Белорусский государственный университет информатики и радиоэлектроники

Кафедра ПОИТ

Отчет по лабораторной работе № 3

«Аналитическое моделирование дискретно-стохастической СМО и построение её имитационной модели»

Проверил:

Огородник Р.В.

Выполнил:

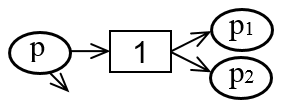
студент группы 05050x

ololo

Минск 2013

**Задание 1**.

Построить граф состояний Q-схемы.

j = {0, 1} – кол-во заявок в очереди

t1 = {0, 1} – кол-во заявок в канале №1

t2 = {0, 1} – кол-во заявок в канале №2

Общий вид кодировки состояния системы:

{j, t1, t2}



По графу построим аналитическую модель и, решив ее, определим вероятности состояний.













Решив систему уравнений (при **p**=0,5 **p1**=0,6 **p2**=0,8), получили:

P000 = 0,133

P010 = 0,231

P011 = 0,308

P001 = 0,082

P111 = 0,246

Исходя из полученных данных, рассчитаем теоретические значения вероятности отказа (Pотк) и коэффициентов загрузки каналов (Kкан):

Pотк = P111\*(1-p)\*p1\*p2 = 0.0590487

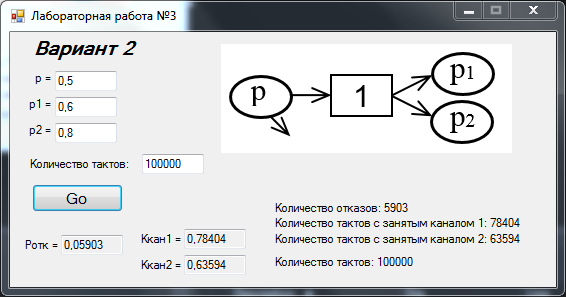
Kкан1 = P010 + P011 + P111 = 0.7845817

Kкан2 = P011 + P001 + P111 = 0.6355932

**Задание 2**

Для СМО из задания 1 построить имитационную модель и исследовать ее (разработать алгоритм и написать имитирующую программу, предусматривающую сбор и статистическую обработку данных для получения оценок заданных характеристик СМО). Распределение интервалов времени между заявками во входном потоке и интервалов времени обслуживания – геометрическое с соответствующим параметром (p, p1, p2).

**Результат работы программы:**



**Вывод:**

В ходе лабораторной работы была аналитически смоделирована дискретно- стохастическая СМО и разработана программа, имитирующая поведение данной СМО. Построенная модель позволяет статистически подсчитать характеристики СМО. Статистическое значение искомой характеристики оказывается близким к теоретически рассчитанному. Значит имитационная модель построена верно. Было также замечено, что на выходные данные влияют параметры СМО, такие как **p**, **p1**, **p2**.

**Листинг программы:**

class QueuingSystem

{

private readonly Random \_rand, \_rand1, \_rand2;

private readonly double \_p, \_p1, \_p2;

public byte J { get; private set; } // Состояние очереди {0, 1}

public byte T1 { get; private set; } // Состояние канала 1 {0, 1}

public byte T2 { get; private set; } // Состояние канала 2 {0, 1}

public int DenialCounter { get; private set; } // Счётчик отказов

public int Counter1 { get; private set; } // Счётчик занятости канала 1

public int Counter2 { get; private set; } // Счётчик занятости канала 2

public QueuingSystem(double p, double p1, double p2)

{

J = 0;

T1 = 0;

T2 = 0;

DenialCounter = 0;

Counter1 = 0;

Counter2 = 0;

\_rand = new Random();

\_rand1 = new Random(101);

\_rand2= new Random(4242440);

\_p = p; // Вероятность не выдачи заявки

\_p1 = p1; // Вероятность не обслуживания заявки каналом 1

\_p2 = p2; // Вероятность не обслуживания заявки каналом 2

}

// Была ли не выдана заявка

private bool IsNoRequest()

{

return \_rand.NextDouble() < \_p;

}

// Была ли не обслужена заявка каналом 1

private bool IsNoService1()

{

return \_rand1.NextDouble() < \_p1;

}

// Была ли не обслужена заявка каналом 2

private bool IsNoService2()

{

return \_rand2.NextDouble() < \_p2;

}

public void GenerateNextState()

{

if (T1 == 1) // Если в канале 1 есть заявка

{

if (IsNoService1()) Counter1++;

else T1 = 0;

}

if (T2 == 1) // Если в канале 2 есть заявка

{

if (IsNoService2()) Counter2++;

else T2 = 0;

}

if(J == 1) // Если есть заявка в очереди

{

if (T1 == 0)

{

J = 0;

T1 = 1;

Counter1++;

}

else if (T2 == 0)

{

J = 0;

T2 = 1;

Counter2++;

}

}

if (IsNoRequest()) // Если источник не выдал заявку

return;

if (J == 0) // Если нет заявок в очереди

{

if (T1 == 0)

{

T1 = 1;

Counter1++;

}

else if (T2 == 0)

{

T2 = 1;

Counter2++;

}

else J = 1;

}

else // Если есть заявка в очереди

{

DenialCounter++; // Увеличиваем счётчик отказов

return;

}

}

}

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

{

QueuingSystem smo = new QueuingSystem(double.Parse(textBox\_p.Text), double.Parse(textBox\_p1.Text), double.Parse(textBox\_p2.Text));

int ticksCount = int.Parse(textBox\_ticks.Text);

for (int i = 0; i < ticksCount - 1; i++)

smo.GenerateNextState();

// Pотк - вероятность отказа

textBox\_denial.Text = (smo.DenialCounter / (double)ticksCount).ToString();

// Коэффициент загрузки канала 1

textBox\_k1.Text = (smo.Counter1/(double) ticksCount).ToString();

// Коэффициент загрузки канала 2

textBox\_k2.Text = (smo.Counter2 /(double)ticksCount).ToString();

label\_denial.Text = "Количество отказов: " + smo.DenialCounter.ToString();

label\_k1.Text = "Количество тактов с занятым каналом 1: " + smo.Counter1.ToString();

label\_k2.Text = "Количество тактов с занятым каналом 2: " + smo.Counter2.ToString();

label\_ticks.Text = "Количество тактов: " + ticksCount.ToString();

}