Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

Звіт
Про виконання лабораторної роботи №3

На тему:

«Розв'язування систем лінійних алгебраїчних рівнянь методом Крамера та методом оберненої матриці»
з дисципліни «Чисельні методи»

Лектор:
доцент каф. ПЗ
Мельник Н. Б.
Виконав:
ст. гр. ПЗ-11
Морозов О. Р.
Прийняла:
доцент каф. ПЗ
Мельник Н. Б.

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

Тема: Розв'язування систем лінійних алгебраїчних рівнянь методом Крамера та методом оберненої матриці
Мета: ознайомлення на практиці з методом Крамера та методом оберненої матриці розв’язування систем лінійних алгебраїчних рівнянь.

Теоретичні відомості

Метод Крамера - розглянемо СЛАР, яка містить n рівнянь та n невідомих, причому визначник її не дорівнює нулеві. Для знаходження невідомих і x застосовують формули Крамера: $x_i = \frac{det\ A_i}{det\ A}$, $i = \overline{1, n}$, де detA- визначник матриці A, detAi - визначник матриці Ai, яку отримують з матриці A шляхом заміни її i -го стовпця стовпцем вільних членів. Щоб розв’язати СЛАР з n невідомими потрібно обчислити (n+1) визначників n -го порядку, що призводить до виконання nn! операцій. Через громіздкість обчислень визначників метод Крамера не застосовують на практиці для великої розмірності матриці коефіцієнтів СЛАР.

Метод оберненої матриці(матричний метод) - у лінійній алгебрі часто використовують матричний метод розв’язування систем лінійних алгебраїчних рівнянь. Цей метод ґрунтується на обчисленні оберненої матриці −1 A, яка існує лише при умові, коли визначник матриці A відмінний від нуля det A≠ 0. Якщо обидві частини матричного рівняння зліва помножити на матрицю −1 A, то отримаємо співвідношення $A^{-1}AX = A^{-1}B$. Враховуючи, що добуток оберненої матриці на саму матрицю дає одиничну матрицю, а результатом добутку одиничної матриці E на матрицю-стовпець X є матриця-стовпець X, тобто EX = X , одержимо матричний розв'язок системи лінійних алгебраїчних рівнянь у вигляді $x = A^{-1}B$

Індивідуальне завдання

Варіант 2

Написати програму розв’язку матриці
0.62*a + 0.56*b - 0.43*c = 1.16
1.32*a - 0.88*b + 1.76*c = 2.07
0.73*a + 1.42*b - 0.34*c = 2.18

Хід роботи

Код програми

```
#include <iostream>
#include <cmath>
#define size 3

char checkTask(char choise);

void printResult(double matrix[size][size], double solveMatrix[size],
double X[size]);

double determinant(double matrix[size][size]);
double det(double minorMatrix[size-1][size-1]);
double minor(double matrix[size][size], int row, int column);

void matrixMethod(double matrix[size][size], double solveMatrix[size]);

void kramerMethod(double matrix[size][size], double solveMatrix[size]);

int main() {

	double solveMatrix[size] = {1.16, 2.07, 2.18}, matrix[size][size] =
{{0.62, 0.56, -0.43},{1.32, -0.88, 1.76},{0.73, 1.42, -0.34}};//standart
matrix
	int sizeMatrix = size;//size of matrix

	char choise = ' ';

	std::cout << "Custom 3x3 matrix - U\nStandart 3x3 matrix - S\nClose
- E" << std::endl;//menu
```

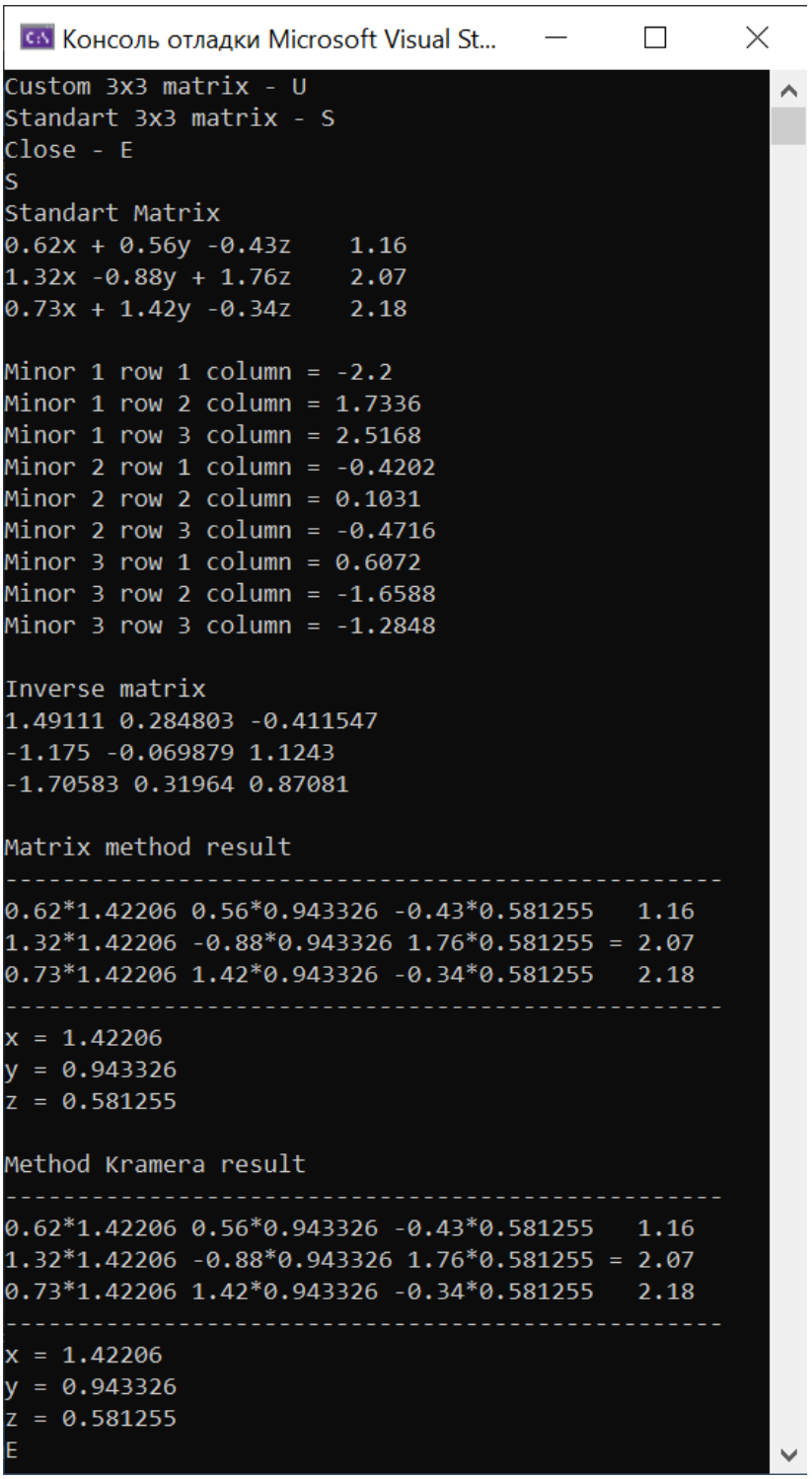
```
	while ((choise = checkTask(choise)) != 'E') {
		switch (choise) {
		case 'U':{			//custom matrix

			for (int i = 0; i < size; i++) {
				for (int j = 0; j <= size; j++) {
					std::cout << "Enter " << i + 1 << " row " << 
j+1 << " column matrix element" << std::endl;
					if (j != size) {
						std::cin >> matrix[i][j];
					}
					else {
						std::cin >> solveMatrix[i];
					}
				}
			}

			std::cout << "Your Matrix\n" << matrix[0][0] << "x " << 
matrix[0][1] << "y " << matrix[0][2] << "z\t = " << solveMatrix[0] << "\n"
				<< matrix[1][0] << "x " << matrix[1][1] << "y " << 
matrix[1][2] << "z\t = " << solveMatrix[1] << "\n"
				<< matrix[2][0] << "x " << matrix[2][1] << "y " << 
matrix[2][2] << "z\t = " << solveMatrix[2] << std::endl;

			if (solveMatrix[0] == 0 && solveMatrix[1] == 0 && 
solveMatrix[2] == 0) {
				std::cout << "\nMatrix homogeneous" << std::endl;
				double x[size] = { 0, 0, 0 };
				printResult(matrix, solveMatrix, x);
				break;
			}
			else {
				matrixMethod(matrix, solveMatrix);
				kramerMethod(matrix, solveMatrix);
			}
			break;
		}
		case 'S': { //standart matrix
			std::cout << "Standart Matrix\n" << matrix[0][0] << "x + 
" << matrix[0][1] << "y " << matrix[0][2] << "z\t" << solveMatrix[0] << 
"\n"
				<< matrix[1][0] << "x " << matrix[1][1] << "y + " 
<< matrix[1][2] << "z\t" << solveMatrix[1] << "\n"
				<< matrix[2][0] << "x + " << matrix[2][1] << "y " 
<< matrix[2][2] << "z\t" << solveMatrix[2] << std::endl;

			matrixMethod(matrix, solveMatrix);
```

```
                kramerMethod(matrix, solveMatrix);
                break;
            }
            default:{   //try again
                if (choise != '\n') {
                    std::cout << "Not correct sumbol" << std::endl;
                }
                break;
            }
            }
      }

      return 0;
}

char checkTask(char choise) {
      std::cin >> choise;
      return choise;
}

void printResult(double matrix[size][size], double solveMatrix[size],
double X[size]) {
      std::cout << "-------------------------------------------------" <<
std::endl;
      std::cout << matrix[0][0] << "*" << X[0] << " " << matrix[0][1] <<
"*" << X[1] << " " << matrix[0][2] << "*" << X[2] << "   " <<
solveMatrix[0] << "\n"
            << matrix[1][0] << "*" << X[0] << " " << matrix[1][1] << "*"
<< X[1] << " " << matrix[1][2] << "*" << X[2] << " = " << solveMatrix[1]
<< "\n"
            << matrix[2][0] << "*" << X[0] << " " << matrix[2][1] << "*"
<< X[1] << " " << matrix[2][2] << "*" << X[2] << "   " << solveMatrix[2]
<< std::endl;
      std::cout << "-------------------------------------------------" <<
std::endl;
      std::cout << "x = " << X[0] << "\ny = " << X[1] << "\nz = " << X[2]
<< std::endl;
}

double determinant(double matrix[size][size]) {
      return (matrix[0][0] * matrix[1][1] * matrix[2][2] + matrix[0][1] *
matrix[2][0] * matrix[1][2] + matrix[1][0] * matrix[2][1] * matrix[0][2] -
(matrix[0][2] * matrix[1][1] * matrix[2][0] + matrix[0][1] * matrix[1][0]
* matrix[2][2] + matrix[0][0] * matrix[1][2] * matrix[2][1]));
}

double det(double minorMatrix[size-1][size-1]) {
```

```
	return (minorMatrix[0][0] * minorMatrix[1][1] - minorMatrix[1][0] *
minorMatrix[0][1]);
}

double minor(double matrix[size][size], int row, int column) {
	int subi = 0, subj = 0;
	double subMatrix[size-1][size-1];
	for (int i = 0; i < size; i++)
	{
		if (i != row)
		{
			for (int j = 0; j < size; j++)
			{
				if (j != column)
				{
					subMatrix[subi][subj] = matrix[i][j];
					subj++;
				}
			}
			subi++;
			subj = 0;
		}
	}
	return ((row + column) % 2 == 0) ? det(subMatrix) : -det(subMatrix);
}

void matrixMethod(double matrix[size][size], double solveMatrix[size]) {
	if (determinant(matrix) == 0) {
		std::cout << "Determinant = 0\nMatrix was singular" <<
std::endl;
		return;
	}
	double a[size][size], aTemp[size][size], x[size];

	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			a[i][j] = minor(matrix, i, j);
		}
	}

	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			aTemp[i][j] = a[j][i];
		}
	}

	double detA = determinant(matrix);
```

```
	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			a[i][j] = (aTemp[i][j]/detA);
		}
	}

	std::cout << "\nInverse matrix\n" << a[0][0] << " " << a[0][1] << "
" << a[0][2] << "\n"
		<< a[1][0] << " " << a[1][1] << " " << a[1][2] << "\n"
		<< a[2][0] << " " << a[2][1] << " " << a[2][2] << std::endl;

	for (int i = 0; i < size; i++) {
		x[i] = a[i][0] * solveMatrix[0] + a[i][1] * solveMatrix[1] +
a[i][2] * solveMatrix[2];
	}
	std::cout << "\nMatrix method result" << std::endl;
	printResult(matrix, solveMatrix, x);
}

void kramerMethod(double matrix[size][size], double solveMatrix[size]) {
	if (determinant(matrix) == 0) {
		std::cout << "Determinant = 0\nMatrix was singular" <<
std::endl;
		return;
	}

	double a[size][size], x[size];
	double detA = determinant(matrix);

	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			for (int k = 0; k < size; k++) {
				if (k == i) {
					a[j][k] = solveMatrix[j];
				}
				else {
					a[j][k] = matrix[j][k];
				}
			}
		}
		x[i] = determinant(a) / detA;
	}

	std::cout << "\nMethod Kramera result" << std::endl;
	printResult(matrix, solveMatrix, x);
}
```

Результат роботи



Висновок

Виконуючи лабораторну роботу №3, я навчився програмувати розв'язок матриці методом Крамера та методом оберненої матриці.