МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №2

з курсу «Чисельні методи»

тема: «МЕТОДИ РОЗВ'ЯЗАННЯ СЛАР»

Виконав: студент 2 курсу

групи КА-23

Деундяк О.В.

Прийняла: Кузнєцова Н. В.

Київ – 2014р.

Варіант 9

Умова: Знайти розв'язок рівняння Ax=b.

Допрограмовий етап

Оскільки система не є симетричною, я обрав для реалізації метод LU – розкладу.

Текст програми:

Matrix.h

```
#pragma once

#include <vector>
#include <string>
#include <fstream>

using std::vector;
using std::string;
using std::ifstream;

typedef double Item;

class Matrix
{
	vector <vector <Item>> A;
	size_t n;

	vector<Item> SolveL(const vector<Item> &b) const;
	vector<Item> SolveU(const vector<Item> &b) const;
public:
	Matrix(size_t dimension, Item fill);
	Matrix(size_t dimension);
	Matrix(size_t dimension, ifstream &file);

	Item Determinant() const;

	Matrix Inverse();

	Matrix operator*(const Matrix &other) const;
	vector<Item> operator*(const vector<Item> &vec) const;

	vector<Item> Solve(const vector<Item> &b) const;
```

```
	void Show() const;
	void LU(Matrix &L, Matrix &U) const;
};
```

Matrix.cpp

```
#include "Matrix.h"
#include <iostream>
#include <iomanip>
#include <fstream>
#include <exception>

const double eps = 1E-5;


using std::cin;
//using std::cout;
using std::endl;
using std::ifstream;

extern std::ofstream cout;

Matrix::Matrix(size_t n_, Item fill) :
	n(n_)
{
	for (size_t i = 1; i <= n; i++)
	{
		vector<Item> T;
		for (size_t j = 1; j <= n; j++)
			T.push_back(fill);
		A.push_back(T);
	}
}


Matrix::Matrix(size_t n_, ifstream &file) :
	n(n_)
{
	for (size_t i = 1; i <= n; i++)
	{
		vector<Item> T;
		Item t;
		for (size_t j = 1; j <= n; j++)
		{
			file >> t;
			T.push_back(t);
		}
		A.push_back(T);
	}
}

Matrix::Matrix(size_t n_) :
	n(n_)
{
	for (size_t i = 1; i <= n; i++)
	{
		vector<Item> T;
		Item t;
		cout << "Enter raw #" << i << ": ";
		for (size_t j = 1; j <= n; j++)
		{
			cin >> t;
			T.push_back(t);
		}
		A.push_back(T);
```

```
	}
	cout << endl;
}

void Matrix::LU(Matrix &L, Matrix &U) const
{
	U = *this;

	for (size_t i = 0; i < n; i++)
	for (size_t j = i; j < n; j++)
	{
		if (abs(U.A[i][i]) < eps)
			throw std::logic_error("LU decompsition doesn't exist");
		L.A[j][i] = U.A[j][i] / U.A[i][i];
	}

	for (size_t k = 1; k < n; k++)
	{
		for (size_t i = k - 1; i < n; i++)
		for (size_t j = i; j < n; j++)
			L.A[j][i] = U.A[j][i] / U.A[i][i];

		for (size_t i = k; i < n; i++)
		for (size_t j = k - 1; j < n; j++)
			U.A[i][j] = U.A[i][j] - L.A[i][k - 1] * U.A[k - 1][j];
	}
}

void Matrix::Show() const
{
	std::streamsize pr = cout.precision(5);
	cout << std::left;
	cout << "MATRIX " << n << 'x' << n << ": " << endl;
	for (size_t i = 0; i < n; i++)
	{
		for (size_t j = 0; j < n; j++)
		{
			cout << std::setw(10);
			cout << A[i][j];
			cout << ' ';
		}
		cout << endl;
	}
	cout << endl;
	cout.precision(pr);
}

Matrix Matrix::operator*(const Matrix &other) const
{
	Matrix P(n, 0);

	for (size_t row = 0; row < n; row++)
		for (size_t col = 0; col < n; col++)
			for (size_t inner = 0; inner < n; inner++)
				P.A[row][col] += A[row][inner] * other.A[inner][col];

	return P;
}

vector<Item> Matrix::SolveL(const vector<Item> &b) const
{
	vector<Item> r;
	r.reserve(n);
	Item t;
	for (size_t i = 0; i < n; i++)
```

```
	{
		t = 0;
		for (size_t j = 0; j < i; j++)
			t += A[i][j] * r[j];
		r.push_back((b[i] - t) / A[i][i]);
	}

	return r;
}

vector<Item> Matrix::SolveU(const vector<Item> &b) const
{
	vector<Item> r;
	r.resize(n);
	Item t;
	for (size_t i = n; i > 0; i--)
	{
		t = 0;
		for (size_t j = n-1; j > i - 1; j--)
			t += A[i - 1][j] * r[j];
		r[i - 1] = (b[i - 1] - t) / A[i - 1][i - 1];
	}

	return r;
}

vector<Item> Matrix::Solve(const vector<Item> &b) const
{
	Matrix L(n, 0), U(n, 0);
	LU(L, U);

	//(L*U).Show();

	return U.SolveU(L.SolveL(b));
}

Item Matrix::Determinant() const
{
	Matrix L(n, 0), U(n, 0);

	Item det = 1;

	LU(L, U);
	for (size_t i = 0; i < n; i++)
		det *= U.A[i][i];

	return det;
}

Matrix Matrix::Inverse()
{
	Matrix m_inv(n, 0);

	vector<Item> b, r;
	b.resize(n);

	for (size_t i = 0; i < n; i++)
	{
		for (size_t j = 0; j < n; j++)
			b[j] = (i == j) ? 1 : 0;
		r = Solve(b);

		for (size_t j = 0; j < n; j++)
			(m_inv.A)[j][i] = r[j];
	}
```

```
	return m_inv;
}

vector<Item> Matrix::operator*(const vector<Item> &vec) const
{
	vector<Item> res;
	for (size_t i = 0; i < n; i++)
	{
		Item r = 0;
		for (size_t j = 0; j < n; j++)
			r += A[i][j] * vec[j];
		res.push_back(r);
	}

	return res;
}
```

SOLE.h

```
#pragma once
#include "Matrix.h"
#include <memory>

using std::shared_ptr;

class SOLE
{
	shared_ptr<Matrix> A;
	vector<Item> b;
public:
	void start();
};
```

SOLE.cpp

```
#include "SOLE.h"
#include <iostream>
#include <string>
#include <fstream>

const double eps = 1E-5;

using std::cin;
//using std::cout;
using std::endl;
using std::ifstream;

std::ofstream cout("output.txt");

const string INFILE = "input.txt";

void SOLE::start()
{
	ifstream file(INFILE);
	size_t n;
	if (!file.is_open())
		throw std::runtime_error("Can't open file");

	file >> n;
	A = std::make_shared<Matrix>(n, file);
	cout << "Input Matrix" << endl;
	A->Show();
```

```
	Item det = A->Determinant();
	if (abs(det) < eps)
		throw std::exception("Singular matrix");

	cout << "Determinant=" << det << endl;

	cout << "A^(-1): " << endl;
	A->Inverse().Show();

	cout << "A*A^(-1):" << endl;
	(A->Inverse()*(*A)).Show();

	for (size_t i = 0; i < n; i++)
	{
		Item t;
		file >> t;
		b.push_back(t);
	}

	vector<Item> r = A->Solve(b);
	cout << "Result={";
	for (size_t i = 0; i < n; i++)
		cout << r[i] << ' ';
	cout << '}' << endl;

	cout << "Ax-b={";
	r = *A * r;
	for (size_t i = 0; i < n; i++)
		cout << r[i] - b[i] << ' ';
	cout << '}' << endl;

}
```

main.cpp

```
#include "SOLE.h"
#include <iostream>


using std::cin;
using std::cout;

int main()
{
	SOLE sole;
	try
	{
		sole.start();
	}
	catch (std::exception e)
	{
		cout << e.what();
	}

	cout << "DONE";
	cin.get();
	cin.get();
}
```

Результати роботи програми

Input Matrix
MATRIX 5x5:

7.25 1.16 0.91 1.105 -1.11
1.04 3.17 1.3 -1.63 0.12
1.03 -2.46 6.43 2.1 0.583
1.375 0.16 2.1 5.11 -6
1.59 -0.78 -0.317 3 6

Determinant=6897.65
A^(-1):
MATRIX 5x5:
0.15732 -0.05658 0.0030372 -0.044668 -0.014728
-0.067015 0.29599 -0.076535 0.092465 0.081584
-0.030385 0.11164 0.13647 -0.00094858 -0.022063
-0.055958 0.018769 -0.036965 0.14691 0.13978
-0.024027 0.049987 0.014938 -0.049649 0.11012

A*A^(-1):
MATRIX 5x5:
1 2.7756e-017 8.6736e-018 -4.1633e-017 0
2.7756e-017 1 4.8572e-017 1.6653e-016 -5.5511e-017
0 -2.0817e-017 1 -1.3878e-017 -2.7756e-017
2.7756e-017 -1.3878e-017 4.1633e-017 1 1.1102e-016
0 0 6.9389e-018 5.5511e-017 1

Result={-0.18417 0.902774 0.275956 1.13733 -0.79292 }
Ax-b={-4.44089e-016 -2.22045e-016 -2.22045e-016 0 4.44089e-016 }

Висновки:

Метод LU – добре працює для розв’язку СЛАР і зручний для знаходження оберненої матреці, визначника матриці. Також цей метод дає дуже малу похибку (порядку 10^{-16}).