**Содержание**

1. Вступление………………………………………………………….2
2. Основные сведения о задаче и метод её решения………………..3
   1. Перечень сокращений, символов и специальных терминов
   2. Описание задачи……………………………………………...4
      1. Математическое описание задачи………………………..4
      2. Метод решения……………………………………….........5
      3. Вывод основных формул…………………………………7
3. Детальное описание программы………………………………......8
   1. Принцип организации программы…………………………..9
4. Результаты испытания программы………………………………..9
5. Результаты исследования и их анализ…………………………….10
6. Выводы……………………………………………………………...10
7. Листинг программы………………………………………………11
8. **Вступление**

Теория массового обслуживания опирается на теорию вероятностей и математическую статистику.

На первичное развитие теории массового обслуживания оказали особое влияние работы датского ученого А.К. Эрланга (1878-1929).

**Теория массового обслуживания** – область прикладной математики, занимающаяся анализом процессов в системах производства, обслуживания, управления, в которых однородные события повторяются многократно, например, на предприятиях бытового обслуживания; в системах приема, переработки и передачи информации; автоматических линиях производства и др.

**Предметом** теории массового обслуживания является установление зависимостей между характером потока заявок, числом каналов обслуживан6ия, производительностью отдельного канала и эффективным обслуживанием с целью нахождения наилучших путей управления этими процессами.

**Задача теории массового обслуживания** – установить зависимость результирующих показателей работы Системы Массового Обслуживания (СМО) (вероятности того, что заявка будет обслужена; математического ожидания числа обслуженных заявок и т.д.) от входных показателей (количества каналов в системе, параметров входящего потока заявок и т.д.). Результирующими показателями или интересующими нас характеристиками СМО являются – показатели эффективности СМО, которые описывают, способна ли данная система справляться с потоком заявок. Примерами СМО могут служить: телефонные станции, ремонтные мастерские, билетные кассы, стоянки такси, парикмахерские и т.п.

Задачи теории массового обслуживания носят оптимизационный характер и в конечном итоге включают экономический аспект по определению такого варианта системы, при котором будет обеспечен минимум суммарных затрат от ожидания обслуживания, потерь времени и ресурсов на обслуживание и простоев каналов обслуживания.

Для проверки той или иной системы на эффективность и целесообразность, производят теоретические расчеты. Часто возникает необходимость проверки полученных теоретических результатов, для чего лучшим средством служит модель. Не маловажный тот факт, что не всегда можно исследовать систему на практике, в связи с ее громоздкостью или дороговизной. В этом случае также целесообразно применять модель.

В данной работе предлагается к рассмотрению модель системы массового обслуживания с отказами. Эта система получила широкое распространение в телефонных сетях.

Задачу массового обслуживания, условно, можно разделить на две задачи: задачу анализа и задачу синтеза.

Задача анализа позволяет, на основе исходных, неизменных данных, произвести оценку эффективности заданной системы.

В то время как целью задачи синтеза служит выбор оптимальных параметров системы, при которых будет достигаться наиболее оптимальный результат. Для этого строиться компьютерная модель системы, в которой производится определенное количество экспериментов с целью выявления оптимального решения.

1. **Основные сведения о задаче и метод её решения**
   1. **Перечень сокращений, символов и специальных терминов**

***n –*** количество каналов обслуживания;

**λ –** интенсивность входящего потока заявок;

**μ** – интенсивность обслуживания канала;

**ε** – точность;

**Р0, Р1, Р2, Р3, Р4** – вероятности состояния системы;

**tср.пост.** – среднее время поступления заявки в систему;

**tср.обсл.** – среднее время обслуживания заявки в системе;

**ρ** – коэффициент загрузки системы;

**q** – относительная пропускная способность системы;

**А** – абсолютная пропускная способность системы;

**k** – среднее число занятых каналов в системе;

**СМО** – система массового обслуживания.

* 1. **Описание задачи**

Постановка задачи:

В СМО, состоящей из **n** каналов обслуживания, поступает простейший (пуассоновский) поток заявок с интенсивностью **λ.** Интенсивность обслуживания канала - **μ**. Необходимо теоретически и практически рассчитать вероятности состояний системы, вероятность отказа, среднее время между поступлением заявок, среднее время обслуживания заявки в канале, коэффициент загрузки системы (приведенная интенсивность), относительную пропускную способность системы, абсолютную пропускную способность системы, среднее число занятых каналов.

Теоретический расчет подразумевает под собой вычисление выше приведенных величин, согласно формул для случая, когда распределение времени поступления заявок подчиняется пуассоновскому закону распределения. Необходимые формулы будут рассмотрены ниже.

В практической части необходимо построить модель системы, при условии, что время поступления заявок подчиняется пуассоновскому закону распределения, и рассчитать выше описанные величины.

Следует отметить, что теоретический расчет параметров системы для случая, когда потоки событий, переводящие систему из состояния в состояние, распределенные по закону отличному от пуассоновского, довольно сложен, т.к. на практике такой вариант встречается очень редко. Поэтому аналитические формулы для характеристик СМО удается получить только для самых простых случаев.

* + 1. **Математическое описание задачи**

Параметры системы:

Тип СМО: СМО с отказами

Входящий поток: пуассоновский

Интенсивность входящего потока заявок: **λ** = 18 заявок в единицу времени

Интенсивность обслуживания канала: **μ** = 5 заявок в единицу времени

Количество каналов обслуживания: **n** = 4 канала

Точность: **ε** = 0,01

* + 1. **Метод решения**

**Простейшим потоком** называется поток, обладающий следующими свойствами:

1. стационарность;

2. ординарность;

3. отсутствие последействия.

**Поток событий** называется **стационарным**, если вероятность попадания того или иного числа событий на участок времени длиной τ зависит только от длины участка и не зависит от того, где именно на оси времени расположен этот участок.

**Поток событий** называется **ординарным,** если вероятность попадания на элементарный участок Δt двух или более событий пренебрежимо мала по сравнению с вероятностью попадания одного события. Ординарность означает, что поток прореженный, т.е. между любыми двумя событиями есть временной интервал.

**Поток событий** называется **потоком без последействия**, если для любых, не перекрывающихся участков времени число событий, попадающих на один из них, не зависит от числа событий, попадающих на другие. Это означает, что заявки попадают в систему не зависимо друг от друга.

Интенсивность поступления потока заявок - **λ**. Простейшие потоки поступления заявок характеризуются показательным законом распределения. Тогда интервал времени поступления потока заявок представляет собой случайную величину с одним и тем же распределением вероятностей F (t).

, (1) где λ>0 – постоянная.

Плотность распределения показательного закона задается формулой:



где λ>0 - интенсивность поступления заявок.

Необходимо учесть, что данная система является СМО с отказами – это означает, что в момент, когда система занята (все каналы заняты), то пришедшая заявка получает отказ и покидает систему. Если в системе свободен хотя бы один канал обслуживания, то пришедшая заявка немедленно поступает на обслуживание в этот канал.

Длительность обслуживания потока заявок также представляет собой случайную величину, подчиняющуюся показательному закону распределения. Интенсивность обслуживания потока заявок - **μ**. Длительность обслуживания потока заявок представляет собой случайную величину с одним и тем же распределением вероятностей F (t).

, (1) где μ>0 – постоянная.

Плотность распределения показательного закона задается формулой:



где μ>0, - интенсивность обслуживания потока заявок.

Изобразим граф состояний заданной СМО:

**S0**

**S1**

**S2**

**S3**

**S4**

**λ λ λ λ**

**μ 2 μ 3 μ 4 μ**

в котором:

**S0** – все каналы свободны (система свободна);

**S1** – занят один канал;

**S2** – занято два канала;

**S3** – занято три канала;

**S4** – все каналы заняты (система занята).

Из свободного состояния в состояние обслуживания, систему переводит поток заявок с интенсивностью **λ.** Тогда как освобождение каналов от обслуживания происходит по такому принципу:

Из состояния **Sn** в состояние **Sn-1** , систему переводит интенсивность потока **n\*μ**.

Если заявка поступает в момент времени, когда система свободна или свободен хотя бы один канал обслуживания, то она немедленно поступает на обслуживание. Если же все каналы заняты, то заявка покидает систему.

* + 1. **Вывод основных формул**

Среднее время поступления заявок в систему:

tпост. = 1/λ

Данный параметр является интервалом времени, из которого генерируется появление заявки.

Среднее время (интервал) обслуживания одной заявки одним каналом:

tобс. = 1/μ

Для того чтобы узнать какое количество заявок поступит в систему за время обслуживания одной заявки, необходимо:

ρ = λ/μ

ρ – приведенная интенсивность или коэффициент загрузки системы.

Составим уравнения Колмогорова для данной системы:

,

,







Уравнения называться уравнениями Эрланга, в которых начальными условиями, для их решения, являются:

р0(0)=1;

р1(0)= р2(0)= р3(0)=…= рn(0)=0,

т.е. в начальный момент времени – система свободна.

Рассмотрим предельные вероятности состояний P0, P1, P2,…, Pn, которые характеризуют установившийся режим работы СМО при t - > ∞.

Р1= \* P0;

Р2= \* P0;

Р3= \* P0;

Р4= \* P0;

Р0 =

Зная все вероятности системы, можно найти характеристики СМО:

Вероятность отказа:

Ротк = Р4 – т.е. все каналы заняты

Относительная пропускная способность:

q = 1 – Ротк

Абсолютная пропускная способность:

А = λ\*q

Важной характеристикой СМО с отказами, является среднее число занятых каналов (в данном случае оно совпадает со средним числом заявок, находящихся в системе):

k = 0\*Р0+1\*Р1+2\*Р2+3\*Р3+4\*Р4 ,

или

k = А/μ

1. **Детальное описание программы**

Данная программа написана в среде разработки Visual C# с использованием визуальных компонентов управления.

Результирующая программа СМО.exe является полностью законченным программным продуктом. Для ее нормальной работы необходимы следующие условия:

- IBM совместимый компьютер с процессором старше 386 серии;

- Операционная система Windows ‘98 и выше;

-Монитор класса VGA;

* 1. **Принцип организации программы**

Данная программа является универсальной для данного типа задач.

Для удобства управления, введения данных, просмотра результатов, присутствует графическая часть (формы, кнопки, поля ввода, поля вывода результатов и т.п.).

Работа программы начинается с формы введения данных.

После того как данные введены, необходимо нажать на кнопку «Пуск», на форме программы, для запуска работы основной части программы.

Для просмотра отчета необходимо нажать на кнопку «Отчет», после чего появиться форма результатов работы программы с рассчитанными характеристиками СМО.

Основные переменные в программе, являются глобальными переменными. После того как пользователь ввел все необходимые данные в основном окне программы, переменным присваиваются введенные характеристики СМО.

1. **Результаты испытания программы**

Входящие данные:

Интенсивность входящего потока заявок: **λ** = 18 заявок в единицу времени

Интенсивность обслуживания канала: **μ** = 5 заявок в единицу времени

Количество каналов обслуживания: **n** = 4 канала

Точность: **ε** = 0,01

* 1. **Результат расчета тестовой задачи**

|  |  |  |
| --- | --- | --- |
|  | **Аналитический расчет** | **Программа** |
| **ρ** (коэф. нагрузки) | 3,6 | 4,5 |
| **Ротказа** | 27 % | 35,96 % |
| **q** (отн. пропускная способн.) | 0,73 | 0,64 |
| **A** (абс. пропускная способн.) | 13,14 | 10,89 |
| **k** (ср.число занятых каналов) | 2,63 | 2,9 |

Полученные результаты отличаются, т.к. аналитический расчет проводился с константными данными без имитационного процесса, тогда как в процессе работы программы данные были динамическими, т.к. создавалась имитация работы системы.

1. **Результаты исследования и их анализ**

После ознакомления с программой и инструкцией пользователя, можно приступить непосредственно к проведению экспериментов и исследований. Их целью может служить, например, сравнение параметров системы при распределении Пуассона и равномерном распределении, можно определить при каких значениях интенсивностей, числе каналов и мест в очереди, параметры будут приблизительно равны. Мною замечено, что они тем меньше отличаются, чем интенсивность обслуживания больше интенсивности поступления заявок. Также возможно исследование загруженности каналов с целью поиска оптимального числа каналов и мест в очереди для дальнейшего применения в сфере обслуживания.

1. **Выводы**

Таким образом, поставленная задача о создании программы, позволяющей смоделировать многоканальную систему массового обслуживания с ограниченной длиной очереди, решена. Созданная программа позволяет произвести расчет таких параметров системы как:

* вероятности состояний **Pi**;
* вероятность  **Ps**
* среднее число заявок в системе **Ns**
* среднее число заявок в очереди **Qоч.**

при заданных начальных условиях :

* число каналов обслуживания **n** ;
* интенсивность поступления заявок **λ ;**
* интенсивность обслуживания **μ;**
* параметр распределения случайного времени ожидания;
* визуально проследить работу системы и сравнить теоретические и практические результаты расчетов.

1. **Приложение 1**

**Листинг программы**

**WorkForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace CMO

{

public partial class WorkForm : Form

{

private int InOrders;

private int ServiceOrders;

private int NumOfChannels;

private double Precision = 0;

private double OrderInterval;

private double ChannelInterval;

private int TotalTime;

private double CmoTime;

private ulong NumOfOrders = 0;

private ulong ServedOrders;

private int CmoPosition;

private int[] ChannelBusy;

private double[] ServiceTimeOfChannel;

private Random rand = new Random();

// функция которая преобразовывает точность для дальнейшего использования в программе и функциях

private int Precition(double e)

{

int Tochnost = 0;

if (e == 0.1)

Tochnost = 10;

if (e == 0.01)

Tochnost = 100;

if (e == 0.001)

Tochnost = 1000;

return Tochnost;

}

// Число знаков для округления исходя из точности

private int ChisloZnakov(double Prec)

{

int a = 0;

if (Prec == 0.1)

a = 1;

if (Prec == 0.01)

a = 2;

if (Prec == 0.001)

a = 3;

return a;

}

// функция которая возвращает интервал времени между заявками, либо интер. времени обслуж. канала

private double OrdersInterval(int NumOfOrders, double Tochnost)

{

int CelayaChast;

double Ostatok;

CelayaChast = Precition(Tochnost) / NumOfOrders;

Ostatok = (Precition(Tochnost) / (double)NumOfOrders - CelayaChast) \* 10;

if (Ostatok >= 5)

CelayaChast++;

return (double)(CelayaChast / (double)Precition(Tochnost));

}

// генератор моментов времени

private double TimeGenerator(bool FirstOrder, double Interval, double Tochnost)

{

double NewTime;

int GenerationInterval = (int)(Interval \* Precition(Tochnost) + 1);

// только если это не первая заявка

if (!FirstOrder)

{

// исключение генерации нуля => повторяющееся время

do

{

NewTime = (double)(rand.Next() % GenerationInterval) / Precition(Tochnost);

} while (NewTime == 0);

}

else

NewTime = (double)(rand.Next() % GenerationInterval) / Precition(Tochnost);

return Math.Round(NewTime,ChisloZnakov(Tochnost));

}

// проверяет наличие новой заявки

private bool NewOrder(double TimeOfCmo, double IntervalOfOrder, ulong NumberOfOrders)

{

if (TimeOfCmo >= IntervalOfOrder \* (NumberOfOrders))

{

return true;

}

else return false;

}

// проверка системы в текущий момент времени

private void CmoCheck(int TotalTime, double CmoTime, int NumOfChannels)

{

int i = 0;

if (CmoPosition != 0)

{

while (i < NumOfChannels)

{

if (ChannelBusy[i] == 1)

{

if (((ServiceTimeOfChannel[i] <= CmoTime) && (CmoTime <= TotalTime)) || ((CmoTime > TotalTime) && (ServiceTimeOfChannel[i] <= TotalTime)))

{

ChannelBusy[i] = 0;

ServiceTimeOfChannel[i] = 0;

CmoPosition--;

richTextBox1.Text += (i + 1).ToString() + "-й канал обслужил заявку" + "\n";

ServedOrders++;

}

}

i++;

}

}

}

// запись заявок в каналы и генерация времени обслуживания в канале + окончание обслуживания

private void CmoWork(double CmoTime, int NumOfChannels, double ChannelInterval, double e)

{

double ChannelWorkTime;

int i = 0;

// если система свободна, то запись осуществляется в первый канал

if (CmoPosition == 0)

{

ChannelBusy[0] = 1;

ChannelWorkTime = TimeGenerator(false, ChannelInterval, e);

ServiceTimeOfChannel[0] = CmoTime + ChannelWorkTime;

CmoPosition++;

InfoOrder(CmoTime, 0, ChannelWorkTime);

}

else // иначе, идет запись в первый свободный канал

{

if (CmoPosition != NumOfChannels)

{

while (i < NumOfChannels)

{

if (ChannelBusy[i] == 0)

{

ChannelBusy[i] = 1;

ChannelWorkTime = TimeGenerator(false, ChannelInterval, e);

ServiceTimeOfChannel[i] = CmoTime + ChannelWorkTime;

CmoPosition++;

InfoOrder(CmoTime, i, ChannelWorkTime);

break;

}

i++;

}

}

else

{

richTextBox1.Text += "Заявка не может быть принята на обслуживание - КАНАЛЫ ЗАНЯТЫ" + "\n";

richTextBox1.Text += "Состояние СМО: " + CmoPosition.ToString() + "\n";

for (i = 0; i < NumOfChannels; i++)

{

richTextBox1.Text += "Состояние " + (i + 1).ToString() + "-го канала : " + Convert.ToString(ChannelBusy[i]) + "\n";

richTextBox1.Text += "Время окончания обслуживания в " + (i + 1).ToString() + " канале: " + Convert.ToString(ServiceTimeOfChannel[i]) + "\n";

}

}

}

}

// информация о поступившей заявке

private void Info(double CmoTime, ulong NumOfOrders)

{

richTextBox1.Text += "Время поступления " + NumOfOrders.ToString() + "-й заявки: " + CmoTime.ToString() + "\n";

}

private void InfoOrder(double CmoTime, int Index, double TimeOfChannel)

{

richTextBox1.Text += "Заявка принята на обслуживание: " + CmoTime.ToString() + " в " + (Index + 1).ToString() + "-й канал" + "\n";

richTextBox1.Text += "Время обслуживания заявки в канале: " + TimeOfChannel.ToString() + "\n";

richTextBox1.Text += "Время окончания обслуживания: " + ServiceTimeOfChannel[Index].ToString() + "\n";

}

// конструктор формы

public WorkForm()

{

InitializeComponent();

}

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

{

Application.Exit();

}

// ПУСК

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

{

// Считывание данных

if (TBoxOrders.Text != "" && TBoxOrderService.Text != "" && TBoxChannels.Text != "" && comboBoxPrecision.Text != "" && TBoxCmoWork.Text != "")

{

this.Enabled = false;

InOrders = Convert.ToInt32(TBoxOrders.Text);

ServiceOrders = Convert.ToInt32(TBoxOrderService.Text);

NumOfChannels = Convert.ToInt32(TBoxChannels.Text);

Precision = Convert.ToDouble(comboBoxPrecision.SelectedItem.ToString());

richTextBox1.Text = "";

int i = 0;

// установление размерности массивов

ChannelBusy = new int[NumOfChannels];

ServiceTimeOfChannel = new double[NumOfChannels];

TotalTime = Convert.ToInt32(TBoxCmoWork.Text);

// Перевод системы в первоначальное состояние (обнуление всех переменных)

CmoTime = 0;

CmoPosition = 0;

NumOfOrders = 0;

ServedOrders = 0;

while (i <= NumOfChannels - 1)

{

ChannelBusy[i] = 0;

ServiceTimeOfChannel[i] = 0;

i++;

}

// Генерация поступления первой заявки

CmoTime = TimeGenerator(true, OrderInterval, Precision);

richTextBox1.Text += "\n" + "-----------------------" + "\n";

richTextBox1.Text += "Текущее время СМО :" + CmoTime.ToString();

richTextBox1.Text += "\n" + "-----------------------" + "\n";

NumOfOrders++;

Info(CmoTime, NumOfOrders);

CmoCheck(TotalTime, CmoTime, NumOfChannels);

CmoWork(CmoTime, NumOfChannels, ChannelInterval, Precision);

// Последующая работа системы

do

{

// Генерация следующих моментов времени СМО

CmoTime = Math.Round((CmoTime + TimeGenerator(false, OrderInterval, Precision)),ChisloZnakov(Precision));

richTextBox1.Text += "\n" + "-----------------------" + "\n";

richTextBox1.Text += "Текущее время СМО :" + CmoTime.ToString();

richTextBox1.Text += "\n" + "-----------------------" + "\n";

// Проверка системы в текущий момент времени

CmoCheck(TotalTime, CmoTime, NumOfChannels);

// Появление новой заявки

if ((NewOrder(CmoTime, OrderInterval, NumOfOrders)) && (CmoTime <= TotalTime))

{

NumOfOrders++;

Info(CmoTime, NumOfOrders);

CmoWork(CmoTime, NumOfChannels, ChannelInterval, Precision);

}

} while (CmoTime < TotalTime);

richTextBox1.Text += "Количество поступивших заявок: " + NumOfOrders.ToString() + "\n";

richTextBox1.Text += "Количество обслуженных заявок: " + ServedOrders.ToString() + "\n";

this.Enabled = true;

this.Focus();

btnResult.Enabled = true;

}

else MessageBox.Show("Все поля обязательны для заполнения!");

}

// проверка вводимых значений

private bool Check(string Txt)

{

try

{

Convert.ToInt32(Txt);

}

catch (Exception)

{

MessageBox.Show("Вводимое значение должно быть целым числом");

return false;

}

return true;

}

// ввод входящего потока

private void TBoxOrders\_TextChanged(object sender, EventArgs e)

{

TBoxOrderService.Enabled = true;

}

// ввод интенсивности обслуживания

private void TBoxOrderService\_TextChanged(object sender, EventArgs e)

{

TBoxChannels.Enabled = true;

}

// ввод количества каналов

private void TBoxChannels\_TextChanged(object sender, EventArgs e)

{

TBoxCmoWork.Enabled = true;

}

// выбор точности

private void comboBoxPrecision\_TextChanged(object sender, EventArgs e)

{

Precision = Convert.ToDouble(comboBoxPrecision.SelectedItem.ToString());

TBoxOrders.Enabled = true;

}

// завершение ввода входящего потока

private void TBoxOrders\_Leave(object sender, EventArgs e)

{

if (TBoxOrders.Text != ""&&(btnExit.ContainsFocus==false))

{

if (Check(TBoxOrders.Text))

{

InOrders = Convert.ToInt32(TBoxOrders.Text);

OrderInterval = OrdersInterval(InOrders, Precision);

labelOrdersInterval.Text = OrderInterval.ToString()+" ед.вр.";

}

else TBoxOrders.Focus();

}

}

// завершение ввода интенсивности обслуживания

private void TBoxOrderService\_Leave(object sender, EventArgs e)

{

if (TBoxOrderService.Text != "" && (btnExit.ContainsFocus == false))

{

if (Check(TBoxOrderService.Text))

{

ServiceOrders = Convert.ToInt32(TBoxOrderService.Text);

ChannelInterval = OrdersInterval(ServiceOrders, Precision);

labelChannelInterval.Text = ChannelInterval.ToString() + " ед.вр.";

}

else TBoxOrderService.Focus();

}

}

// завершение ввода количества каналов

private void TBoxChannels\_Leave(object sender, EventArgs e)

{

if (TBoxChannels.Text != "" && (btnExit.ContainsFocus == false))

{

if (Check(TBoxChannels.Text))

{

NumOfChannels = Convert.ToInt32(TBoxChannels.Text);

}

else TBoxChannels.Focus();

}

}

// ОТЧЕТ

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

{

ResultForm FResult = new ResultForm(this,NumOfChannels,Precision,TotalTime,NumOfOrders,ServedOrders);

FResult.Show();

}

private void WorkForm\_Load(object sender, EventArgs e)

{

btnResult.Enabled = false;

}

}

}

**ResultForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace CMO

{

public partial class ResultForm : Form

{

private Form MForm;

private double inOrders;

private double ordersInService;

private int numOfChannels;

private double precision;

private double ordersInterval;

private double channelsInterval;

private int workTime;

private ulong totalInOrders;

private ulong servedOrders;

//------------------------

private double koefNagruzki;

private double otkaz;

private double otnSposob;

private double absolSposob;

private double zanyatKanaly;

private double Po;

//------------------------

public ResultForm(Form MainF, int NumChan,double Prec, int TotTime,ulong TotOrd,ulong ServOrd)

{

this.MForm = MainF;

numOfChannels = NumChan;

precision = Prec;

workTime = TotTime;

totalInOrders = TotOrd;

servedOrders = ServOrd;

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

MForm.Enabled = true;

ResultForm.ActiveForm.Close();

}

private void ResultForm\_Load(object sender, EventArgs e)

{

int i = 0;

double[] P=new double[numOfChannels+1];

dataGridView1.ColumnCount=numOfChannels+1;

string[] row = new string[numOfChannels + 1];

MForm.Enabled = false;

// Расчеты

// Расчет ср. числа потупивших заявок в ед. времени (Лямда)

inOrders = totalInOrders / (double)workTime;

// Расчет ср. числа обслужваемых заявок одним каналом (Мю)

ordersInService = (servedOrders / (double)workTime) / (double)numOfChannels;

// Расчет ср. интервала времени между поступл. заявок

ordersInterval = 1 / inOrders;

// Расчет срю интервала времени обслуживания канала

channelsInterval = 1 / ordersInService;

// Коеффициент загрузки

koefNagruzki = inOrders / ordersInService;

// Р0

Po = 1 / (1 + Sum(koefNagruzki, numOfChannels));

// Вероятность отказа

otkaz = ((Stepen(koefNagruzki, numOfChannels) / Factor(numOfChannels)) \* Po);

// Относительная пропускная способность

otnSposob = 1 - otkaz;

// Абсолютная пропускная способность

absolSposob = (totalInOrders / (double)workTime) \* otnSposob;

// Число занятых каналов

zanyatKanaly = absolSposob / (double)((servedOrders / (double)workTime) / (double)numOfChannels);

// Вывод на форму

labelInOrders.Text = Math.Round(inOrders,ChisloZnakov(precision)).ToString()+" заявок в ед.вр.";

labelServiceOrders.Text = Math.Round(ordersInService, ChisloZnakov(precision)).ToString() + " заявок в ед.вр.";

labelNumOfChannels.Text = numOfChannels.ToString();

labelPrecision.Text = precision.ToString();

labelOrdersInterval.Text = Math.Round(ordersInterval, ChisloZnakov(precision)).ToString() + " ед.вр.";

labelChannelsInterval.Text = Math.Round(channelsInterval, ChisloZnakov(precision)).ToString() + " ед.вр.";

labelWorkTime.Text = workTime.ToString() + " ед.вр.";

labelTotalOrders.Text = totalInOrders.ToString() + " заявок";

labelServedOrders.Text = servedOrders.ToString() + " заявок";

labelZagruzka.Text = Math.Round(koefNagruzki, ChisloZnakov(precision)).ToString() + " заявок";

labelOtkaz.Text = (Math.Round(otkaz \* 100, ChisloZnakov(precision))).ToString() + " %";

labelOtnPropusk.Text = (Math.Round(otnSposob \* 100, ChisloZnakov(precision))).ToString() + " %";

labelAbsPropusk.Text = (Math.Round(absolSposob, ChisloZnakov(precision))).ToString() + " заявок";

labelZanChannels.Text = (Math.Round(zanyatKanaly,ChisloZnakov(precision))).ToString();

// Вероятности состояния системы

while (i <= numOfChannels)

{

if (i == 0)

{

P[i] = Po;

dataGridView1.Columns[i].Name = "P" + i.ToString();

row[i] = Math.Round(P[i] \* 100, ChisloZnakov(precision)).ToString() + " %";

}

else

{

P[i] = (Stepen(koefNagruzki, i) / Factor(i)) \* Po;

dataGridView1.Columns[i].Name = "P" + i.ToString();

row[i] = Math.Round(P[i] \* 100, ChisloZnakov(precision)).ToString() + " %";

}

i++;

}

dataGridView1.Rows.Add(row);

}

// Сумма в знаменателе для расчета Ро

private double Sum(double ro, int numChan)

{

double summa = 0;

int n=1;

while (n <= numChan)

{

summa = summa + Stepen(ro,n) / (double)Factor(n);

n++;

}

return summa;

}

// Факториал

private ulong Factor(int num)

{

ulong factor = 1;

int n=0;

while (n <= num)

{

if ((n == 0) || (n == 1))

return 1;

else

factor = factor \* (ulong)n;

n++;

}

return factor;

}

// Возведение в степень

private double Stepen(double X, int step)

{

int n=1;

double result=X;

if (X == 0)

return 0;

if (X == 1)

return 1;

if (step == 1)

return X;

if (step == 0)

return 1;

while (n <= step)

{

if (n>1)

result = result \* X;

n++;

}

return result;

}

// Число знаков после запятой для округления

private int ChisloZnakov(double Prec)

{

int a=0;

if (Prec == 0.1)

a = 1;

if (Prec == 0.01)

a = 2;

if (Prec == 0.001)

a = 3;

return a;

}

}

}