**МИНОБРНАУКИ РОССИИ**

**«Юго-Западный государственный университет»**

**Кафедра «Вычислительная техника»**

**КУРСОВАЯ РАБОТА**

по дисциплине «Технология программирования»

на тему «Программа имитационного моделирования системы массового обслуживания»

Направление подготовки 09.03.02 Информационные системы и технологии

Автор работы Романов Г.А\_ , .

фамилия, инициалы подпись, дата

Группа ИС-91б

Руководитель проекта Панищев В.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, инициалы подпись, дата

Проект защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_.

подпись, дата фамилия, инициалы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_.

подпись, дата фамилия, инициалы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_.

подпись, дата фамилия, инициалы

Курск 2021 г.

**МИНОБРНАУКИ РОССИИ**

**«Юго-Западный государственный университет»**

Кафедра «Вычислительная техника»

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студента Романова Георгий Александровича шифр 19-06-0146 группа ИС-91б

1.Тема «Программа имитационного моделирования системы массового обслуживания»  
2. Срок представления проекта к защите «14» мая 2021 г.

3. Исходные данные для проектирования.

Разработать программу для моделирования автоматизированной справочно-информационной системы.

4. Содержание пояснительной записки курсового проекта:

4.1. Введение

4.2. Описание модели  
4.3. Разработка Q-схемы

4.4. Разработка иерархии классов

4.5. Разработка руководства пользователя

4.6. Заключение

4.7. Список использованной литературы

4.8. Приложения

Руководитель работы (проекта) 19.02.21 В.С. Панищев

подпись, дата инициалы, фамилия

Задание принял к исполнению , 19.02.21 Романов Г.А

подпись, дата инициалы, фамилия

**РЕФЕРАТ**

Курсовая работа: «Моделирование систем массового обслуживания» состоит из пяти глав, введения и заключения общим объемом 33 страницы. В работе использовано 10 источников литературы.

Объектом исследования является процесс моделирование систем массового обслуживания.

Предметом исследования в работе являются методы и инструментальные среды разработки, отладки и тестирования программного продукта.

Целью курсовой работы является разработка программного средства для моделирования автоматизированной справочно-информационной системы. Задачами курсовой работы являются:

1. Анализ подходов к реализации процесса моделирования;
2. Выбор инструментальной среды и разработка алгоритма моделирования;
3. Анализ полученных результатов разработанного приложения.

В работе дано описание создания программы для моделирования автоматизированной справочно-информационной системы, позволяющей смоделировать работу автоматизированной справочно-информационной системы. Разработано приложения для моделирования.

**Содержание**

[Введение 6](#_Toc70865263)

[1 Описание модели 7](#_Toc70865264)

[2 Разработка q-схемы 8](#_Toc70865265)

[3 Разработка иерархии классов 11](#_Toc70865266)

[4 Разработка руководства пользователя 15](#_Toc70865267)

[Заключение 19](#_Toc70865268)

[Список использованных источников 20](#_Toc70865269)

[Листинг программы 21](#_Toc70865270)

ПЕРЕЧЕНЬ УСЛОВНЫХ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ОП – основная память.

НМД – накопитель на магнитных дисках

СМО – система массового обслуживания

# **Введение**

Данная курсовая работа имеет целью разработку прикладной программы, позволяющей пользователю смоделировать работу автоматизированной справочно-информационной системы.

Во многих областях производства, бытового обслуживания, экономики и финансов важную роль играют системы специального вида, реализующие многократное выполнение однотипных задач [1]. Подобные системы называют системами массового обслуживания (СМО). Каждая СМО предназначена для обслуживания (выполнения) некоторого потока заявок (или требований), поступающих на вход системы большей частью не регулярно, а в случайные моменты времени [2]. Обслуживание заявок, в общем случае, также длится не постоянное, заранее известное, а случайное время. После обслуживания запроса канал освобождается и готов к приему следующей запроса [3].

Цель работы – разработка прикладной программы для моделирования работы автоматизированной справочно-информационной системы.

Для достижения поставленной цели следует решить следующие задачи:

1. изучить и описать среду программирования;
2. разработать приложение для моделирования системы.
3. проанализировать работу разработанного приложения.

# **Описание модели**

Автоматизированная справочно-информационная система содержит 3 терминала, процессор, основную память (ОП) и два накопителя на магнитных дисках (НМД). Имеется пять типов запросов, которые, как и ответы на них, поступают на терминалы 1, 2 и 3. Терминалы 1 и 2 обрабатывают сообщения, относящиеся к первым четырем типам запросов, а терминал 3 — сообщения, относящиеся к запросам пятого типа. Запросы типов 1, 2 и 5 характеризуются равномерным распределением интервалов между моментами поступления: 1 — 600±10 с; 2 — 720±20 с; 5 — 660±30 с, а запросы 3 и 4 являются пуассоновскими со средними значениями интервалов: 3 — 480 с; 4 — 540 с. Запросы через ОП и процессов записываются в НМД 1 за среднее время 20 с. Существует 5 прикладных программ, хранящихся в НМД 2, каждая из которых обрабатывает сообщения, относящиеся к запросу определенного типа. При поступлении в ОП запроса с НМД 1 обрабатывающая этот запрос программа загружается в ОП, где одновременно могут находиться до 5 сообщений, прочитанных с терминалов, и до 8 сообщений, прочитанных с НМД 1. Процессор в каждый момент времени может обрабатывать только одно сообщение. Процесс обработки запросов в системе включает следующие этапы: 1) формирование и передача запроса с терминала и запись его в НМД 1; 2) считывание запроса из НМД 1 в ОП; 3) загрузка в ОП с НМД 2 прикладной программы и поиск информации; 4) чтение найденной информации в ОП, ее обработка и вывод сообщения из ОП на терминал. Суммарное время обработки запроса в среднем составляет 60 с. Среднее время записи в НМД 1 — 20 с, записи с НМД 1 в ОП — 10 с; время поиска информации 30±5 с. Смоделировать работу системы в течение N часов. Определить загрузки устройств, средние времена ожидания ответа на запрос, параметры очередей сообщений.

# **Разработка Q-схемы**

Особенности непрерывно-стохастического подхода рассмотрим на примере использования в качестве типовых математических схем систем массового обслуживания, которые будем называть Q-схемами [4]. Системы массового обслуживания представляют собой класс математических схем, разработанных в теории массового обслуживания и различных приложениях для формализации процессов функционирования систем, которые по своей сути являются процессами обслуживания [5].

В качестве процесса обслуживания могут быть представлены различные по своей физической природе процессы функционирова­ния экономических, производственных, технических и других си­стем, например, потоки поставок продукции некоторому предприя­тию, потоки деталей и комплектующих изделий на сборочном конвейере цеха, запроса на обработку информации ЭВМ от удаленных терминалов и т. д. При этом характерным для работы таких объек­тов является случайное появление заявок (требований) на обслу­живание и завершение обслуживания в случайные моменты време­ни, т.е. стохастический характер процесса их функционирования [6]. Остановимся на основных понятиях массового обслуживания, не­обходимых для использования Q-схем как при аналитическом, так и при имитационном подходе.



Рисунок 1 – Схема программы

Генератор запросов создаёт запрос с определённой периодичностью и отправляет его на один из трёх терминалов. С терминала запрос отправляется в накопитель на магнитных дисках 1. Если основная память свободна, то запрос переходит в основную память. Если в основную память пришёл запрос, то из накопителя на магнитных дисках 2 в основную память направляются инструкции по решению запроса. Из основной памяти запрос и инструкция отправляются в процессор, где обрабатываются. После этого решённый запрос отправляется на терминал, с которого был получен запрос.

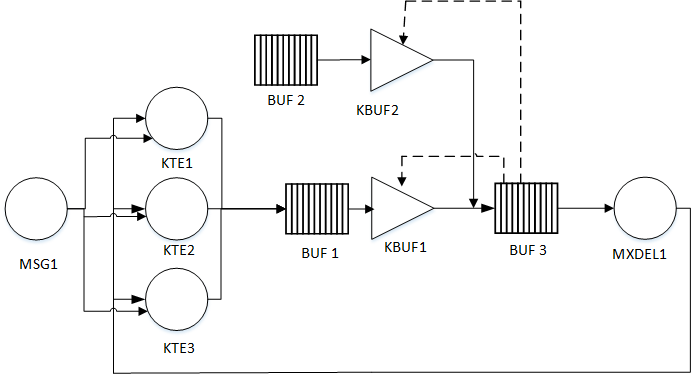


Рисунок 2 – Q-схема программы

Назначение элементов разработанной модели состоит в следующем.

Генератор MSG1 имитируют входящие потоки запросов.

Клапаны KTEi являются терминалами приёма запросов и вывода их решения.

Очереди BUFi соответствуют буферным блокам.

Клапаны KBUFi предназначены для разрешения / запрета выбора запросов из соответствующих буферных блоков для обработки.

Каналы MXDELi предназначены для имитации обработки считанного запроса.

Открытие клапанов обеспечивается согласно следующим условиям:

KBUF1: BUF 3 пуст;

KBUF2: BUF 3 не пуст;

С помощью созданной Q-схемы составим иерархию классов для автоматизированной справочно-информационной системы.

# **Разработка иерархии классов**

C# является полноценным объектно-ориентированным языком. Это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов [7].

Описанием объекта является класс, а объект представляет экземпляр этого класса. Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке, у которого есть имя, возраст, какие-то другие характеристики. То есть некоторый шаблон - этот шаблон можно назвать классом. Конкретное воплощение этого шаблона может отличаться, например, одни люди имеют одно имя, другие - другое имя. И реально существующий человек (фактически экземпляр данного класса) будет представлять объект этого класса [8].

Управлять большим количеством разрозненных классов довольно сложно. С этой проблемой можно справиться путем упорядочивания и ранжирования классов, то есть объединяя общие для нескольких классов свойства в одном классе и используя его в качестве базового. Эту возможность предоставляет механизм наследования [9].

Наследование применяется для следующих взаимосвязанных целей:

1. исключения из программы повторяющихся фрагментов кода;
2. упрощения модификации программы;
3. упрощения создания новых программ на основе существующих.

Более того, наследование является единственной возможностью использовать объекты, исходный код которых недоступен, но в которые требуется внести изменения [10].

Создадим иерархию классов для автоматизированной справочно-информационной системы.

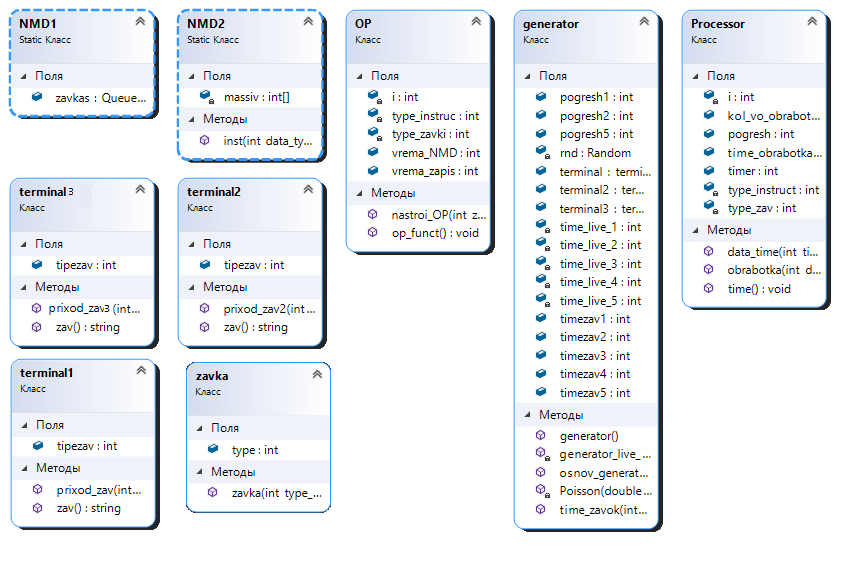


Рисунок 3 – Иерархия классов

Таблица 1 - описание классов

|  |  |  |
| --- | --- | --- |
| Название | Поля | Методы |
| generator | int pogresh1 – погрешность в расчётах времени появления для 1 типа запросов.  int pogresh2 – погрешность в расчётах времени появления для 2 типа запросов.  int pogresh5 – погрешность в расчётах времени появления для 5 типа запросов.  Random rnd – конструктор генератора псевдослучайных чисел.  terminal – конструктор класса terminal1.  terminal2 – конструктор класса terminal2.  terminal3 – конструктор класса terminal3.  int timezav1 – время появления запросов 1 типа.  int timezav2 – время появления запросов 2 типа.  int timezav3 – время появления запросов 3 типа.  int timezav4 – время появления запросов 4 типа.  int timezav5 – время появления запросов 5 типа. | generator() – функция для объявления некоторых конструкторов и первичной генерации времени жизни запросов.  generator\_live\_zav(int)– функция для генерации времени появления запросов.  osnov\_generator() – функция для генерации запросов.  Poisson(double) – функция для расчёта пуассоновского потока.  time\_zavok(int, int, int, int, int, int, int, int) – функция для получения настроек генерации запросов. |

Продолжение таблицы 1:

|  |  |  |
| --- | --- | --- |
| Название | Поля | Методы |
| terminal1 | int tipezav – тип запроса пришедшей в терминал. | prixod\_zav(int) – функция отвечающая за приём и отправку запроса в накопитель NMD1.  zav() – функция для передачи типа запроса. |
| terminal2 | int tipezav – тип запроса пришедшей в терминал. | prixod\_zav2(int) – функция отвечающая за приём и отправку запроса в накопитель NMD1.  zav() – функция для передачи типа запроса. |
| terminal3 | int tipezav – тип запроса пришедшей в терминал. | prixod\_zav3(int) – функция отвечающая за приём и отправку запроса в накопитель NMD1.  zav() – функция для передачи типа запроса. |
| NMD1 | Queue zavkas – очередь с принципом «Первый пришёл, первый ушёл». | - |
| NMD2 | int massiv – массив с инструкциями к запросам. | inst(int) – функция передающая нужную инструкцию для решения запроса. |
| OP | int i – переменная для цикла расчёта времени.  int type\_instruct – инструкция к заявке.  int type\_zavki – тип запроса.  int vrema\_NMD – время записи запроса в НМД1  int vrema\_zapis – время записи запроса в ОП. | nastroi\_OP (int, int) – функция для получения настроек времени записи.  op\_fuct() – функция для отправки данных в процессор. |

Продолжение таблицы 1:

|  |  |  |
| --- | --- | --- |
| Название | Поля | Методы |
| Processor | int i – переменная для цикла расчёта времени.  int kol\_vo\_obrabotan\_zav – количество обработанных запросов.  int pogresh – погрешность в расчёте времени обработки.  int time\_obrabotka – время обработки запроса.  int timer – время обработки запроса с погрешностью.  int type\_instruct – инструкция к запросу.  int type\_zav – тип запроса. | data\_time(int, int) – функция для получения времени обработки и погрешности.  obrabotka() – функция для обработки поступивших запросов.  time() – функция для генерации времени обработки с погрешностью. |
| zavka | int type – тип запроса | zavka(int) – функция записи. |

Используя Q-схему и иерархию классов создадим прикладную программу на языке программирования C#

# **Разработка руководства пользователя**

Проверим созданную прикладную программу. При запуске программы открывается главное окно программы.

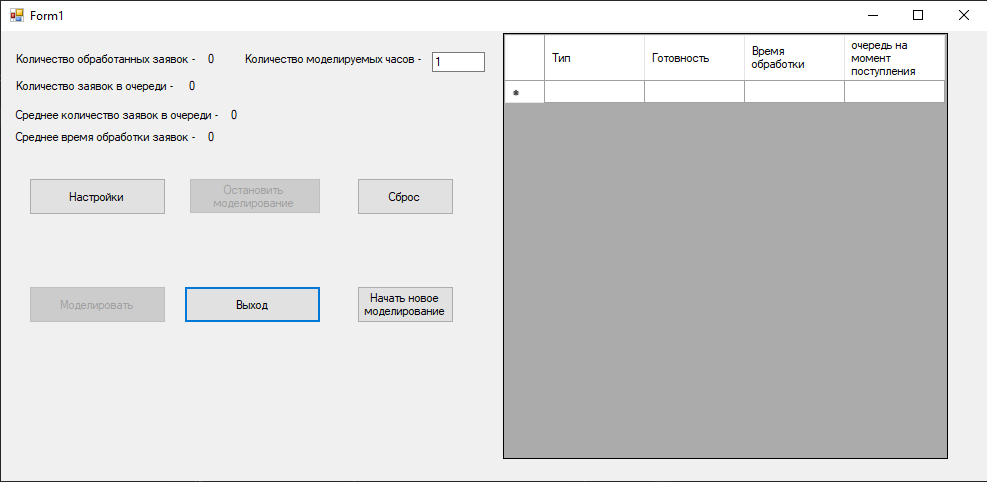


Рисунок 4 – Главная форма приложения

Перед началом моделирования надо ввести количество моделируемых часов. По умолчанию один час. Для изменения настроек моделирования надо нажать кнопку «Настройки».

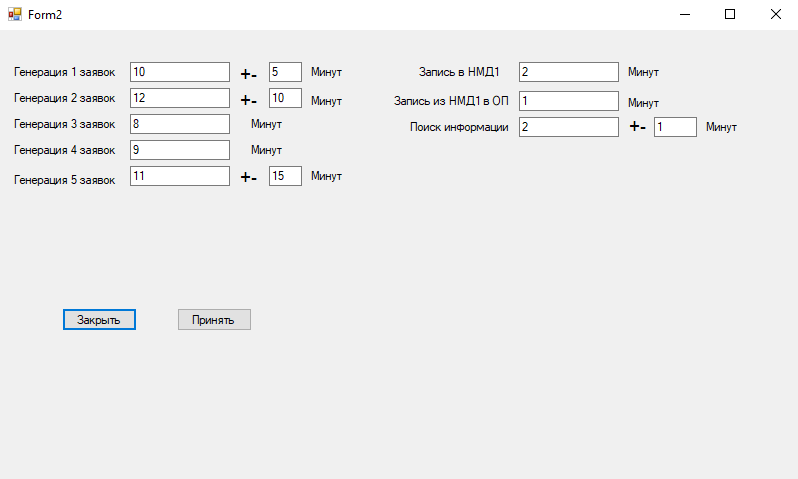


Рисунок 5 – форма настроек

При нажатии кнопки «Принять» происходит установка новых настроек для моделирования. При нажатии кнопки «Назад» происходит выход из формы. Для того, чтобы подготовить программу к моделированию, необходимо нажать кнопку «Начать новое моделирование». После этого, чтобы начать моделирование надо нажать кнопку «Моделировать».

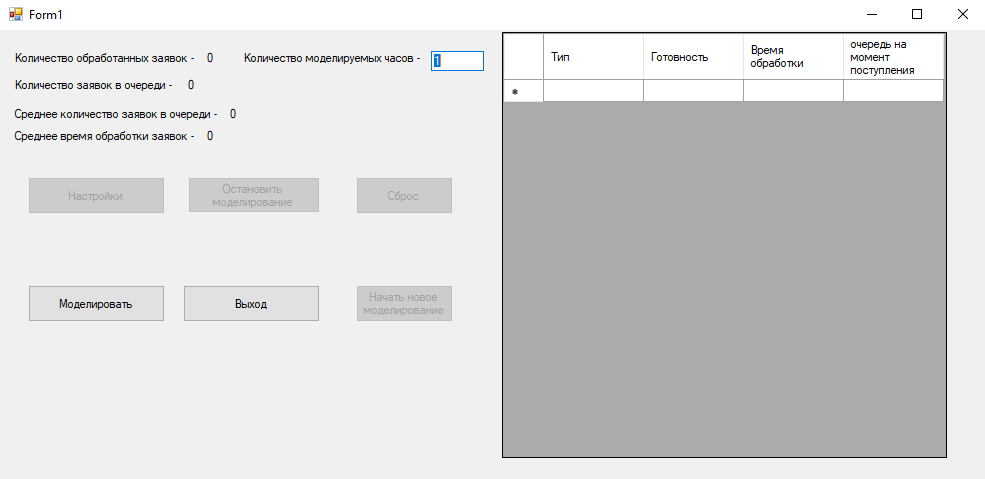


Рисунок 6 – Подготовка к моделированию

Начинается моделирование запросов. Для остановки моделирования необходимо нажать кнопку «Остановить моделирование».

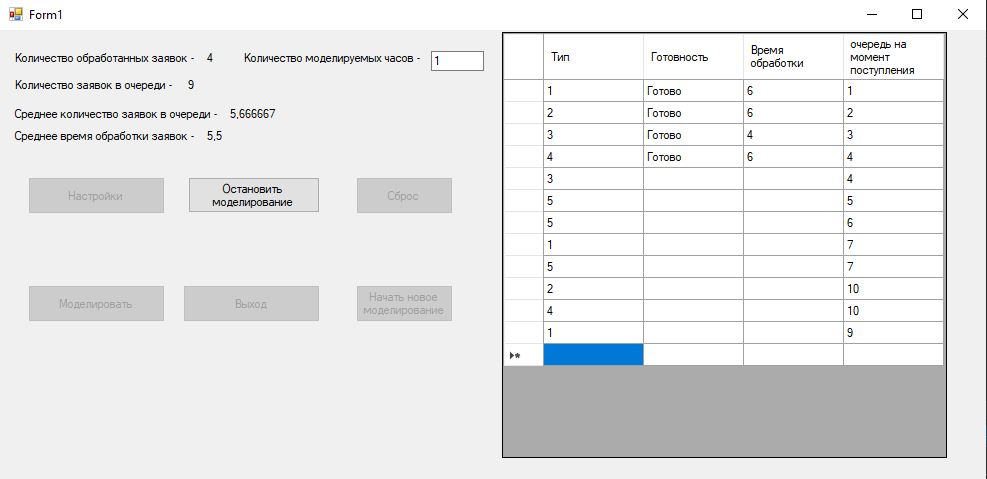


Рисунок 7 – Процесс моделирования

После остановки моделирования можно поменять настройки моделирования, сбросить моделирование или продолжить. Для сброса надо нажать кнопку «Сброс».

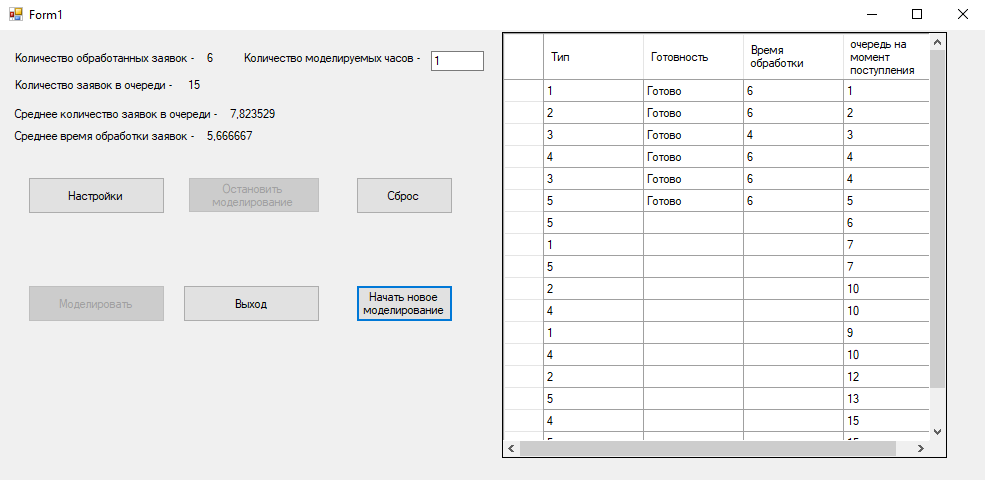


Рисунок 8 – Процесс остановки программы

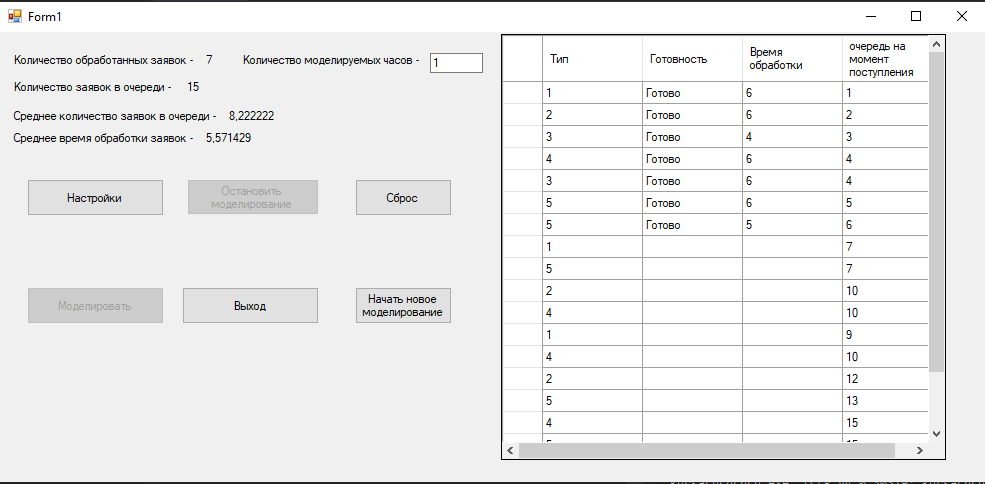


Рисунок 9 – Продолжение моделирования

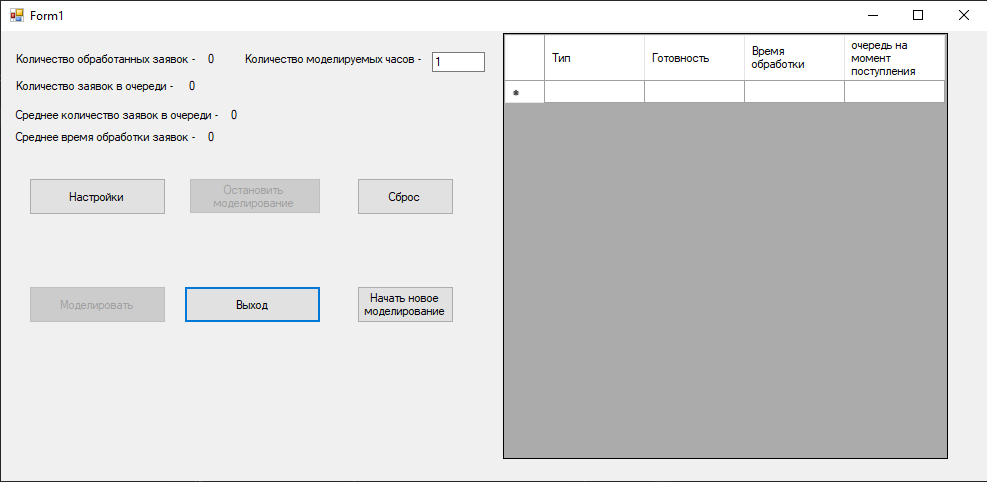


Рисунок 10 – Сброс работы программы

Прикладная программа работает без ошибок. Исходный код прикладной программы. (См. Листинг программы)

# **Заключение**

Данная работа выполнена на языке высокого уровня программирования в среде Visual studio 2019. При написании и разработке программного обеспечения использовались различные возможности C# и Windows Form.

Для разработки прикладной программы были созданы: схема программы, Q-схема программы, иерархия классов и руководство пользователя.

Разработанная прикладная программа имеет удобный пользовательский интерфейс. Все элементы прикладной программы представлены как единый проект, управление которым осуществляется через окно главной формы проекта. Программа работает без ошибок, выполняет все функции.

Проведено моделирование СМО в течении 1 часа. Выявлено, что было обработано 7 запросов, количество запросов, оставшихся в очереди 15, среднее количество заявок в очереди 8,2, среднее время обработки запросов 5,57 минут.

# **Список использованных источников**

1. Гнеденко, Б. В. Введение в теорию массового обслуживания [Текст]/ Б.В. Гнеденко, И.Н. Коваленко. - М.: Главная редакция физико-математической литературы издательства "Наука", 2018. - 336 c.

2. Карташевский, В.Г. Основы теории массового обслуживания. [Текст]: Учебник для вузов / Карташевский Вячеслав Григорьевич. - М.: Горячая линия - Телеком, 2017. - 449 c.

3. Кофман, А. Массовое обслуживание. Теория и приложения [Текст]/ А. Кофман, Р. Крюон. - М.: Мир, 2017. - 302 c.

4. Певзнер, Л. Д. Математические основы теории систем [Текст]/ Л.Д. Певзнер, Е.П. Чураков. - М.: Высшая школа, 2017. - 504 c.

5. Гнеденко, Б.В. Введение в теорию массового обслуживания [Текст]/ Б.В. Гнеденко. - Москва: Огни, 2017. - 195 c.

6. Зыков, С. В. Программирование. Объектно-ориентированный подход [Текст]: учебник и практикум для академического бакалавриата/ С. В. Зыков. - М. : Издательство Юрайт, 2019. - 155 с.

7. Гниденко, И. Г. Технология разработки программного обеспечения [Текст]: учебное пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. - М. : Издательство Юрайт, 2017. - 235 с.

8. Кудрина, Е. В. Основы алгоритмизации и программирования на языке c# [Текст]: учебное пособие для СПО / Е. В. Кудрина, М. В. Огнева. - М. : Издательство Юрайт, 2019. - 322 с.

9. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем [Текст]: учебник для вузов / Е. М. Лаврищева. - 2-е изд., испр. и доп. - М. : Издательство Юрайт, 2019. — 432 с.

10. Иванова, Г.С. Технология программирования

[Текст]: учебник / Г. С. Иванова. - М. : Кнорус, 2016. - 336 с.2

# **Листинг программы**

Класс Form1:

public partial class Form1 : Form

{int i = 0; // переменная необходимая для отсчёта количества строк

int b = 0; // переменная необходимая для отсчёта количества строк

int time; // переменная необходимая для получения времени моделирования

int tekyshai\_time = 0; // переменная необходимая для отсчёта времени прошедшего с начала моделирования

private generator generator; // объявление класса generator

private OP OP; // объявление класса OP

public Form1()

{InitializeComponent();

label1.TextChanged += tabl; }

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

{timer1.Start();

button1.Enabled = false;

button5.Enabled = false;

button2.Enabled = false;

button4.Enabled = true;}

private void button2\_Click(object sender, EventArgs e)

{Close();}

private void button3\_Click(object sender, EventArgs e)

{Form2 f2 = new Form2();

f2.Show();}

private void Form1\_Load(object sender, EventArgs e)

{button1.Enabled = false;button4.Enabled = false;}

private void timer1\_Tick(object sender, EventArgs e)

{if (tekyshai\_time < time){

generator.osnov\_generator();

OP.op\_funct();

update();

tekyshai\_time++;}

else

{update();

timer1.Stop();

button2.Enabled = true;

button5.Enabled = true;

button4.Enabled = false;

button3.Enabled = true;

button6.Enabled = true;}}

public void update() // функция для обновления статистики

{Single tmp = 0;

Single tmp2 = 1;

Single tmp3 = 0;

Single tmp4 = 1;

if (!(generator.terminal.tipezav == 0)){

dataGridView1.Rows.Add();

dataGridView1.Rows[i].Cells[0].Value = generator.terminal.zav();

dataGridView1.Rows[i].Cells[3].Value = NMD1.zavkas.Count();

i++;}

if(!(generator.terminal2.tipezav == 0))

{dataGridView1.Rows.Add();

dataGridView1.Rows[i].Cells[0].Value = generator.terminal2.zav();

dataGridView1.Rows[i].Cells[3].Value = NMD1.zavkas.Count();

i++;}

if (!(generator.terminal3.tipezav == 0))

{dataGridView1.Rows.Add();

dataGridView1.Rows[i].Cells[0].Value = generator.terminal3.zav();

dataGridView1.Rows[i].Cells[3].Value = NMD1.zavkas.Count();

i++;}

label1.Text = Convert.ToString(Processor.kol\_vo\_obrabotan\_zav);

label4.Text = Convert.ToString(NMD1.zavkas.Count());

for(int c = 0; c < (dataGridView1.RowCount); c++)

{if (!(dataGridView1.Rows[c].Cells[3].Value == null))

{tmp = tmp + Convert.ToSingle(dataGridView1.Rows[c].Cells[3].Value);

tmp2++;}}

for (int p = 0; p < (dataGridView1.RowCount); p++)

{

if (!(dataGridView1.Rows[p].Cells[2].Value == null))

{tmp3 = tmp3 + Convert.ToSingle(dataGridView1.Rows[p].Cells[2].Value);

tmp4++;}}

label8.Text = Convert.ToString(tmp / (tmp2-1));

label9.Text = Convert.ToString(tmp3 / (tmp4-1));}

private void button4\_Click(object sender, EventArgs e)

{timer1.Stop();

button2.Enabled = true;

button5.Enabled = true;

button4.Enabled = false;

button3.Enabled = true;

button6.Enabled = true;}

private void button5\_Click(object sender, EventArgs e)

{generator = new generator();

OP = new OP();

Processor.time();

if (textBox1.Text == "")

MessageBox.Show("Отсутствует время моделирования");

else

{

time = Convert.ToInt32(textBox1.Text)\*60;

button3.Enabled = false;

button1.Enabled = true;

button5.Enabled = false;

button6.Enabled = false;

}

}

private void tabl(object sender, EventArgs e) // функция для заполнения таблицы

{

dataGridView1.Rows[b].Cells[1].Value = "Готово";

dataGridView1.Rows[b].Cells[2].Value = Convert.ToString(Processor.timer + OP.vrema\_zapis + OP.vrema\_NMD);

b++;

}

private void button6\_Click(object sender, EventArgs e)

{

tekyshai\_time = 0;

Processor.kol\_vo\_obrabotan\_zav = 0;

label1.Text = "0";

label4.Text = "0";

label8.Text = "0";

label9.Text = "0";

i = 0;

b = 0;

dataGridView1.Rows.Clear();

NMD1.zavkas.Clear();}}

Класс Form2:

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

Close();

}

private void button2\_Click(object sender, EventArgs e)

{

generator objgenerator = new generator();

objgenerator.time\_zavok(Convert.ToInt32(textBox1.Text), Convert.ToInt32(textBox2.Text), Convert.ToInt32(textBox3.Text), Convert.ToInt32(textBox4.Text), Convert.ToInt32(textBox5.Text), Convert.ToInt32(textBox6.Text), Convert.ToInt32(textBox7.Text), Convert.ToInt32(textBox8.Text)); // передача новых настроек в класс generator

OP OP = new OP();

OP.nastroi\_OP(Convert.ToInt32(textBox10.Text), Convert.ToInt32(textBox9.Text)); // передача новых настроек в класс OP

Processor.data\_time(Convert.ToInt32(textBox11.Text), Convert.ToInt32(textBox12.Text)); // передача новых настроек в класс Processor

}

}

Класс generator:

class generator

{ public terminal1 terminal;

public terminal2 terminal2;

public terminal3 terminal3;

static public int timezav1 = 10;//базовое время генерации запроса 1 типа

static public int timezav2 = 12; //базовое время генерации запроса 2 типа

static public int timezav3 = 8; //базовое время генерации запроса 3 типа

static public int timezav4 = 9; //базовое время генерации запроса 4 типа

static public int timezav5 = 11; //базовое время генерации запроса 5 типа

static public int pogresh1 = 5; //базовая погрешность генерации запроса 1 типа

static public int pogresh2 = 10; //базовая погрешность генерации запроса 2 типа

static public int pogresh5 = 15; //базовая погрешность генерации запроса 5 типа

private int time\_live\_1; // время генерации запроса с погрешностью

private int time\_live\_2; // время генерации запроса с погрешностью

private int time\_live\_3; // время генерации запроса с погрешностью

private int time\_live\_4; // время генерации запроса с погрешностью

private int time\_live\_5; // время генерации запроса с погрешностью

private Random rnd;

public void time\_zavok(int date\_zav1, int date\_zav2, int date\_zav3, int date\_zav4, int date\_zav5, int date\_pog1, int date\_pog2, int date\_pog5) // функция для получения настроек времени для генерации запросов

{ timezav1 = date\_zav1;

timezav2 = date\_zav2;

timezav3 = date\_zav3;

timezav4 = date\_zav4;

timezav5 = date\_zav5;

pogresh1 = date\_pog1;

pogresh2 = date\_pog2;

pogresh5 = date\_pog5; }

public generator() // функция для генерации времени получения запроса

{rnd = new Random();

terminal = new terminal1();

terminal2 = new terminal2();

terminal3 = new terminal3();

time\_live\_1 = Convert.ToInt32(rnd.NextDouble() \* ((timezav1 + pogresh1) - (timezav1 - pogresh1)) + (timezav1 - pogresh1));

time\_live\_2 = Convert.ToInt32(Poisson(Convert.ToDouble(timezav3)));

time\_live\_3 = Convert.ToInt32(Poisson(Convert.ToDouble(timezav4)));

time\_live\_4 = Convert.ToInt32(rnd.NextDouble() \* ((timezav1 + pogresh1) - (timezav1 - pogresh1)) + (timezav1 - pogresh1));

time\_live\_5 = Convert.ToInt32(rnd.NextDouble() \* ((timezav5 + pogresh5) - (timezav5 - pogresh5)) + (timezav5 - pogresh5));}

public void osnov\_generator() // функция для генерации запросов

{if (time\_live\_1 < 1)

{terminal.prixod\_zav(1);

generator\_live\_zav(1);}

else time\_live\_1--;

if (time\_live\_2 < 1)

{terminal.prixod\_zav(2);

generator\_live\_zav(2);}

else time\_live\_2--;

if (time\_live\_3 < 1)

{terminal2.prixod\_zav2(3);generator\_live\_zav(3);}

else time\_live\_3--;

if (time\_live\_4 < 1)

{terminal2.prixod\_zav2(4);

generator\_live\_zav(4);}

else time\_live\_4--;

if (time\_live\_5 < 1)

{terminal3.prixod\_zav3(5);

generator\_live\_zav(5);}

else time\_live\_5--;}

private void generator\_live\_zav(int type\_zav) // генератор времени получения запроса

{Random rnd = new Random();

switch (type\_zav)

{

case 1:

time\_live\_1 = Convert.ToInt32(rnd.NextDouble() \* ((timezav1 + pogresh1) - (timezav1 - pogresh1)) + (timezav1 - pogresh1));

break;

case 2:

time\_live\_2 = Convert.ToInt32(rnd.NextDouble() \* ((timezav2 + pogresh2) - (timezav2 - pogresh2)) + (timezav2 - pogresh2));

break;

case 3:

time\_live\_3 = Convert.ToInt32(Poisson(Convert.ToDouble(timezav3)));

break;

case 4:

time\_live\_4 = Convert.ToInt32(Poisson(Convert.ToDouble(timezav3)));

break;

case 5:

time\_live\_5 = Convert.ToInt32(rnd.NextDouble() \* ((timezav5 + pogresh5) - (timezav5 - pogresh5)) + (timezav5 - pogresh5));

break; } }

private int Poisson(double lambda) // функция для Пуассоновского распределения

{Random rand = new Random();

double l = Math.Exp(-lambda);

double p = 1.0;

int k = 0;

while (p > l)

{

k++;

p \*= rand.NextDouble();

};

return k - 1;}}

Класс terminal1:

class terminal1

{

public int tipezav;

public void prixod\_zav(int data\_type\_zav) // функция для записи запроса в NMD1

{

tipezav = data\_type\_zav;

NMD1.zavkas.Enqueue(new zavka(tipezav));

}

public string zav() // функция для получения типа заявки

{

string str = "";

str = Convert.ToString(tipezav);

tipezav = 0;

return str;

}

Класс terminal2:

class terminal2

{

public int tipezav;

public void prixod\_zav2(int data\_type\_zav) //функция для записи запроса в NMD1

{

tipezav = data\_type\_zav;

NMD1.zavkas.Enqueue(new zavka(tipezav));

}

public string zav() // функция для получения типа заявки

{

string str2 = "";

str2 = Convert.ToString(tipezav);

tipezav = 0;

return str2;}

Класс terminal3:

class terminal3

{

public int tipezav;

public void prixod\_zav3(int data\_type\_zav) //функция для записи запроса в NMD1

{

tipezav = data\_type\_zav;

NMD1.zavkas.Enqueue(new zavka(tipezav));

}

public string zav()// функция для получения типа заявки

{

string str3 = "";

str3 = Convert.ToString(tipezav);

tipezav = 0;

return str3;}

Класс NMD1: // класс для хранения информации о запросах

static class NMD1

{

public static Queue<zavka> zavkas = new Queue<zavka>();}

Класс NMD2: // класс для хранения инструкций

static class NMD2

{

private static int[] massiv = new int[6] { 1, 2, 3, 4, 5, 8 };

public static int inst(int data\_type)// функция получения инструкций для запроса

{

switch (data\_type)

{

case 1:

return massiv[0];

case 2:

return massiv[1];

case 3:

return massiv[2];

case 4:

return massiv[3];

case 5:

return massiv[4];

case 0:

return massiv[5];

}

return massiv[5];}}

Класс OP:

class OP

{

private int type\_zavki;

private int type\_instruc;

private int i = 0;

public static int vrema\_zapis = 1;

public static int vrema\_NMD = 2;

public void op\_funct() // функция записи запроса в OP и отправки в Processor

{

if (i < vrema\_zapis)

i++;

else

if (NMD1.zavkas.Count >0)

{

type\_zavki = NMD1.zavkas.Peek().type;

type\_instruc = NMD2.inst(type\_zavki);

Processor.obrabotka(type\_zavki, type\_instruc);

i = 0;

}

}

public void nastroi\_OP(int zapis, int NMD)// функция для получения настроек OP

{

vrema\_zapis = zapis;

vrema\_NMD = NMD;

} }

Класс Processor:

class Processor

{static private int i = 0;

static private int type\_zav;

static private int type\_instruct;

static public int time\_obrabotka = 2;

static public int pogresh = 1;

static public int kol\_vo\_obrabotan\_zav;

static public int timer;

static public void obrabotka(int data\_type\_zav,int data\_type\_instruct) //функция для обработки запросов

{type\_zav = data\_type\_zav;

type\_instruct = data\_type\_instruct;

if (i < timer)

i++;

else

{ if (type\_zav == type\_instruct)

{NMD1.zavkas.Dequeue();

kol\_vo\_obrabotan\_zav++;}

i = 0;

time();}}

static public void time() // функция для генерации времени обработки запроса

{Random rnd = new Random();

timer = Convert.ToInt32(rnd.NextDouble() \* ((time\_obrabotka + pogresh) - (time\_obrabotka - pogresh)) + (time\_obrabotka - pogresh));}

static public void data\_time(int time, int pog)

{time\_obrabotka = time;

pogresh = pog; }

Класс zavka:

class zavka

{ public int type;

public zavka(int type\_zavka)

{ this.type = type\_zavka}}