МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

(САМАРСКИЙ УНИВЕРСИТЕТ)

Отчёт по лабораторной работе №3  
по курсу «Моделирование информационно-вычислительных систем»

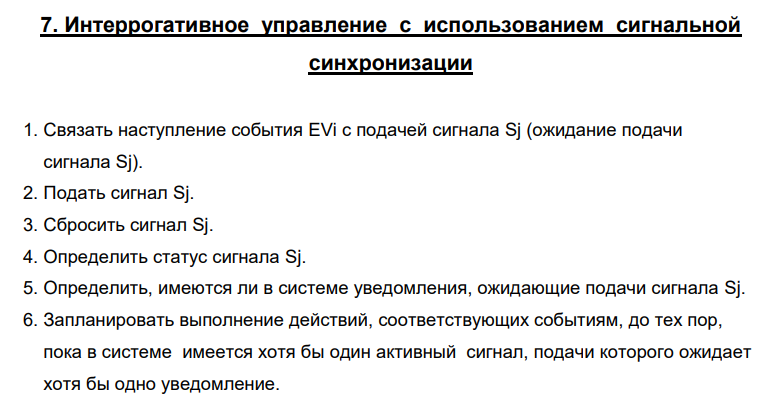
Выполнил:

Алеев И.И.

студент гр. №6304-090301D  
Проверила:

Симонова Е.В.

# Задание



**Описание реализации дискретно-событийного монитора(классов)**

Переход к следующему моменту времени происходит после нажатия на клавишу. В процессе перехода генерируются новые события и уведомления, которые связаны с ними, которые потом заносятся в соответствующие списки.

Сигналы Sj подаются и сбрасываются в случайном порядке, после генерации сигналов определяется их статус, и в соответствии с ним вызывается событие EVi (класс Event) по уведомлению (класс Event2).

После того, как все сигналы были поданы на ожидающие уведомления и были вызваны все события, связанные с ними, монитор сообщает об этом и начинает отчёт заново, сбросив всё.

# Листинг программы моделирования

Класс двусвязного циклического списка:

namespace LAB3

{

class DoubleLinkedList<T>

{

private DoubleLinkedListNode<T> Head;

public DoubleLinkedListNode<T> head

{

get

{

return Head;

}

set

{

Head = value;

}

}

public DoubleLinkedList()

{

head = new DoubleLinkedListNode<T>();

head.Next = head;

head.Prev = head;

}

public void Insert(T v)

{

if (head != null)

{

var p = new DoubleLinkedListNode<T>(v);

p.Next = head;

p.Prev = head.Prev;

head.Prev.Next = p;

head.Prev = p;

p.Val = v;

}

}

public void Clear()

{

head = new DoubleLinkedListNode<T>();

head.Next = head;

head.Prev = head;

}

}

class DoubleLinkedListNode<T>

{

T val ;

public DoubleLinkedListNode<T> Next { get; set; }

public DoubleLinkedListNode<T> Prev { get; set; }

public T Val { get => val; set => val = value; }

public DoubleLinkedListNode()

{

Next = null;

Prev = null;

val = default(T);

}

public DoubleLinkedListNode(T val\_)

{

val = val\_;

}

public DoubleLinkedListNode(T val\_, DoubleLinkedListNode<T> next, DoubleLinkedListNode<T> prev)

{

val = val\_;

Next = next;

Prev = prev;

}

}

}

Классы для отображения непосредственно событий, и уведомлений:

namespace LAB3

{

class Event1

{

private string name;

private int status;

private int notification;

public Event1(string name, int status, int notice)

{

this.name = name;

this.status = status;

this.notification = notice;

}

public string getName()

{

return name;

}

public int getStatus()

{

return status;

}

public int getNotification()

{

return notification;

}

public Event1() { }

public Event1(string name, int status, int notice, Event1 next, Event1 prev)

{

this.name = name;

this.status = status;

this.notification = notice;

}

}

class Event2

{

private string name;

private int time;

private int status;

private int moment;

public Event2(string name, int status, int time, int moment)

{

this.name = name;

this.time = time;

this.status = status;

this.moment = moment;

}

public string getName()

{

return name;

}

public int getStat()

{

return status;

}

public int getTime()

{

return time;

}

public int Moment()

{

return moment;

}

public Event2() { }

public Event2(string name, int status, int time, int moment, Event2 next, Event2 prev)

{

this.name = name;

this.status = status;

this.time = time;

this.moment = moment;

}

}

}

Монитор:

using System;

using System.Windows.Forms;

namespace LAB3

{

public partial class Form1 : Form

{

DoubleLinkedList<Event1> signal = new DoubleLinkedList<Event1>();

DoubleLinkedList<Event2> notice = new DoubleLinkedList<Event2>();

Random random;

private string[] subj = new string[4];

private int[] status = new int[4];

private int[] notification = new int[4] {1,1,1,1};

private int[] a = new int[4];

private int[] b = new int[4];

private int[] moment = new int[4];

public Form1()

{

InitializeComponent();

}

public void Signals()

{

var p = signal.head.Next;

while (p != signal.head)

{

var arr = new string[3];

arr[0] = p.Val.getName();

arr[1] = p.Val.getStatus().ToString();

arr[2] = p.Val.getNotification().ToString();

listViewEvent.Items.Add(new ListViewItem(arr));

p = p.Next;

}

}

public void Notifications()

{

var q = notice.head.Next;

while (q != notice.head)

{

var arr = new string[4];

arr[0] = q.Val.getName();

arr[1] = q.Val.getStat().ToString();

arr[2] = q.Val.getTime().ToString();

arr[3] = q.Val.Moment().ToString();

listView.Items.Add(new ListViewItem(arr));

q = q.Next;

}

}

private void buttonGenerate\_Click(object sender, EventArgs e)

{

if ((notification[0] == 0) && (notification[1] == 0) && (notification[2] == 0) && (notification[3] == 0))

{

listView.Items.Clear();

listViewEvent.Items.Clear();

signal.Clear();

notice.Clear();

for (int i = 0; i < 4; i++)

{

a[i] = 0;

b[i] = 0;

moment[i] = 0;

}

}

subj[0] = "ООП";

subj[1] = "Ассемблер";

subj[2] = "Комп. сети";

subj[3] = "OS";

random = new Random();

int moments = 0;

for (int i = 0; i < 4; i++)

{

status[i] = random.Next(0, 2);

notification[i] = random.Next(0, 2);

Event1 evi = new Event1(subj[i], status[i], notification[i]);

signal.Insert(evi);

b[i] = b[i] + status[i];

a[i] = a[i] + notification[i];

moment[i] = b[i] + a[i];

moments += moment[i];

var v = new Event2(subj[i], b[i], a[i], moment[i]);

notice.Insert(v);

}

Sum.Text = "Время моделирования: " + moments.ToString();

listViewEvent.Items.Clear();

listView.Items.Clear();

Signals();

Notifications();

pictureBoxYellow.Visible = status[0] == 1;

label1.Visible = status[0] == 1;

pictureBoxBlue.Visible = status[1] == 1;

label2.Visible = status[1] == 1;

pictureBoxGreen.Visible = status[2] == 1;

label7.Visible = status[2] == 1;

pictureBoxBlack.Visible = status[3] == 1;

label8.Visible = status[3] == 1;

if ((notification[0] == 0) && (notification[1] == 0) && (notification[2] == 0) && (notification[3] == 0))

{

MessageBox.Show("Все уведомления обработаны!");

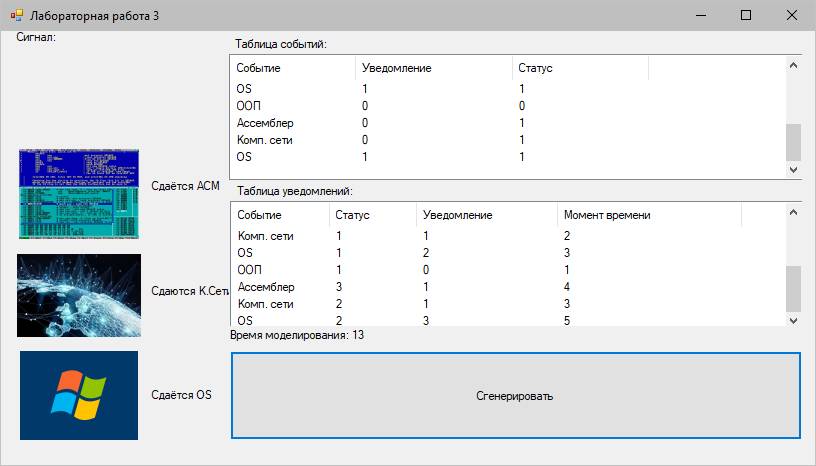
}

}

}

}

# Результат моделирования



В данной программе для каждого из 4 видов событий существует соответствующее изображение, видимость которого отображает статус события (происходит оно или нет).

При нажатии кнопки «Сгенерировать», к списку текущих событий добавляются новые и генерируются новые уведомления, в соответствии к которым были поставлены события