Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

*з предмету: «ТПКС»*

*по темі:*

«Автоматизація розмітки блок-схем алгоритмів»

Виконав: студент ФІОТ

групи ІО-92

Петрук В.О.

Київ 2012р.

**Мета:** Здобуття навичок з автоматизації процедури розмітки алгоритмів за

методами Мілі та Мура

**Завдання**

1. Представити номер залікової книжки в двійковому вигляді:

(NNNN)10=(n15n14…n2n1)2

2. В залежності від значення розрядів номера залікової книжки розмітити алгоритм,

перевірений на наявність помилок у попередній роботі згідно методу:

n4 xor n1 = 0 xor 1 = 1 -> Метод розмітки : Мура3. Згідно отриманої розмітки графічно відобразити граф переходів для алгоритму.

4. Розробити формат файлу для зберігання графу переході. Реалізувати функцію

збереження/відновлення графу переходів. Тип формату в залежності від розряду

номера залікової книжки: n2 = 1: Тип фрмату: текстовий.

**Опис програми**

Розмітка алгоритму згідно методу Мура відбувається на основі ЛСА та створеної раніше матричної форми ЛСА. Складається таблична форма автомату Мура у наступному форматі:

{Список логічних умов},{Список вершин автомату}

{}{Список сигналів вершин}

{Умови такту}{Вершини, куди йде перехід автомату}

На основі табличної форми будується граф автомата Мура. Графічно граф відображається за допомогою бібліотеки jgraphx (рис 1.)

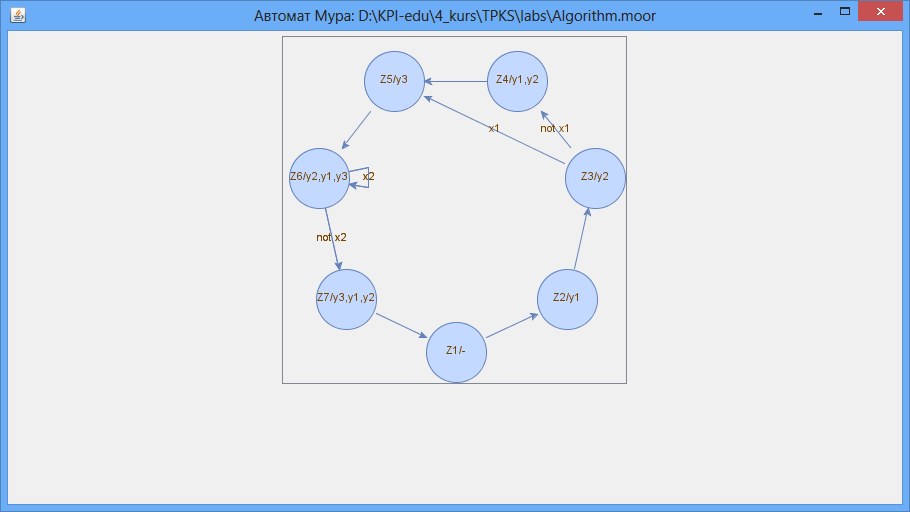


Рисунок 1 – Відображення графу автомата Мура

Для збереження та відновлення автомату Мура у вигляді текстового файлу необхідно вибрати пункти меню Файл «Зберегти автомат Мура» та « Відкрити автомат Мура» відповідно(рис 2). У діалоговому вікні необхідно ввести ім’я файлу.

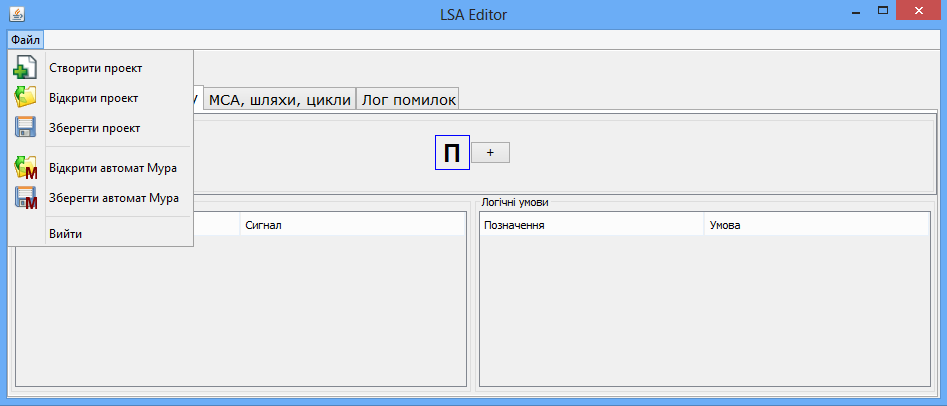


Рисунок 2 – Збереження та відкриття автомата Мура в текстовому файлі.

Для відображення графу автомата Мура поточного проекту необхідно натиснути кнопку D:\KPI-edu\4_kurs\TPKS\labs\src\lab1\icons\openMoorPanel.gifпанелі інструментів(рис 3.). Граф буде відображений в окремому вікні.

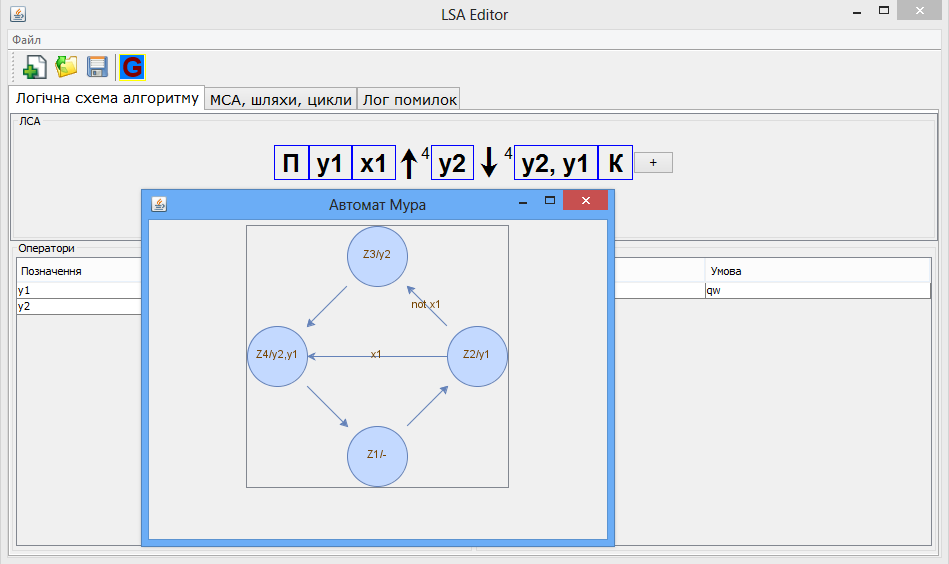


Рисунок 3 – Відображення графу автомата Мура поточного проекту

**Код програми**

Представлені лише класи та методи, які безпосередньо були реалізовані або змінені для виконання даної лабораторної роботи.

package lab1.graph;

import lab1.LSA;

import lab1.MSA;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class MooreMachine {

private ArrayList<String> moorGraphTable;

private String splitLine = ":";

private int[] rowType;

private int[][] msaTable;

private MSA msa;

//конструктор для створення нового автомату Мура.

public MooreMachine() {

}

// конструктор для створення нового автомату Мура на основі вже існуючої таблиці переходів

public MooreMachine(ArrayList<String> moorGraphTable) {

this.moorGraphTable = moorGraphTable;

}

//повертає табличну форму представлення даного автомата Мура

public ArrayList<String> getMoorGraphTable() {

return moorGraphTable;

}

//Створює і повертає табличну форму представлення автомата Мура.

public ArrayList<String> createGraph(LSA lsa) {

msa = lsa.getMsa();

moorGraphTable = new ArrayList<String>();

msaTable = msa.getUnitsUnion();

rowType = new int[msaTable.length];

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

if (isConditionRow(msaTable[i])) {

rowType[i] = 1;

} else {

rowType[i] = 0;

}

}

//заголовок + сигнали у вузлах

String titleRow = "";

String secondRow = "";

int countOfConditions = lsa.getConditionUnitsArray()[0].length;

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

titleRow += lsa.getConditionUnitsArray()[0][i] + splitLine;

secondRow += splitLine;

}

secondRow += "-" + splitLine;

titleRow += "Z1" + splitLine;//початкова вершина

int countOfOpertions = 1;

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

if (rowType[i] == 0) {

countOfOpertions++;

if (i == rowType.length - 1) {

titleRow += ("Z" + countOfOpertions);

secondRow += lsa.getOperationUnitMarking(countOfOpertions - 2);

} else {

titleRow += ("Z" + countOfOpertions + splitLine);

secondRow += lsa.getOperationUnitMarking(countOfOpertions - 2) + splitLine;

}

}

}

moorGraphTable.add(titleRow);

moorGraphTable.add(secondRow);

//переходи

//кодування умов

//+перехід з Z1(початок) в Z2

int numberOfRows = (int) Math.pow(2, countOfConditions);

String[] transitions = new String[countOfOpertions]; //0,1,10,11,100...

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

transitions[i] = getBinaryEncoding(i, countOfConditions);

if (countOfOpertions != 1) {

transitions[i] += "Z2" + splitLine;

}

}

//переходи між вершинами

for (int i = 0; i < msaTable.length; i++) {//прохід по вершинам

if (rowType[i] == 0) { //якщо Y

for (int j = 0; j < (int) (Math.pow(2, countOfConditions)); j++) { //прохід по станам 0,1,10,11...

Map<String, String> conditionSignal = setConditioSignals(transitions, j, countOfConditions, titleRow);

//куди переходимо

if (i == msaTable.length - 1) {

transitions[j] += whereItsGo(i, conditionSignal);

} else {

transitions[j] += whereItsGo(i, conditionSignal) + splitLine;

}

}

}

}

// copy transitions to moorGraphTable

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

moorGraphTable.add(transitions[i]);

}

// printMoorMachine();

return moorGraphTable;

}

//первірка чи даний рядок МСА є умовною вершиною

private boolean isConditionRow(int[] row) {

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

if (row[i] == -1) {

return true;

}

}

return false;

}

//повертає двійкову форму умов алгоритму для даної ітерацій

private String getBinaryEncoding(int idx, int numbOfConditions) {

String[] s = Integer.toBinaryString(idx).split("");

String sEnd = "";

for (int i = 0; i < numbOfConditions - s.length + 1; i++) {

sEnd += "0" + splitLine;

}

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

sEnd += s[i] + splitLine;

}

return sEnd;

}

//Встановлення значень бінарних сигналів для умовних вершин алгоритму. Повертає множину бінарних значень.

private Map<String, String> setConditioSignals(String[] transitions, int step, int countOfConditions, String titleRow) {

Map<String, String> conditionSignal = new HashMap<String, String>();

String[] titleArray = titleRow.split(splitLine);

String[] line = transitions[step].split(splitLine);

for (int j = 0; j < countOfConditions; j++) {

conditionSignal.put(titleArray[j], line[j]);

}

return conditionSignal;

}

//Повертає текстове значення вершини графа куди преходить вершина МСА з номером msaRow.

private String whereItsGo(int msaRow, Map<String, String> conditionSignal) {

String whereGo = "Z";

int whGo = -1;

whGo = defineWhereGo(msaRow, conditionSignal);

if (whGo != -2) {

int count = 0;

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

if (rowType[i] == 0) {

count++;

if (i == whGo) {

whereGo += "" + (count + 1);

break;

}

}

}

} else {

whereGo += "1";

}

return whereGo;

}

//Повертає позицію, куди йде перехід з даної вершини ЛСА.

private int defineWhereGo(int msaRow, Map<String, String> conditionSignal) {

if (msaRow == msaTable.length - 1) {

return -2;

}

int value = -2;

if (rowType[msaRow] == 0) {

value = 1;

} else {

String cond = msa.getUnitSignals().get(msaRow).getUnitMarking()[0];

value = Integer.parseInt(conditionSignal.get(cond));

if (value == 0) {

value = -1;

}

}

for (int i = 0; i < msaTable[msaRow].length; i++) {

if (msaTable[msaRow][i] == value) {

if (rowType[i] == 0) {

return i;

} else {

return defineWhereGo(i, conditionSignal);

}

}

}

return -1;

}

//Вивід в стандартний потік виводу матричну форму представлення автомата Мура.

public void printMoorMachine() {

for (String s : moorGraphTable) {

System.out.println(s);

}

}

}

package lab1.gui;

import com.mxgraph.layout.mxCircleLayout;

import com.mxgraph.layout.mxGraphLayout;

import com.mxgraph.swing.mxGraphComponent;

import com.mxgraph.util.mxConstants;

import com.mxgraph.view.mxGraph;

import com.mxgraph.view.mxStylesheet;

import lab1.graph.MooreMachine;

import javax.swing.\*;

import java.util.ArrayList;

import java.util.Hashtable;

public class GraphPanel extends JPanel {

ArrayList<String> vertexNames;//найменування вершин

ArrayList<String> vertexNamesPlus;//найменування вершин(з сигналами)

private int counter = 0; //кількість вершин;

private MooreMachine mooreMachine;

private String splitLine = ":";

private String[] firstLine;

private ArrayList<Integer> fromVertex;

private ArrayList<Integer> toVertex;

private String[] conditions;

private mxGraphComponent graphComponent;

// Конструктор панелі графу автомата Мура.

// mooreMachine - таблична форма автомату Мура

public GraphPanel(MooreMachine mooreMachine) {

this.mooreMachine = mooreMachine;

createGraph();

}

// Створення графу автомата МУра.

private void createGraph() {

// removeAll();

ArrayList<String> moorTable = mooreMachine.getMoorGraphTable();

//позначення вершин (+сигнали)

String[] secondLine = moorTable.get(1).split(splitLine);

int countOfConditions = 0;

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

if (!secondLine[i].equals("-")) {

countOfConditions++;

} else {

break;

}

}

firstLine = moorTable.get(0).split(splitLine);

vertexNames = new ArrayList<String>();

vertexNamesPlus = new ArrayList<String>();

int countOfVertexes = firstLine.length - countOfConditions;

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

vertexNames.add(firstLine[i + countOfConditions]);

vertexNamesPlus.add(firstLine[i + countOfConditions] + "/" + secondLine[i + countOfConditions]);

}

//встановлюємо переходи

defineEdges(moorTable, countOfConditions, countOfVertexes);

//рисуємо граф

drawGraph(countOfVertexes);

}

//Створення вершин графу.

private Integer defineVertex(int i, int countOfConditions, String[] line) {

String zName = line[countOfConditions + i];

for (int j = 0; j < vertexNames.size(); j++) {

if (vertexNames.get(j).equals(zName)) {

return j;

}

}

return -1;

}

//Створення переходів графу.

private void defineEdges(ArrayList<String> moorTable, int countOfConditions, int countOfVertexes) {

ArrayList<String[]> lines = new ArrayList<String[]>();

if (moorTable.size() != 2) {

for (int i = 2; i < moorTable.size(); i++) {

lines.add(moorTable.get(i).split(splitLine));

}

}

fromVertex = new ArrayList<Integer>();

toVertex = new ArrayList<Integer>();

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

for (int j = 0; j < Math.pow(2, countOfConditions); j++) {

fromVertex.add(i);

toVertex.add(defineVertex(i, countOfConditions, lines.get(j)));

if (fromVertex.size() > 1) {

int fvs = fromVertex.size();

int tvs = toVertex.size();

if ((fromVertex.get(fvs - 2) == fromVertex.get(fvs - 1)) & (toVertex.get(tvs - 2) == toVertex.get(tvs - 1))) {

fromVertex.remove(fvs - 1);

toVertex.remove(tvs - 1);

}

}

}

}

int[][] condIsActive = new int[countOfConditions][countOfVertexes];//масив значимості умов переходах

// 0 - активний

// 1 - перехід не залежить від умови

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

for (int j = 0; j < countOfConditions; j++) {

String toV = lines.get(0)[countOfConditions + i];

for (int k = 0; k < Math.pow(2, countOfConditions); k++) {

if (lines.get(k)[j].equals("1")) {

if (toV.equals(lines.get(k)[countOfConditions + i])) {

condIsActive[j][i] = 1;

break;

}

}

}

}

}

conditions = new String[fromVertex.size()];

for (int i = 0; i < fromVertex.size(); i++) {

conditions[i] = "";

for (int j = 0; j < countOfConditions; j++) {

if (condIsActive[j][fromVertex.get(i)] == 0) {

// умова чи антиумова

int row = -1;

for (int k = 0; k < Math.pow(2, countOfConditions); k++) {

if (lines.get(k)[countOfConditions + fromVertex.get(i)].equals(firstLine[countOfConditions + toVertex.get(i)])) {

if (lines.get(k)[j].equals("0")) {

if (!conditions[i].equals(firstLine[j])) {

conditions[i] += firstLine[j];

}

} else {

if (!conditions[i].equals("not " + firstLine[j])) {

conditions[i] += "not " + firstLine[j];

}

}

}

}

}

}

}

}

//рисування графу на панелі

public void drawGraph(int counter) {

mxGraph graph = new mxGraph();

Object parent = graph.getDefaultParent();

graph.getModel().beginUpdate();

try {

mxStylesheet stylesheet = graph.getStylesheet();

Hashtable<String, Object> vertexStyle = new Hashtable<String, Object>();

vertexStyle.put(mxConstants.STYLE\_SHAPE, mxConstants.SHAPE\_ELLIPSE);

vertexStyle.put(mxConstants.STYLE\_OPACITY, 100);

stylesheet.putCellStyle("ROUNDED", vertexStyle);

Hashtable<String, Object> arrowStyle = new Hashtable<String, Object>();

arrowStyle.put(mxConstants.STYLE\_EDGE, mxConstants.EDGESTYLE\_TOPTOBOTTOM);

arrowStyle.put(mxConstants.STYLE\_OPACITY, 100);

arrowStyle.put(mxConstants.STYLE\_FONTCOLOR, "#774400");

stylesheet.putCellStyle("ARROW", arrowStyle);

ArrayList<Object> vertexes = new ArrayList<Object>();

ArrayList<Object> edges = new ArrayList<Object>();

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

Object vertex = graph.insertVertex(parent, null, vertexNamesPlus.get(i), 60, 60, 60, 60, "ROUNDED");

vertexes.add(vertex);

}

//рисуємо переходи

for (int i = 0; i < fromVertex.size(); i++) {

if (toVertex.get(i) != -1) {

Object edge = graph.insertEdge(parent, null, conditions[i], vertexes.get(fromVertex.get(i)),

vertexes.get(toVertex.get(i)), "ARROW");

edges.add(edge);

}

}

} finally {

graph.getModel().endUpdate();

}

graph.setCellsEditable(false);

graph.setCellsResizable(false);

graph.setCellsBendable(false);

graph.setConnectableEdges(false);

graphComponent = new mxGraphComponent(graph);

add(graphComponent);

mxGraphLayout graphLayout = new mxCircleLayout(graph);

graphLayout.execute(parent);

}

}

package lab1.file;

import lab1.Unit;

import lab1.graph.MooreMachine;

import lab1.gui.GraphPanel;

import lab1.gui.MainPanel;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.ArrayList;

public class OpenMoorListener extends JFrame implements ActionListener {

/\*\*

\* Конструктор обробника кнопки відкриття файлу.

\*/

public OpenMoorListener() {

setTitle("Автомат Мура");

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

setSize\_Position();

}

/\*\*

\* Обробник кнопки відкриття файлу.

\*/

public void actionPerformed(ActionEvent e) {

JFileChooser chooser = new JFileChooser();

chooser.setCurrentDirectory(new File("."));

chooser.setSelectedFile(new File(".moor"));

int result = chooser.showOpenDialog(null);

if (result != JFileChooser.CANCEL\_OPTION) {

String fileName = chooser.getSelectedFile().getPath();

try {

createMoorFrame(fileName);

setTitle("Автомат Мура: "+fileName);

} catch (IOException e1) {

JOptionPane.showMessageDialog(null, ("Не вдалося відкрити файл " + fileName));

} catch (ClassNotFoundException e1) {

e1.printStackTrace(); //To change body of catch statement use File | Settings | File Templates.

}

}

}

/\*\*

\* Десереалізує обєкт класу MSA з файлу.

\*

\* @param fileName

\* @throws java.io.IOException

\* @throws ClassNotFoundException

\*/

private void createMoorFrame(String fileName) throws IOException, ClassNotFoundException {

BufferedReader moorFile = new BufferedReader (new FileReader(fileName));

ArrayList<String> newMoorTable = new ArrayList<String>();

String line = "";

while (null!=(line = moorFile.readLine())){

newMoorTable.add(line);

}

for (int i = 0; i < newMoorTable.size(); i++) {

System.out.println(newMoorTable.get(i));

}

GraphPanel graphPanel = new GraphPanel(new MooreMachine(newMoorTable));

add(graphPanel);

setVisible(true);

}

// відображення вікна з графом автомата Мура, створеного на основі табличної форми графу

public void createMoorFrame(ArrayList<String> newMoorTable){

for (int i = 0; i < newMoorTable.size(); i++) {

System.out.println(newMoorTable.get(i));

}

GraphPanel graphPanel = new GraphPanel(new MooreMachine(newMoorTable));

add(graphPanel);

setVisible(true);

}

private void setSize\_Position() {

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int frameWidth = screenSize.width \* 2 / 3;

int frameHeight = screenSize.height \* 2 / 3;

int leftCornerH = (screenSize.height - frameHeight) / 2;

int leftCornerW = (screenSize.width - frameWidth) / 2;

setBounds(leftCornerW, leftCornerH, frameWidth, frameHeight);

setPreferredSize(new Dimension(frameWidth, frameHeight));

setLayout(new BorderLayout());

}

}

public class MainFrame extends JFrame {

/\*\*

\* Створення панелі інструментів.

\*/

private void createToolBar() {

JToolBar jtb = new JToolBar("Панель інструментів");

jtb.setOrientation(SwingConstants.HORIZONTAL);

//new

JButton newButton;

Icon newIcon = new ImageIcon("src/lab1/icons/new.png");

newButton = new JButton(newIcon);

newButton.setToolTipText("Створити новий проект");

newButton.addActionListener(new NewProjectListener(this.mainPanel));

jtb.add(newButton);

//open

JButton openButton;

Icon openIcon = new ImageIcon("src/lab1/icons/open.gif");

openButton = new JButton(openIcon);

openButton.setToolTipText("Відкрити раніше створений проект");

openButton.addActionListener(new OpenButtonListener(this.mainPanel));

jtb.add(openButton);

//save

Icon saveIcon = new ImageIcon("src/lab1/icons/save.png");

JButton saveButton = new JButton(saveIcon);

saveButton.setToolTipText("Зберегти проект");

saveButton.addActionListener(new SaveButtonListener(mainPanel));

jtb.add(saveButton);

jtb.addSeparator();

Icon mooreIcon = new ImageIcon("src/lab1/icons/openMoorPanel.gif");

JButton showMoorGraphButton = new JButton(mooreIcon);

showMoorGraphButton.setBorder(BorderFactory.createLineBorder(Color.YELLOW));

showMoorGraphButton.setFont(new Font("TimesRoman", Font.BOLD, 14));

showMoorGraphButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

OpenMoorListener openMoorListener = new OpenMoorListener();

MooreMachine mooreMachine = new MooreMachine();

openMoorListener.createMoorFrame(mooreMachine.createGraph(mainPanel.getThisLsa()));

}

});

jtb.add(showMoorGraphButton);

add(jtb, BorderLayout.NORTH);

}

**Висновок**

Розроблені та реалізовані алгоритми процедур побудови автомату Мура на основі логічної схеми алгоритмів. Результати роботи процедур (граф автомата Мура) показуються в графічному інтерфейсі користувача.

Розроблено процедури збереження та відновлення автомата Мура в табличному представленні в текстовий файл. Реаліована можливість преглянути збережені алгоритми у вигляді графу в графічному інтерефесі користувача.