**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний Технічний Університет України**

**«Київський Політехнічний Інститут»**

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

**Лабораторна робота №5**

*з дисципліни «Алгоритми та методи обчислень»*

*на тему: «Розв’язання систем лінійних алгебраїчних рівнянь»*

**Виконав:**

студент 2-го курсу ФІОТ

групи ІО-44

*Барабаш Т.А.*

**Варіант:** 3

**Перевірив:**

Старший викладач

*Порєв В. М.*

**Київ – 2016**

**ЛАБОРАТОРНА РОБОТА №5**

*Розв’язання систем лінійних алгебраїчних рівнянь.*

**Мета:** Вивчити алгоритми методів розв'язання систем лінійних алгебраїчних рівнянь на ЕОМ.

1. **Завдання**

Відповідно до варіанту завдання розробити блок-схеми обчислення виразів для лінійного алгоритму, алгоритму, що розгалужується та циклічного алгоритму. У відповідності до блок-схеми створити програму обчислення виразy.

***Варіант: 3***

***Метод:*** Жордана-Гауса

***Матриця коефіцієнтів системи:***

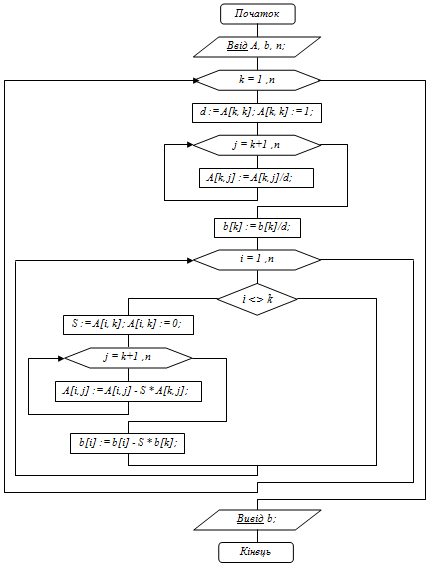
|  |  |  |
| --- | --- | --- |
| 1 | -3 | 2 |
| 3 | -4 | 0 |
| 2 | -5 | 3 |

***Стовпець вільних членів:***

|  |
| --- |
| 5 |
| 7 |
| 9 |

***Розв'язки СЛАР:*** х1 = 5, х2 = 2, х3 = 3

**ІІ. Блок-схема алгоритму**

****

**ІІІ. Код програми**

**MainActivity**

public void loadFromTask(View view) {  
 Context mineContext = MainActivity.this;  
 *a1* = (EditText) findViewById(R.id.*a1*);  
 *a2* = (EditText) findViewById(R.id.*a2*);  
 *a3* = (EditText) findViewById(R.id.*a3*);  
 *b1* = (EditText) findViewById(R.id.*b1*);  
 *b2* = (EditText) findViewById(R.id.*b2*);  
 *b3* = (EditText) findViewById(R.id.*b3*);  
 *c1* = (EditText) findViewById(R.id.*c1*);  
 *c2* = (EditText) findViewById(R.id.*c2*);  
 *c3* = (EditText) findViewById(R.id.*c3*);  
 *d1* = (EditText) findViewById(R.id.*d1*);  
 *d2* = (EditText) findViewById(R.id.*d2*);  
 *d3* = (EditText) findViewById(R.id.*d3*);  
 *x1* = (EditText) findViewById(R.id.*x1*);  
 *x2* = (EditText) findViewById(R.id.*x2*);  
 *x3* = (EditText) findViewById(R.id.*x3*);  
 *fam* = (FloatingActionMenu) findViewById(R.id.*fam*);  
 pasteFromTask.*load*(view, mineContext);  
}  
  
public void clearTask(View view) {  
 *a1* = (EditText) findViewById(R.id.*a1*);  
 *a2* = (EditText) findViewById(R.id.*a2*);  
 *a3* = (EditText) findViewById(R.id.*a3*);  
 *b1* = (EditText) findViewById(R.id.*b1*);  
 *b2* = (EditText) findViewById(R.id.*b2*);  
 *b3* = (EditText) findViewById(R.id.*b3*);  
 *c1* = (EditText) findViewById(R.id.*c1*);  
 *c2* = (EditText) findViewById(R.id.*c2*);  
 *c3* = (EditText) findViewById(R.id.*c3*);  
 *d1* = (EditText) findViewById(R.id.*d1*);  
 *d2* = (EditText) findViewById(R.id.*d2*);  
 *d3* = (EditText) findViewById(R.id.*d3*);  
 *x1* = (EditText) findViewById(R.id.*x1*);  
 *x2* = (EditText) findViewById(R.id.*x2*);  
 *x3* = (EditText) findViewById(R.id.*x3*);  
 *fam* = (FloatingActionMenu) findViewById(R.id.*fam*);  
 *fam*.close(true);  
 pasteFromTask.*clear*(view);  
}  
  
public void calculate(View view) {  
 *a1* = (EditText) findViewById(R.id.*a1*);  
 *a2* = (EditText) findViewById(R.id.*a2*);  
 *a3* = (EditText) findViewById(R.id.*a3*);  
 *b1* = (EditText) findViewById(R.id.*b1*);  
 *b2* = (EditText) findViewById(R.id.*b2*);  
 *b3* = (EditText) findViewById(R.id.*b3*);  
 *c1* = (EditText) findViewById(R.id.*c1*);  
 *c2* = (EditText) findViewById(R.id.*c2*);  
 *c3* = (EditText) findViewById(R.id.*c3*);  
 *d1* = (EditText) findViewById(R.id.*d1*);  
 *d2* = (EditText) findViewById(R.id.*d2*);  
 *d3* = (EditText) findViewById(R.id.*d3*);  
 *x1* = (EditText) findViewById(R.id.*x1*);  
 *x2* = (EditText) findViewById(R.id.*x2*);  
 *x3* = (EditText) findViewById(R.id.*x3*);  
 *fam* = (FloatingActionMenu) findViewById(R.id.*fam*);  
 *fam*.close(true);  
 Calculations.*calculate*(view);  
}  
  
public void restore(View view) {  
 *a1* = (EditText) findViewById(R.id.*a1*);  
 *a2* = (EditText) findViewById(R.id.*a2*);  
 *a3* = (EditText) findViewById(R.id.*a3*);  
 *b1* = (EditText) findViewById(R.id.*b1*);  
 *b2* = (EditText) findViewById(R.id.*b2*);  
 *b3* = (EditText) findViewById(R.id.*b3*);  
 *c1* = (EditText) findViewById(R.id.*c1*);  
 *c2* = (EditText) findViewById(R.id.*c2*);  
 *c3* = (EditText) findViewById(R.id.*c3*);  
 *d1* = (EditText) findViewById(R.id.*d1*);  
 *d2* = (EditText) findViewById(R.id.*d2*);  
 *d3* = (EditText) findViewById(R.id.*d3*);  
 *x1* = (EditText) findViewById(R.id.*x1*);  
 *x2* = (EditText) findViewById(R.id.*x2*);  
 *x3* = (EditText) findViewById(R.id.*x3*);  
 pasteFromTask.*restore*(view);  
}

**Calculations**

public class Calculations {

public static double matrixCoef[][] = new double[pasteFromTask.editTextsCoefficients.length][pasteFromTask.editTextsCoefficients[0].length];

public static double matrixAns[] = new double[pasteFromTask.editTextAnswer.length]; public static double matrixX[] = new double[pasteFromTask.editTextAnswer.length];

public static boolean flagManyAnsw = false;

public static boolean flagNoAnsw = false;

public static void calculate(View view) {

if (!ErrorsCheck.clearnessCheck(pasteFromTask.editTextAnswer) && !ErrorsCheck.clearnessCheck(pasteFromTask.editTextsCoefficients)) {

getDoubleFromEditText(pasteFromTask.editTextsCoefficients, matrixCoef);

getDoubleAnswer(pasteFromTask.editTextAnswer, matrixAns);

for (int i = 0; i < matrixCoef.length; i++) {

boolean flag = false;

for (int j = i; j < matrixCoef[i].length; j++) {

if (matrixCoef[i][j] != 0) {

flag = true;

}

}

if (!flag) {

if (matrixAns[i] == 0) {

Snackbar.make(view, R.string.manySolvings, Snackbar.LENGTH\_LONG).show();

flagManyAnsw = true;

} else {

Snackbar.make(view, R.string.noSorving, Snackbar.LENGTH\_LONG).show();

flagNoAnsw = true;

}

break;

}

double temp = matrixCoef[i][i];

int tempN = i;

for (int j = i + 1; j < matrixCoef[i].length; j++) {

if (Math.abs(matrixCoef[j][i]) > Math.abs(temp)) {

temp = matrixCoef[j][i];

tempN = j;

}

}

if (tempN != i) {

for (int j = i; j < matrixCoef[i].length; j++) {

double buf = matrixCoef[i][j];

matrixCoef[i][j] = matrixCoef[tempN][j];

matrixCoef[tempN][j] = buf;

}

double buf = matrixAns[i];

matrixAns[i] = matrixAns[tempN];

matrixAns[tempN] = buf;

}

for (int j = i + 1; j < matrixCoef[i].length; j++) {

matrixCoef[i][j] = matrixCoef[i][j] / matrixCoef[i][i];

}

matrixAns[i] = matrixAns[i] / matrixCoef[i][i];

matrixCoef[i][i] = 1;

for (int j = 0; j < matrixCoef[i].length; j++) {

if (j != i) {

for (int k = i + 1; k < matrixCoef.length; k++) {

matrixCoef[j][k] = matrixCoef[j][k] - matrixCoef[i][k] \* matrixCoef[j][i];

}

matrixAns[j] = matrixAns[j] - matrixAns[i] \* matrixCoef[j][i];

}

}

}

System.arraycopy(matrixAns, 0, matrixX, 0, matrixX.length);

if (!flagManyAnsw && !flagNoAnsw) {

MainActivity.x1.setText(String.valueOf(matrixX[0]));

MainActivity.x2.setText(String.valueOf(matrixX[1]));

MainActivity.x3.setText(String.valueOf(matrixX[2]));

}

} else {

Snackbar.make(view, R.string.datamissing, Snackbar.LENGTH\_LONG).show();

MainActivity.x1.setText("");

MainActivity.x2.setText("");

MainActivity.x3.setText("");

}

}

public static void getDoubleFromEditText(EditText[][] editTexts, double[][] array) {

for (int i = 0; i < editTexts.length; i++) {

for (int j = 0; j < editTexts[i].length; j++) {

array[i][j] = Double.valueOf(editTexts[i][j].getText().toString());

}

}

}

public static void getDoubleAnswer(EditText [][] editTexts, double [] array) {

for (int i = 0; i < editTexts.length; i++) {

array[i] = Double.valueOf(editTexts[i][0].getText().toString());

}

}

}

**GraphView**

public class pasteFromTask {

public static EditText[][] editTextsCoefficients = {{MainActivity.a1, MainActivity.b1, MainActivity.c1},

{MainActivity.a2, MainActivity.b2, MainActivity.c2},

{MainActivity.a3, MainActivity.b3, MainActivity.c3}};

public static double resEditTextsCoefficients[][] = new double[editTextsCoefficients.length][editTextsCoefficients[0].length];

public static EditText editTextAnswer[][] = {{MainActivity.d1}, {MainActivity.d2}, {MainActivity.d3}};

public static double resEditTextAnswer[][] = new double[editTextAnswer.length][editTextAnswer[0].length];

public static EditText editTextX[][] = {{MainActivity.x1}, {MainActivity.x2}, {MainActivity.x3}};

public static double resEditTextX[][] = new double[editTextX.length][editTextX[0].length];

public static void load(final View view, Context mine) {

if (ErrorsCheck.clearnessCheck(editTextAnswer) && ErrorsCheck.clearnessCheck(editTextsCoefficients)) {

loadProcedure();

Snackbar snack = Snackbar.make(view, R.string.loadComplete, Snackbar.LENGTH\_LONG);

snack.setAction(R.string.cancel, new View.OnClickListener() {

@Override

public void onClick(View v) {

restore(view);

}

});

snack.show();

} else {

AlertDialog.Builder ad = new AlertDialog.Builder(mine);

ad.setTitle(mine.getString(R.string.rewriteData));

ad.setPositiveButton(mine.getString(R.string.yes), new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

getResCoefficients(resEditTextAnswer, editTextAnswer);

getResCoefficients(resEditTextsCoefficients, editTextsCoefficients);

loadProcedure();

Snackbar snack = Snackbar.make(view, R.string.success, Snackbar.LENGTH\_LONG);

snack.setAction(R.string.cancel, new View.OnClickListener() {

@Override

public void onClick(View v) {

restore(view);

}

});

snack.show();

}

});

ad.setNegativeButton(mine.getString(R.string.No), new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

}

});

ad.setCancelable(false);

ad.show();

}

}

public static void loadProcedure() {

loadCoefficients();

loadAnswer();

}

public static void loadCoefficients() {

double toSet[][] = {{1, -3, 2}, {3, -4, 0}, {2, -5, 3}};

paste(editTextsCoefficients, toSet);

}

public static void loadAnswer() {

double toSet[][] = {{5}, {7}, {9}};

paste(editTextAnswer, toSet);

}

public static void paste(EditText[][] editTexts, double[][] toSet) {

for (int i = 0; i < editTexts.length; i++) {

for (int j = 0; j < editTexts[i].length; j++) {

if (toSet[i][j] == Double.MAX\_VALUE) editTexts[i][j].setText("");

else editTexts[i][j].setText(String.valueOf(toSet[i][j]));

}

}

}

public static void clear(final View view) {

if (!ErrorsCheck.clearnessCheck(editTextAnswer) && !ErrorsCheck.clearnessCheck(editTextsCoefficients)) {

getResCoefficients(resEditTextsCoefficients, editTextsCoefficients);

getResCoefficients(resEditTextAnswer, editTextAnswer);

if (!ErrorsCheck.clearnessCheck(editTextX)) {

getResCoefficients(resEditTextX, editTextX);

}

double toSetC[][] = new double[editTextsCoefficients.length][editTextsCoefficients[0].length];

double toSetA[][] = new double[editTextAnswer.length][editTextAnswer[0].length];

double toSetX[][] = new double[editTextX.length][editTextX[0].length];

setMaxValue(toSetX);

setMaxValue(toSetC);

setMaxValue(toSetA);

paste(editTextsCoefficients, toSetC);

paste(editTextAnswer, toSetC);

paste(editTextX, toSetX);

final Snackbar snack = Snackbar.make(view, R.string.ClearSuccess, Snackbar.LENGTH\_LONG);

snack.setAction(R.string.cancel, new View.OnClickListener() {

@Override

public void onClick(View v) {

restore(view);

}

});

snack.show();

} else Snackbar.make(view, R.string.clearingError, Snackbar.LENGTH\_LONG).show();

}

public static void setMaxValue(double toSetC[][]) {

for (int i = 0; i < toSetC.length; i++) {

for (int j = 0; j < toSetC[i].length; j++) {

toSetC[i][j] = Double.MAX\_VALUE;

}

}

}

public static void getResCoefficients(double[][] reservEditText, EditText[][] editTexts) {

for (int i = 0; i < editTexts.length; i++) {

for (int j = 0; j < editTexts[i].length; j++) {

reservEditText[i][j] = Double.valueOf(editTexts[i][j].getText().toString());

}

}

}

public static void restore(View view) {

if (!ErrorsCheck.clearnessCheckD(resEditTextAnswer) && !ErrorsCheck.clearnessCheckD(resEditTextsCoefficients)) {

paste(editTextsCoefficients, resEditTextsCoefficients);

paste(editTextAnswer, resEditTextAnswer);

if (!ErrorsCheck.clearnessCheckD(resEditTextX)) paste(editTextX, resEditTextX);

Snackbar.make(view, R.string.dataRestoringCompleted, Snackbar.LENGTH\_LONG).show();

} else

Snackbar.make(view, R.string.completedDataRestoring, Snackbar.LENGTH\_LONG).show();

}

**ІV. Висновок**

У ході виконання лабораторної роботи було використано метод Жордана-Гауса для обчислення систем лінійних алгебраїчних рівнянь з трьома невідомими.

**V. Результат**

