

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №2

З КУРСУ

«ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ»

Виконав:

студент IV курсу

групи ІО-21

Журо Георгій Олександрович

Київ – 2015

**Тема:** Автоматизація аналізу блок-схем алгоритмів.

**Мета:** Здобуття навичок з розробки та реалізації методів перевірки на помилки блок-схем алгоритмів.

**Завдання:**

1. Реалізувати процедуру пошуку всіх шляхів та циклів у блок-схемі алгоритму, представленому в матричній формі.
2. В залежності від значення розрядів номера залікової книжки виявити наступні помилки в алгоритмі:

**Тип перевірки:** Виявити нескінченні цикли.

1. При наявності помилок локалізувати місце помилки, виділити його у редакторі алгоритму, надати можливість корекції та повторної перевірки.

**Опис програми:**

Для побудови та редагування блок-схем алгоритмів використовується програма з лабораторної роботи №1. Для реалізації процедури пошуку всіх шляхів та циклів у блок-схемі був написаний клас GSAWorker, що містить метод findWaysAndLoops(), що рекурсивно викликає приватний метод step(). З результатів виклику методу findWaysAndLoops() можна виявити нескінчені цикли в алгоритмі. Цикли є нескінченними, якщо в блок-схемі не знайдено ні одного шляху, але цикли є.

Для пошуку всіх шляхів та циклів в алгоритмі необхідно вибрати пункт головного меню “Analyze” -> ”Find All Ways And Loops…”. Результат роботи для блок-схеми алгоритму, що зображений на рисунку 1, показаний на рисунку 2.

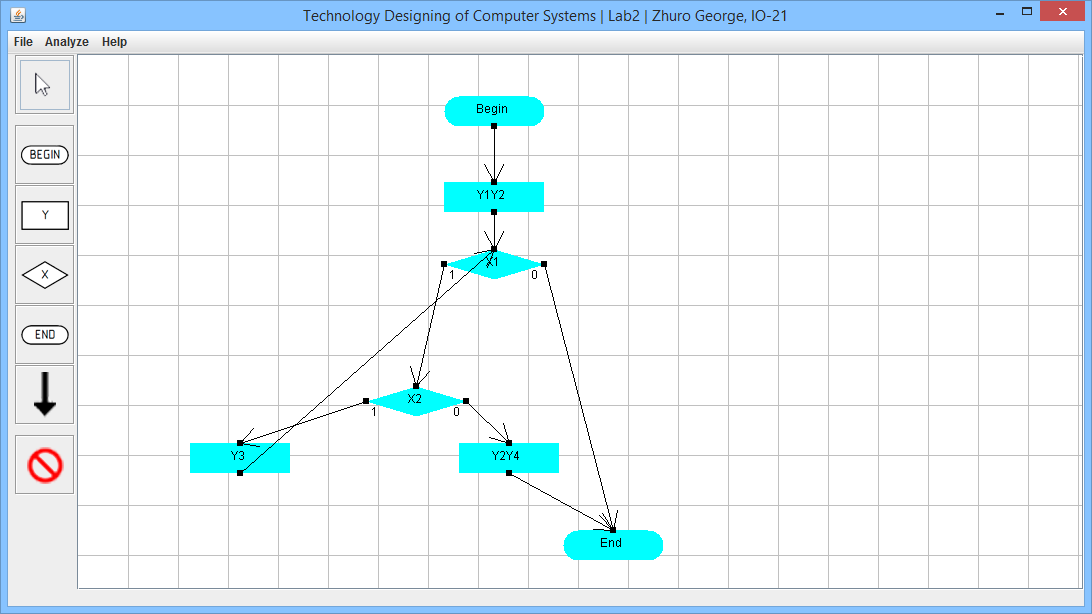


Рисунок 1

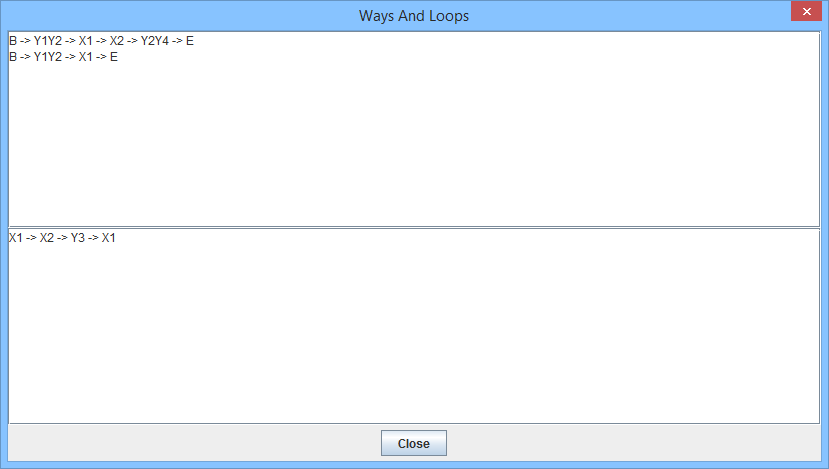


Рисунок 2

Для виявлення нескінченних циклів необхідно вибрати пункт головного меню “Analyze” -> ”Check GSA For Infinite Loops”. Для коректного алгоритму буде виведено повідомлення, що зображено на рисунку 3.

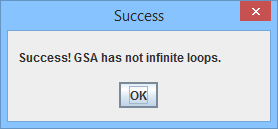


Рисунок 3

Для блок-схеми алгоритму, що містить нескінчені цикли, буде виведено повідомлення, що зображено на рисунку 4, та ці цикли будуть виділені в редакторі (рисунок 5).

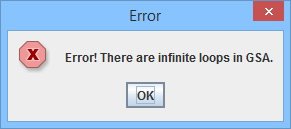


Рисунок 4

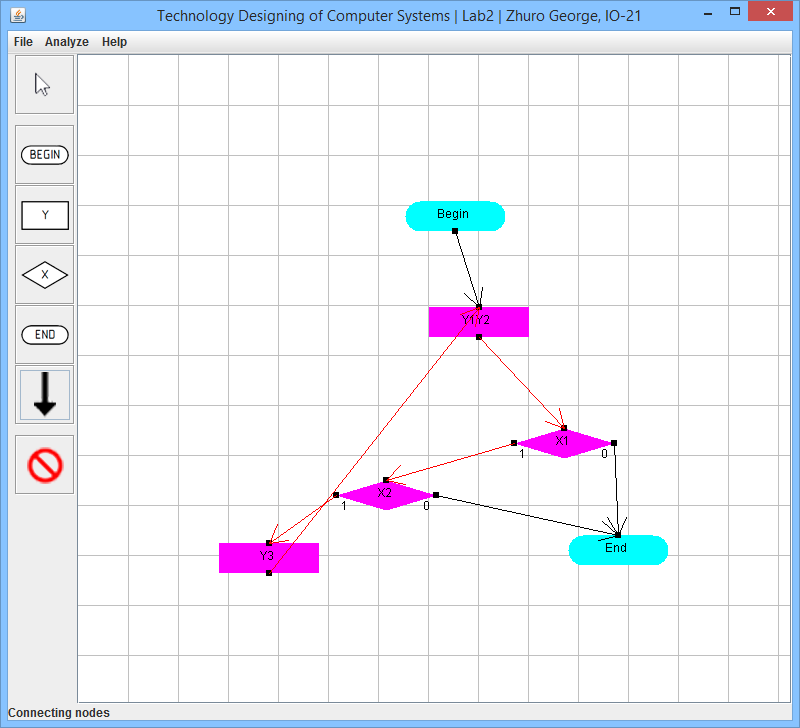


Рисунок 5

**Лістинг програми:**

package gsa;

import java.util.ArrayList;

/\*\*

\* Created by IntelliJ IDEA.

\* User: Zak

\* Date: 03.10.2010

\* Time: 12:24:35

\* To change this template use File | Settings | File Templates.

\*/

public class GSAWorker {

private int[] nodesType;

private int[][] connectionMatrix;

private int[][] signals;

private int beginNode;

private int endNode;

private ArrayList<ArrayList<Integer>> waysNodes;

private ArrayList<ArrayList<Integer>> loopsNodes;

private ArrayList<String> ways;

private ArrayList<String> loops;

public GSAWorker(int[] nodesType, int[][] connectionMatrix, int[][] signals) {

this.nodesType = nodesType;

this.connectionMatrix = connectionMatrix;

this.signals = signals;

}

private void step(int currentNode, ArrayList<Integer> wayNodes, boolean[] visitedNodes, String way) {

if (currentNode == endNode) {

wayNodes.add(currentNode);

way = way + " -> E";

visitedNodes[currentNode] = true;

ways.add(way);

waysNodes.add(wayNodes);

} else {

if (!visitedNodes[currentNode]) {

wayNodes.add(currentNode);

visitedNodes[currentNode] = true;

way = way + " -> ";

if (nodesType[currentNode] == 1) {

for (int i = 0; i < signals[currentNode].length; i++) {

way = way + "Y" + String.valueOf(signals[currentNode][i]);

}

} else {

if (nodesType[currentNode] == 2) {

way = way + "X" + String.valueOf(signals[currentNode][0]);

}

}

boolean findChild = false;

int i = 0;

while (((!findChild) || (nodesType[currentNode] == 2)) && (i < connectionMatrix[currentNode].length)) {

if (connectionMatrix[currentNode][i] > 0) {

ArrayList<Integer> wayNodesClone = new ArrayList<Integer>();

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

wayNodesClone.add(wayNodes.get(j));

}

boolean[] visitedNodesClone = new boolean[visitedNodes.length];

for (int j = 0; j < visitedNodes.length; j++) {

visitedNodesClone[j] = visitedNodes[j];

}

String wayClone = new String(way);

findChild = true;

step(i, wayNodesClone, visitedNodesClone, wayClone);

}

i++;

}

} else {

wayNodes.add(currentNode);

int i = -1;

boolean found = false;

while (!found)

{

i++;

if (wayNodes.get(i) == currentNode) {

found = true;

}

}

ArrayList<Integer> loopNodes = new ArrayList<Integer> ();

loopNodes.add(wayNodes.get(i));

StringBuilder builder = new StringBuilder();

if (nodesType[wayNodes.get(i)] == 1) {

for (int k = 0; k < signals[wayNodes.get(i)].length; k++) {

builder.append("Y");

builder.append(String.valueOf(signals[wayNodes.get(i)][k]));

}

}

else {

if (nodesType[wayNodes.get(i)] == 2) {

builder.append("X");

builder.append(String.valueOf(signals[wayNodes.get(i)][0]));

}

}

for (int j = i + 1; j < wayNodes.size(); j++) {

loopNodes.add(wayNodes.get(j));

builder.append(" -> ");

if (nodesType[wayNodes.get(j)] == 1) {

for (int k = 0; k < signals[wayNodes.get(j)].length; k++) {

builder.append("Y");

builder.append(String.valueOf(signals[wayNodes.get(j)][k]));

}

}

else {

if (nodesType[wayNodes.get(j)] == 2) {

builder.append("X");

builder.append(String.valueOf(signals[wayNodes.get(j)][0]));

}

}

}

loops.add(builder.toString());

loopsNodes.add(loopNodes);

}

}

}

package gsa;

/\*\*

\* Created by IntelliJ IDEA.

\* User: Zak

\* Date: 07.10.2010

\* Time: 5:44:47

\* To change this template use File | Settings | File Templates.

\*/

public class BeginEndNodesNotFoundException extends Exception {

private static String BEGIN\_TEXT = "Begin node ";

private static String END\_TEXT = "End node ";

private static String BEGIN\_END\_TEXT = "Begin and End nodes ";

private static String COMMON\_TEXT1 = "has not been found.";

private static String COMMON\_TEXT2 = "have not been found.";

private String text;

public BeginEndNodesNotFoundException(boolean beginNode, boolean endNode) {

if (beginNode && endNode) {

text = BEGIN\_END\_TEXT + COMMON\_TEXT2;

}

else {

if (beginNode) {

text = BEGIN\_TEXT + COMMON\_TEXT1;

}

else {

if (endNode) {

text = END\_TEXT + COMMON\_TEXT1;

}

}

}

}

@Override

public String getMessage() {

return text;

}

}

package face;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.util.ArrayList;

/\*\*

\* Created by IntelliJ IDEA.

\* User: Zak

\* Date: 07.10.2010

\* Time: 0:05:33

\* To change this template use File | Settings | File Templates.

\*/

class WaysAndLoopsFrame extends JDialog {

private JTextArea waysArea;

private JTextArea loopsArea;

private JButton closeButton;

public WaysAndLoopsFrame(MainFrame frame, Rectangle bounds, ArrayList<String> ways, ArrayList<String> loops) {

super(frame);

setBounds(bounds);

setResizable(false);

setModal(true);

setTitle("Ways And Loops");

setLayout(new BorderLayout());

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new GridLayout(2, 1));

waysArea = new JTextArea();

waysArea.setEditable(false);

StringBuilder builder = new StringBuilder();

if (!ways.isEmpty()) {

for (int i = 0; i < (ways.size() - 1); i++) {

builder.append(ways.get(i));

builder.append("\n");

}

builder.append(ways.get(ways.size() - 1));

}

waysArea.setText(builder.toString());

builder = new StringBuilder();

loopsArea = new JTextArea();

loopsArea.setEditable(false);

if (!loops.isEmpty()) {

for (int i = 0; i < (loops.size() - 1); i++) {

builder.append(loops.get(i));

builder.append("\n");

}

builder.append(loops.get(loops.size() - 1));

}

loopsArea.setText(builder.toString());

mainPanel.add(new JScrollPane(waysArea));

mainPanel.add(new JScrollPane(loopsArea));

closeButton = new JButton(new AbstractAction () {

public void actionPerformed(ActionEvent e) {

setVisible(false);

}

});

closeButton.setText("Close");

JPanel buttonPanel = new JPanel();

buttonPanel.add(closeButton);

add(buttonPanel, BorderLayout.SOUTH);

add(mainPanel);

}

}

**Висновки:** В результаті даної лабораторної роботи я здобув навички з розробки методів перевірки на помилки блок-схем алгоритмів. Для побудови блок-схем алгоритмів використовувалася програма з лабораторної роботи №1. Мною був написаний клас, що містив методи для рекурсивного алгоритму пошуку всіх шляхів та циклів на блок-схемі. За допомогою результатів виконання цього пошуку можна виявити нескінчені цикли в алгоритмі. Також мною були написані методи для виділення помилок в редакторі. Для цього я використав засоби бібліотеки Swing.