

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №1

З КУРСУ

«ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ»

Виконав:

студент IV курсу

групи ІО-21

Журо Георгій Олександрович

Київ – 2015

**Тема:** Редактор блок-схем алгоритмів.

**Мета роботи:** Здобуття навичок з побудови редактора блок-схем алгоритмів. Розробка інтерфейсу користувача та функціонального наповнення. Розробка засобів перетворення форматів зберігання даних.

**Завдання:**

1. Представити номер залікової книжки в двійковому вигляді:

(NNNN)10=(n15n14…n2n1)2

1. В залежності від молодшого розряду номера залікової книжки визначити тип редактора.
2. Розробити формат файлу для зберігання схеми алгоритму у матричному вигляді (матриця зв’язків між вузлами та матриця співвідношення сигналів та вузлів).
3. Реалізувати редактор алгоритму заданого типу (п.2) з можливістю збереження/відновлення результатів роботи програми у матричному вигляді згідно розробленого формату (п.3). Передбачити в редакторі наступні функції:
   * створення нової блок схеми алгоритму;
   * модифікація алгоритму (створення/видалення початкового, кінцевого, логічних(X) та операційних(Y) вузлів, редагування сигналів вузлів, створення/видалення зв’язків між вузлами);
   * контроль вводу (тільки один початковий та один кінцевий вузли, логічні та операційні вузли мають містити не менш одного сигналу, логічні вузли містять тільки вхідні сигнали, а операційні – тільки вихідні, тощо);
   * збереження алгоритму у матричному вигляді згідно (п.3);
   * відновлення алгоритму з матричного вигляду згідно (п.3).

**Варіант завдання:**

Тип редактору: Редактор графічних схем алгоритмів (ГСА).

Тип формату: Текстовий.

**Код програми:**

import face.MainFrame;

import javax.swing.\*;

import java.awt.\*;

public class Program {

private static int MIN\_WIDTH = 800;

private static int MIN\_HEIGHT = 600;

private static Rectangle getDefaultBounds() {

Toolkit kit = Toolkit.getDefaultToolkit();

Dimension screenSize = kit.getScreenSize();

int width = (int) (screenSize.getWidth() / 10 \* 8);

if (width < MIN\_WIDTH) {

width = MIN\_WIDTH;

}

int height = (int) (screenSize.getHeight() / 10 \* 8);

if (height < MIN\_HEIGHT) {

height = MIN\_HEIGHT;

}

return new Rectangle(((int) screenSize.getWidth() - width) / 2, ((int) screenSize.getHeight() - height) / 2,

width, height);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

MainFrame frame = new MainFrame(getDefaultBounds());

frame.setMinimumSize(new Dimension(MIN\_WIDTH, MIN\_HEIGHT));

frame.setVisible(true);

}

});

}

}

package face;

import gsa.GSAFileFilter;

import gsa.GSAModel;

import gsa.GSAPanel;

import gsa.IllegalNodeException;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.\*;

import java.util.Scanner;

public class MainFrame extends JFrame {

private JMenuBar menuBar;

private JToolBar toolBar;

private GSAPanel gsaPanel;

private JLabel statusLabel;

private JFileChooser chooser;

private NewAction newAction;

private OpenAction openAction;

private SaveAction saveAction;

private SaveAsAction saveAsAction;

private CloseAction closeAction;

private ExitAction exitAction;

private AboutAction aboutAction;

private File openedFile;

public MainFrame(Rectangle bounds) {

super();

setBounds(bounds);

setTitle("Technology Designing of Computer Systems | Lab1 | Zhuro George, IO-21");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

newAction = new NewAction(this);

openAction = new OpenAction(this);

saveAction = new SaveAction(this);

saveAsAction = new SaveAsAction(this);

closeAction = new CloseAction(this);

exitAction = new ExitAction(this);

aboutAction = new AboutAction(this);

addWindowListener(new WindowHandler(exitAction));

menuBar = new JMenuBar();

JMenu fileMenu = new JMenu("File");

JMenu helpMenu = new JMenu("Help");

JMenuItem tempItem = new JMenuItem(newAction);

tempItem.setText("New");

fileMenu.add(tempItem);

tempItem = new JMenuItem(openAction);

tempItem.setText("Open...");

fileMenu.add(tempItem);

tempItem = new JMenuItem(saveAction);

tempItem.setText("Save");

fileMenu.add(tempItem);

tempItem = new JMenuItem(saveAsAction);

tempItem.setText("Save As...");

fileMenu.add(tempItem);

tempItem = new JMenuItem(closeAction);

tempItem.setText("Close");

fileMenu.add(tempItem);

fileMenu.addSeparator();

tempItem = new JMenuItem(exitAction);

tempItem.setText("Exit");

fileMenu.add(tempItem);

tempItem = new JMenuItem(aboutAction);

tempItem.setText("About...");

helpMenu.add(tempItem);

menuBar.add(fileMenu);

menuBar.add(helpMenu);

setJMenuBar(menuBar);

setLayout(new BorderLayout());

toolBar = new JToolBar(JToolBar.VERTICAL);

toolBar.setFloatable(false);

toolBar.setRollover(true);

JButton tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(0);

statusLabel.setText(" ");

}

});

tempButton.setIcon(new ImageIcon("img/no\_action.png"));

tempButton.setToolTipText("No action");

toolBar.addSeparator();

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(1);

statusLabel.setText("Adding Begin Node");

}

});

tempButton.setIcon(new ImageIcon("img/begin\_node.png"));

tempButton.setToolTipText("Begin Node");

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(2);

statusLabel.setText("Adding Operator Node");

}

});

tempButton.setIcon(new ImageIcon("img/operator\_node.png"));

tempButton.setToolTipText("Operator Node");

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(3);

statusLabel.setText("Adding Logic Node");

}

});

tempButton.setIcon(new ImageIcon("img/logic\_node.png"));

tempButton.setToolTipText("Logic Node");

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(4);

statusLabel.setText("Adding End Node");

}

});

tempButton.setIcon(new ImageIcon("img/end\_node.png"));

tempButton.setToolTipText("End Node");

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(6);

statusLabel.setText("Connecting nodes");

}

});

tempButton.setIcon(new ImageIcon("img/line.png"));

tempButton.setToolTipText("Connect nodes");

toolBar.addSeparator();

tempButton = toolBar.add(new AbstractAction() {

public void actionPerformed(ActionEvent e) {

gsaPanel.getModel().setActionType(5);

statusLabel.setText("Deleting");

}

});

tempButton.setIcon(new ImageIcon("img/delete\_node.png"));

tempButton.setToolTipText("Delete");

add(toolBar, BorderLayout.WEST);

statusLabel = new JLabel(" ");

add(statusLabel, BorderLayout.SOUTH);

gsaPanel = new GSAPanel(new GSAModel(), this);

add(new JScrollPane(gsaPanel));

chooser = new JFileChooser();

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

chooser.addChoosableFileFilter(new GSAFileFilter());

chooser.setMultiSelectionEnabled(false);

openedFile = null;

}

private class NewAction extends AbstractAction {

private MainFrame frame;

public NewAction(MainFrame frame) {

super();

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

if (gsaPanel.getModel().isChanged()) {

closeAction.actionPerformed(e);

}

gsaPanel.setModel(new GSAModel());

gsaPanel.setVisible(true);

statusLabel.setText(" ");

frame.repaint();

}

}

private class OpenAction extends AbstractAction {

private MainFrame frame;

public OpenAction(MainFrame frame) {

super();

this.frame = frame;

}

public void actionPerformed(ActionEvent e) {

int result = chooser.showOpenDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

try {

Scanner input = new Scanner(new BufferedReader(new FileReader(chooser.getSelectedFile())));

int n = input.nextInt();

int[] nodesTypeMatrix = new int[n];

int[][] boundsMatrix = new int[n][];

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

boundsMatrix[i] = new int[4];

}

int[][] connectivityMatrix = new int[n][];

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

connectivityMatrix[i] = new int[n];

}

int[][] signalMatrix = new int[n][];

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

nodesTypeMatrix[i] = input.nextInt();

}

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

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

boundsMatrix[i][j] = input.nextInt();

}

}

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

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

connectivityMatrix[i][j] = input.nextInt();

}

}

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

signalMatrix[i] = new int[input.nextInt()];

}

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

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

signalMatrix[i][j] = input.nextInt();

}

}

try {

gsaPanel.getModel().setGSA(nodesTypeMatrix, connectivityMatrix, signalMatrix, boundsMatrix);

gsaPanel.setVisible(true);

openedFile = chooser.getSelectedFile();

input.close();

statusLabel.setText(" ");

frame.repaint();

} catch (IllegalNodeException e1) {

JOptionPane.showMessageDialog(frame, "Error! Incorrect GSA.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (FileNotFoundException e1) {

JOptionPane.showMessageDialog(frame, "Error! Can't open selected file.",

"Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

**Висновок:**

Під час виконання лабораторної роботи я отримав навички з побудови редактора блок-схем алгоритмів. Розробки інтерфейсу користувача та функціонального наповнення. Розробки засобів перетворення форматів зберігання даних.