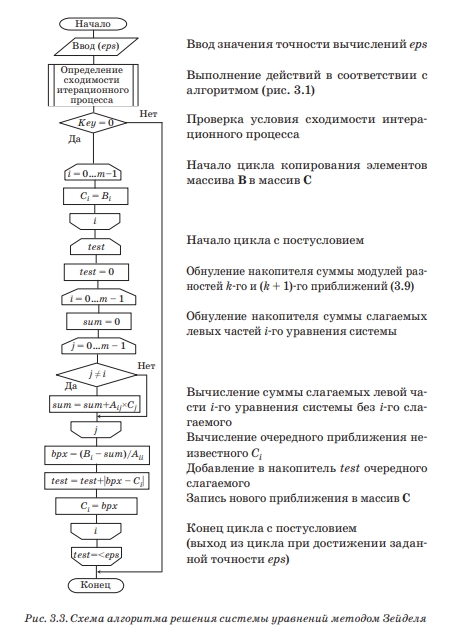
б) совершенствование навыков по алгоритмизации и программированию вычислительных задач.

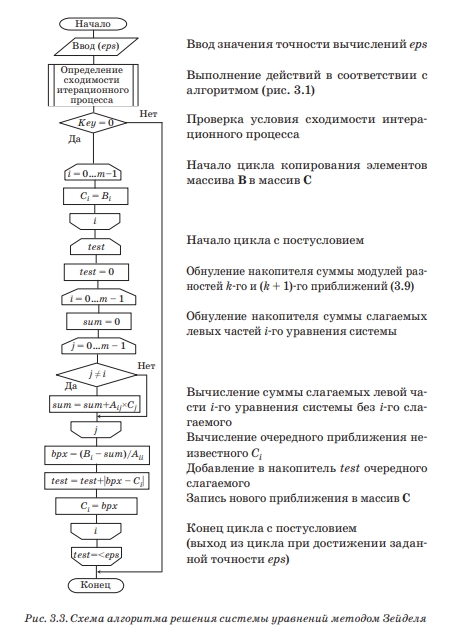
**Задание:**



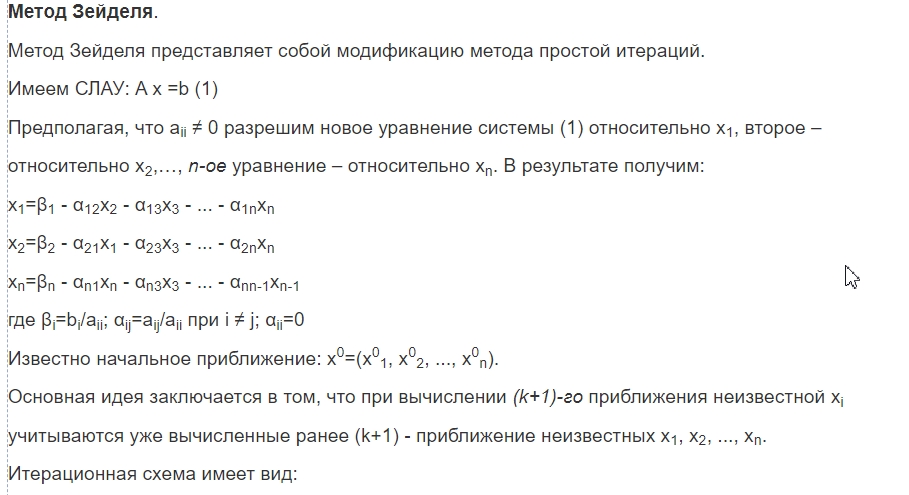
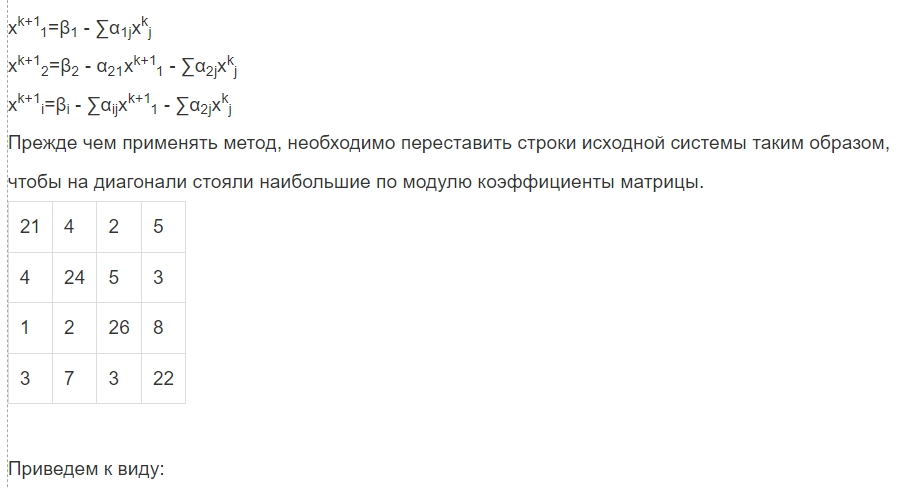
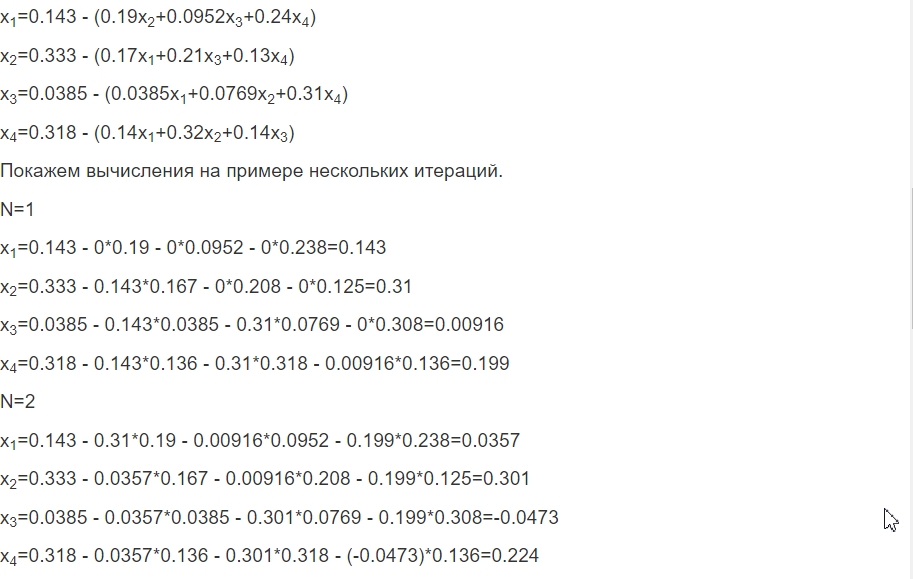
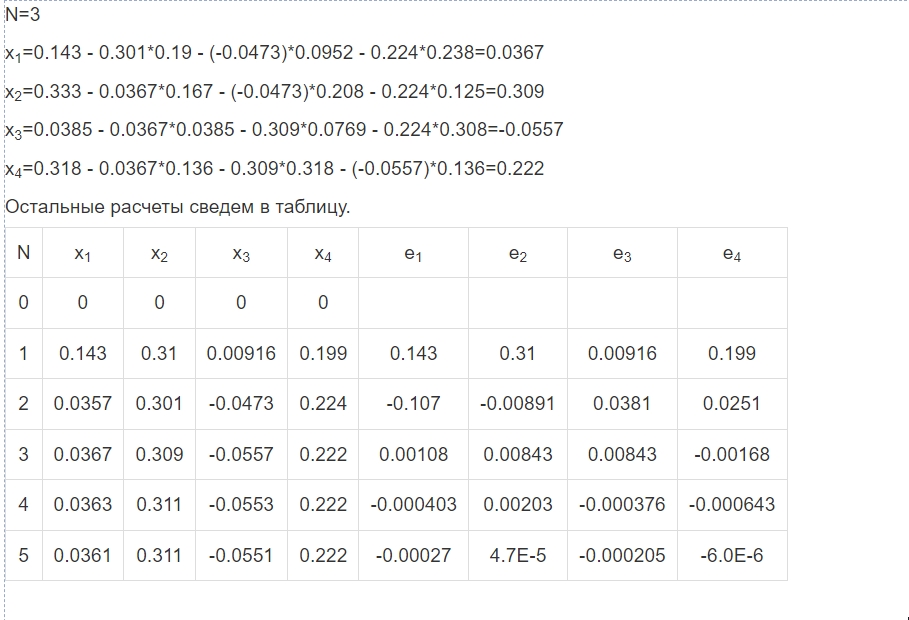
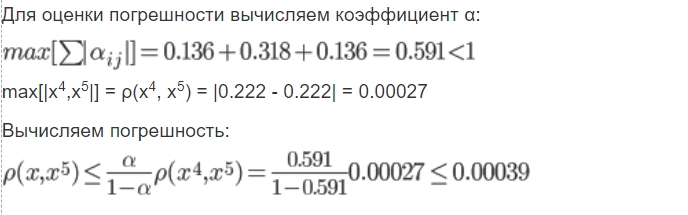
**Математическая часть:**

Метод Зейделя относится к итерационным методам решения систем линейных уравнений, обеспечивающим хорошую сходимость итерационного процесса поиска корней системы. Пусть задана система (3.2). Прежде всего необходимо задать значения начальных (нулевых) приближений корней X1 (0), X2 (0),..., Xm(0). Если отсутствует какая–нибудь информация об этих значениях, то их можно принять равными свободным членам системы уравнений (3.2) или даже принять равными нулю. Выбранные таким образом нулевые приближения корней подставим в первое уравнение системы (3.2) и получим первое приближение корня X1 X1 (1) = β1 + α12 X2 (0) + α13 X3 (0) + … + α1 m Xm(0). Используя во втором уравнении системы (3.2) найденное первое приближение корня X1 и нулевые приближения остальных корней, получим первое приближение корня X2 X2 (1) = β2 + α21 X1 (1) + α23 X2 (0) + … + α2 m Xm(0). Повторяя эту процедуру последовательно для всех уравнений системы (3.2), получим в итоге первое приближение корня Xm(1) = βm + αm 1 X1 (1) + αm2 X2 (1) + … + αm m–1 Xm–1(1). Используя первые приближение корня системы, можно аналогичным образом найти вторые приближения X1 (2) = β1 + α12 X2 (1) + α13 X3 (1) + … + α1 mXm(1) X2 (2) = β1 + α21 X1 (2) + α23 X3 (1) + … + α2 mXm(1) .............................................................................................. Xm(2) = βm + αm 1 X1 (2) + αm 2 X2 (2) + … + αm m–1 Xm–1(2). Затем, используя вторые приближения, можно вычислить третьи и т.д. Итерационный процесс решения системы линейных уравнений методом Зейделя сходится к единственному решению при любом выборе начальных приближений искомых корней, если выполняется одно из условий (3.5). Погрешность решения системы уравнений методом Зейделя принято оценивать по формулам (3.7) — (3.9). 3.4. Схема алгоритма решения системы линейных уравнений методом Зейделя Схема алгоритма решений системы уравнений (1.7) методом Зейделя представлена на рис. 3.3. В схему алгоритма определения сходимости итерационного процесса (рис.3.1), которая используется в методе Зейделя, необходимо внести следующие изменения: заменить блок вычисления значения переменной Flag = Flag + 1 на Flag = 1; условие Flag = m заменить на условие Flag = 1. Эта замена объясняется Замечанием, приведенным на стр. 29, и связано с тем, что для метода Зейделя достаточно, чтобы неравенство (3.6) выполнялось хотя бы для одной строки матрицы A.





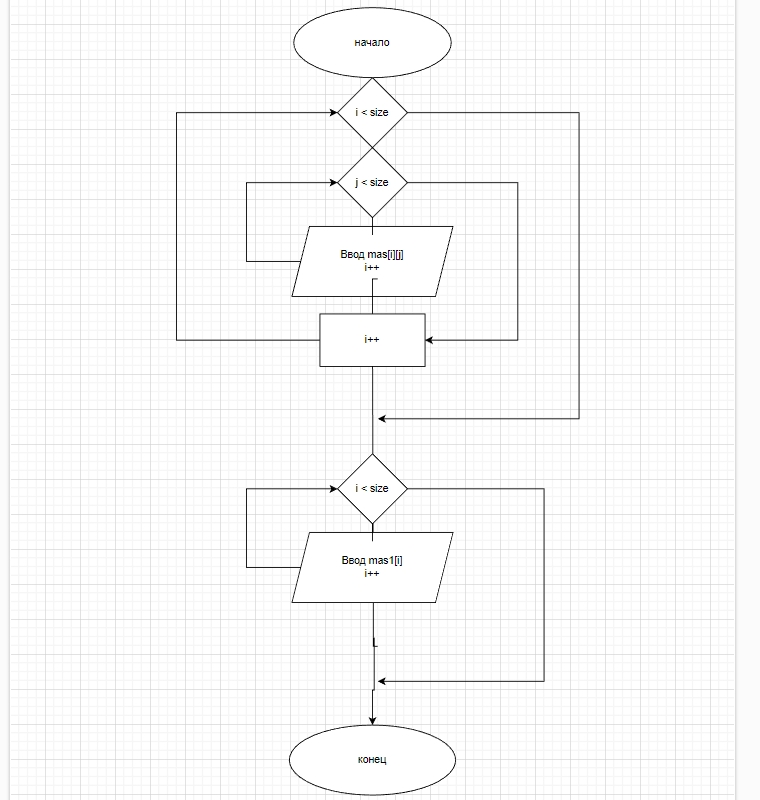
**Аналитические расчеты:**



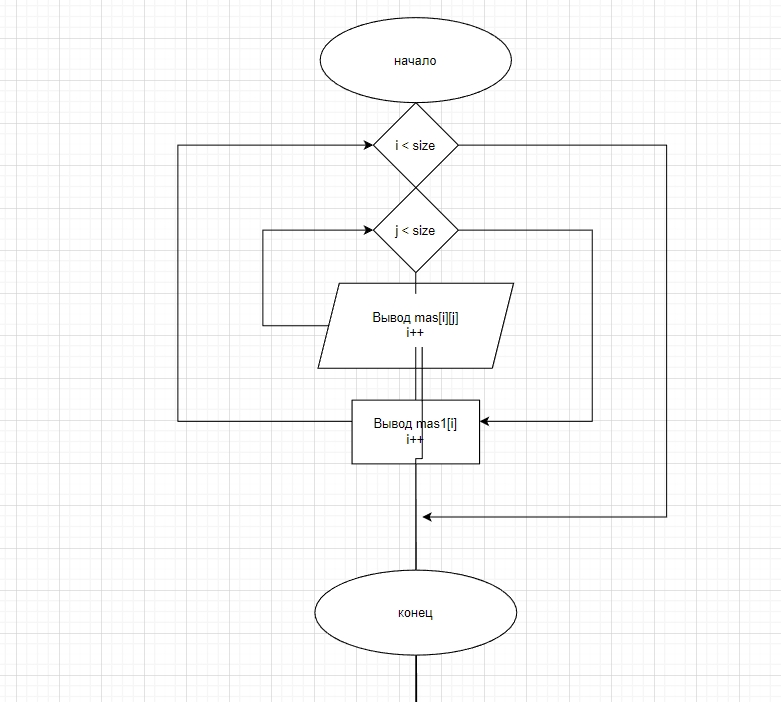
**Схема алгоритма:**

Class Matr:

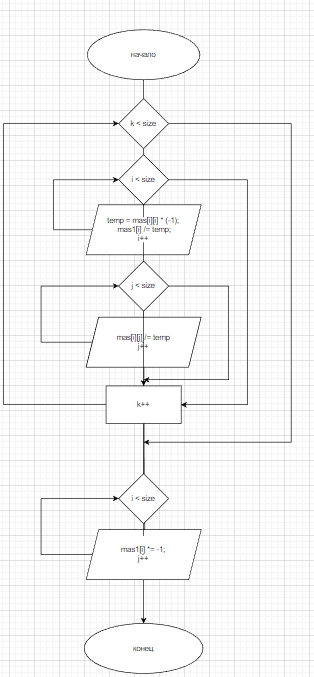
Add:



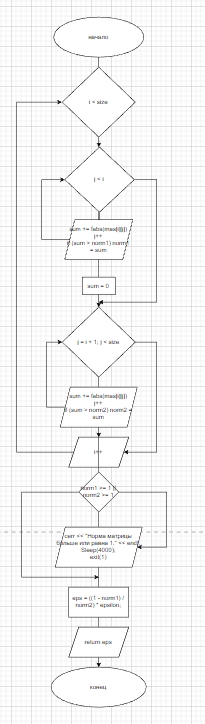
Print:



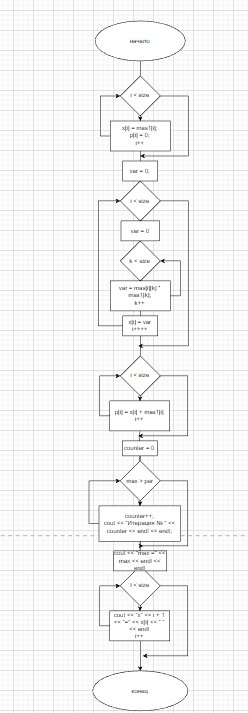
Preob:



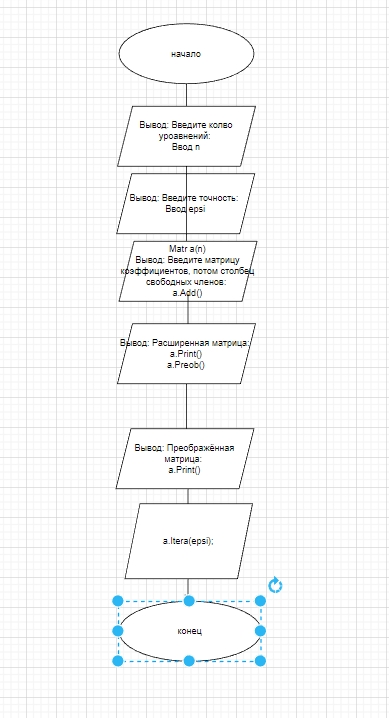
Pogr:



Itera:



main:



**Листинг кода программы:**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <windows.h>

using namespace std;

class Matr

{

private:

int size;

double\*\* mas;

double\* mas1;

public:

Matr()

{

size = 0;

mas = NULL;

mas1 = NULL;

}

Matr(int l)

{

size = l;

mas = new double\* [l];

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

mas[i] = new double[l];

mas1 = new double[l];

}

void Add()

{

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

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

cin >> mas[i][j];

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

{

cin >> mas1[i];

}

}

void Print()

{

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

{

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

{

cout << setw(4) << mas[i][j] << " ";

}

cout << " " << mas1[i] << endl;

}

}

void Preob()

{

double temp = 0;

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

{

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

{

temp = mas[i][i] \* (-1);

mas1[i] /= temp;

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

{

mas[i][j] /= temp;

}

}

}

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

{

mas1[i] \*= -1;

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

mas[i][i] = 0;

}

}

double Pogr(double\*\* mas, double epsilon)

{

double eps = 0; double sum = 0, max = 0;

double norm1 = 0, norm2 = 0;

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

{

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

{

sum += fabs(mas[i][j]);

if (sum > norm1) norm1 = sum;

}

sum = 0;

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

{

sum += fabs(mas[i][j]);

if (sum > norm2) norm2 = sum;

}

sum = 0;

}

if (norm1 >= 1 || norm2 >= 1)

{

cerr << "Норма матрицы больше или равна 1." << endl;

Sleep(4000);

exit(1);

} //проверка

eps = ((1 - norm1) / norm2) \* epsilon;

return eps;

}

void Itera(double epsilon)

{

double\* x = new double[size];

double\* p = new double[size];

double\* a = new double[size];

double\* E = new double[size];

double per = Pogr(mas, epsilon), max = 0;

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

{

x[i] = mas1[i];

p[i] = 0;

}

double var = 0;

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

{

var = 0;

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

var = mas[i][k] \* mas1[k];

x[i] = var;

}

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

p[i] = x[i] + mas1[i];

int counter = 0;

do

{

counter++;

cout << "Итерация № " << counter << endl << endl;

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

{

var = 0;

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

var += (mas[i][j] \* p[j]);

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

var += (mas[i][j] \* x[j]);

a[i] = var;

x[i] = mas1[i] + a[i];

}

max = 0;

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

{

E[i] = fabs(x[i] - p[i]);

if (max < E[i]) max = E[i];

p[i] = x[i];

cout << "x" << i + 1 << "=" << x[i] << " " << endl;

}

cout << endl;

cout << "max =" << max << endl << endl;

} while (max > per);

cout << endl << "Результат: \n\n";

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

cout << "x" << i + 1 << "=" << x[i] << " " << endl;

delete[] x;

delete[] p;

delete[] E;

delete[] a;

}

~Matr()

{

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

delete mas[i];

delete mas;

}

};

int main()

{

setlocale(LC\_ALL, "rus");

int n; double epsi;

cout << "Введите количество уравнений: ";

cin >> n;

cout << "Введите желаемую точность: ";

cin >> epsi;

Matr a(n);

cout << "Введите матрицу коэффициентов, потом столбец свободных членов:" << endl;

a.Add();

cout << endl << "Расширенная матрица:" << endl;

a.Print();

a.Preob();

cout << endl << "Преображенная матрица" << endl;

a.Print();

cout << endl;

a.Itera(epsi);

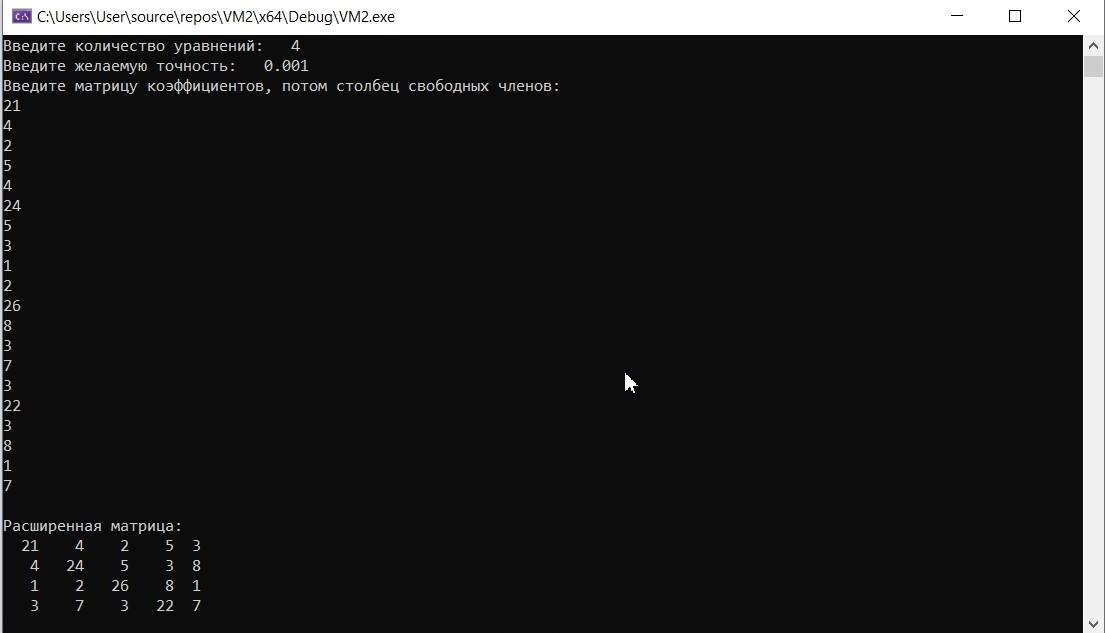
cout << endl;

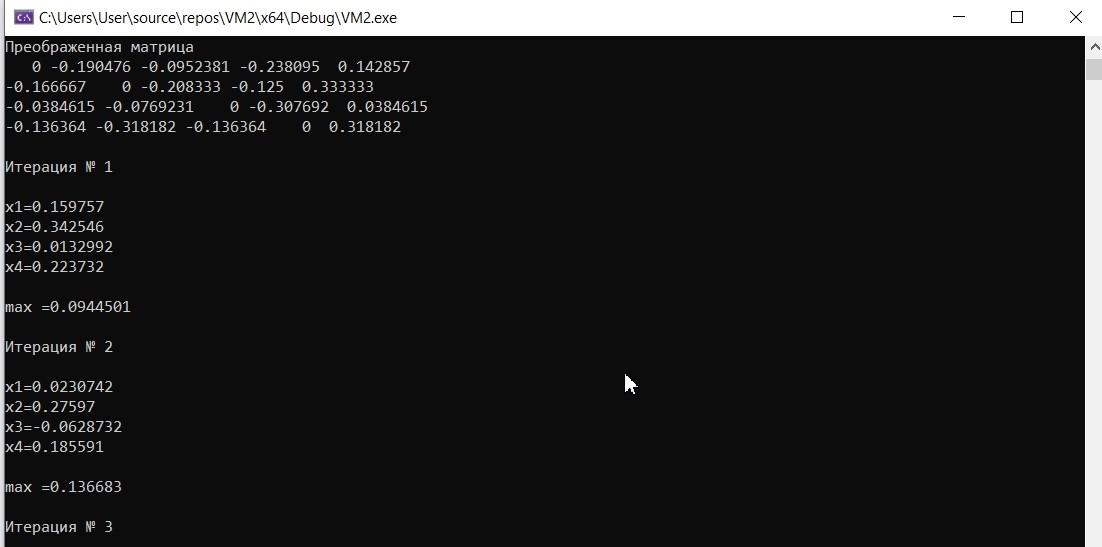
system("pause");

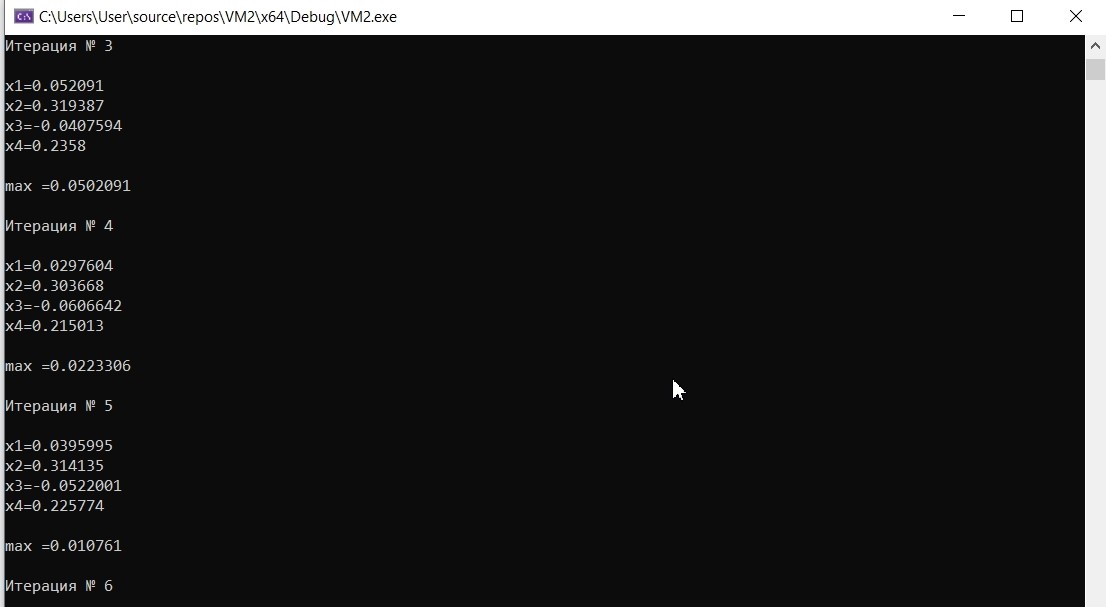
return 0;

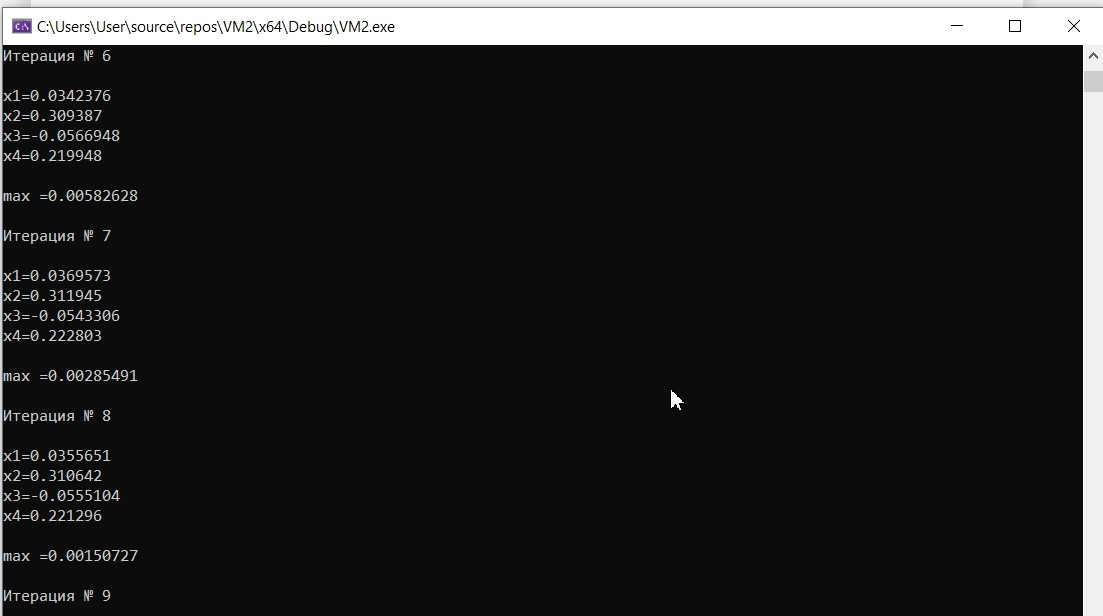
}

**Результаты программных расчетов:**

****

****

****

****

****

**Сравнение результатов программных и аналитических расчетов:**

Исходя из результатов мы видим, что результаты сходятся с допустимой разницей.

**Вывод**

В ходе выполнения практической работы №2 был освоен метод решения

СЛАУ – метод Зейделя. Также были улучшены навыки по

алгоритмизации и программированию вычислительной задачи на языке C++ в программе Microsoft Visual Studio.