Оглавление

[**Введение** 2](#_Toc57021557)

[**Постановка задачи** 3](#_Toc57021558)

[**Принцип особых состояний.** 4](#_Toc57021559)

[**Формализованная схема и описание СМО.** 4](#_Toc57021560)

[**Исходные данные** 6](#_Toc57021561)

[**Блок-схема** 7](#_Toc57021562)

[**Список особых событий** 9](#_Toc57021563)

[**Временная диаграмма** 10](#_Toc57021564)

[**Вывод законов распределения** 11](#_Toc57021565)

[**Ограничения и требуемые характеристики** 12](#_Toc57021566)

[**Модульная структура** 13](#_Toc57021567)

[**Описание работы программы** 14](#_Toc57021568)

[**Результаты работы** 16](#_Toc57021569)

[**Анализ результатов** 17](#_Toc57021570)

[**Вывод** 18](#_Toc57021571)

# **Введение**

Целью практической курсовой является создание модели ВС или ее компонентов на некотором уровне детализации, описывающей и имитирующей ее структуру и функциональность.

Каждый реальный объект ВС обладает огромной сложностью, определяемой множеством состояний, множеством внутренних и внешних связей, множеством анализируемых характеристик. Модель дает приближенное описание объекта с целью получения требуемых результатов с определенной точностью и достоверностью. Степень приближения модели к описываемому объекту может быть различной и зависит от требований задачи.

Существуют различные типы моделей ВС: аналитические, аналоговые, физические и имитационные. В данной работе будет использоваться имитационная модель ВС. Одним из подходов к построению имитационной модели является построение ее в виде системы массового обслуживания (СМО).

# **Постановка задачи**

Целью курсовой работы является создание модели вычислительной системы (ВС) или ее части на некотором уровне детализации, описывающей и имитирующей ее структуру и функциональность.

Каждый реальный объект (реальная ВС) обладает бесконечной сложностью, множеством характеристик, внутренних и внешних связей. Модель есть приближенное описание объекта с целью получения требуемых результатов с определенной точностью и достоверностью.

При необходимости исследования поведенческих характеристик ВС в процессе исследования выгодно использовать не сам объект, а его модель. Степень приближения модели к описываемому объекту может быть различной и зависит от требований задачи.

Существуют различные типы моделей:

* Аналитические (математические) модели
* Аналоговые модели
* Физические модели
* Имитационные модели

Последний тип моделей является предметом нашего изучения. Одним из подходов к построению имитационной модели

является построение ее в виде системы массового обслуживания (СМО), с характерной для СМО терминологией: источник, буфер, прибор, диспетчер, заявка (требование).

Существуют два подхода к построению моделирующего алгоритма:

Принцип Δt

Универсальный метод построения моделирующего алгоритма, когда состояние объекта проверяется через фиксированный интервал модельного времени. Суть его заключается в следующем: в каждый

момент времени *ti* = *ti*-1+ D*ti*-1 получают приближенные значения

характеристик исследуемого объекта. Δt можно получить детерминированным способом.

Основной критерий выбора Δt — он должен быть настолько мал, чтобы не пропустить событие в моделируемой системе, которое должно быть учтено при выбранной детальности моделирования. Метод неэффективен, т.к. постоянно проверяет состояние объектов моделирования, не изменяющихся при этом, особенно при малых Δt.

## **Принцип особых состояний.**

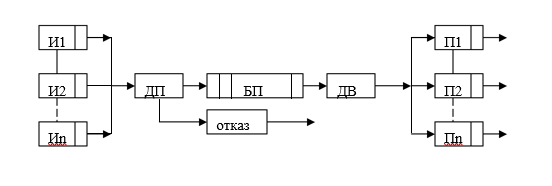
При исследовании реальной системы интервалы, в которых состояние ее не меняется, не представляют интереса. Имеют значение только переходы системы из одного состояния в другое в некоторые моменты времени. Эти переходы определяются особыми состояниями или событиями.

Рассмотрим некоторые типы особых событий, которые изменяют состояние системы:

* Поступление заявки в СМО (момент генерации заявки источником).
* Освобождение прибора (готовность прибора взять заявку на обслуживание).
* Окончание процесса моделирования.

Использование принципа особых событий для построения имитационной модели наиболее эффективно. В настоящей курсовой работе предлагается использовать именно этом принцип.

## **Формализованная схема и описание СМО.**



Здесь **Иi (i= 1..n)** – источник заявок, который генерирует заявки, а все вместе n источников создают входной поток заявок в систему.

Каждая заявка приходит в СМО со своими характеристиками. Это Tвх — время генерации заявки (время поступления её в СМО) и

номер заявки составленный из номера источника, сгенерировавшего заявку, и порядкового номера заявки от этого источника. Например, (2.3) – третья заявка от второго источника.

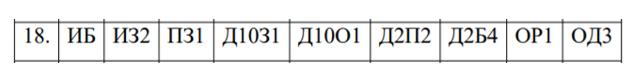
* — приборы, которые обслуживают заявки и создают выходной поток заявок после обслуживания.

**БП** —буферная память(место для хранения очереди заявок).

В общей памяти хранятся заявки от различных источников. Порядок их записи в БП определяется только дисциплиной буферизации.

**ДП** —диспетчер постановки заявок. **ДВ** —диспетчер выбора заявок.

# **Исходные данные**



**Источники:**

ИБ–бесконечный источник;

И32–равномерный закон распределения заявок;

**Приборы:**

П31 – экспоненциальный закон распределения времени обслуживания;

**Описание дисциплин постановки и выбора:**

***Дисциплина буферизации:***

Д1031–по кольцу;

***Дисциплина отказа:***

Д1001–под указателем;

***Дисциплина постановки на обслуживание:***

Д2П2–по кольцу;

Д2Б4–приоритет по номеру источника, по одной заявке;

**Виды отображения результатов работы программной модели:**

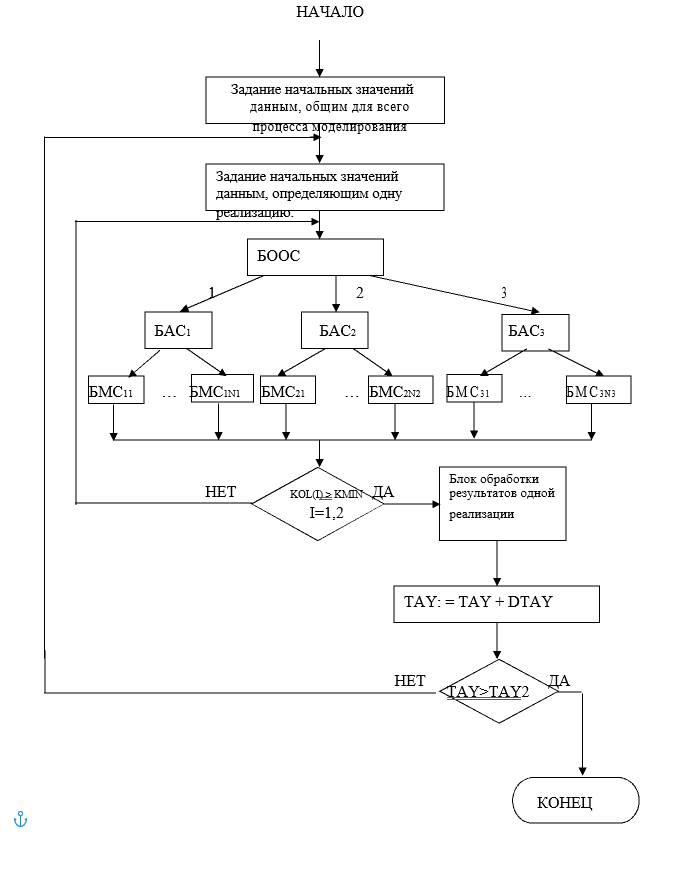
***Отражение результатов после сбора статистики:***

ОР1– сводная таблица результатов.

***Динамическое отражение результатов:***

ОД3– временные диаграммы, текущее состояние

# **Блок-схема**



БАС являются переключателем по номеру события.

Блоки БМС обеспечивают модификацию состояний элементов СМО и моделирование событий, являющихся следствием событий в активных элементах системы – источниках заявок и приборе.

Каждый блок БМС может представлять цепочку действий, состоящих из некоторых типовых, а именно:

1. Запись заявки в буфер;
2. Выборка заявки из буфера на обслуживание;
3. Обслуживание заявки;
4. Генерирование следующей заявки от источника с заданным номером;
5. Фиксация обслуженных заявок, времени ожидания заявок в буфере, числа заявок, получивших отказ и общего числа заявок по каждому из источников;
6. Имитация освобождения прибора.

# **Список особых событий**

В основе реализации моделирования методом особых событий лежит реализация перемещения по оси времени от особого события к особому событию на основе информационной структуры - календарь особых событий (КАОС).

КАОС – список ОС, где по каждому ОС приведены параметры:

- тип ОС (поступление заявки на вход системы, освобождение прибора, конец моделирования)

- Тос - момент времени наступления очередного ОС данного типа

- ПР признак, который принимает значение :

0 – известен момент наступления очередного ОС

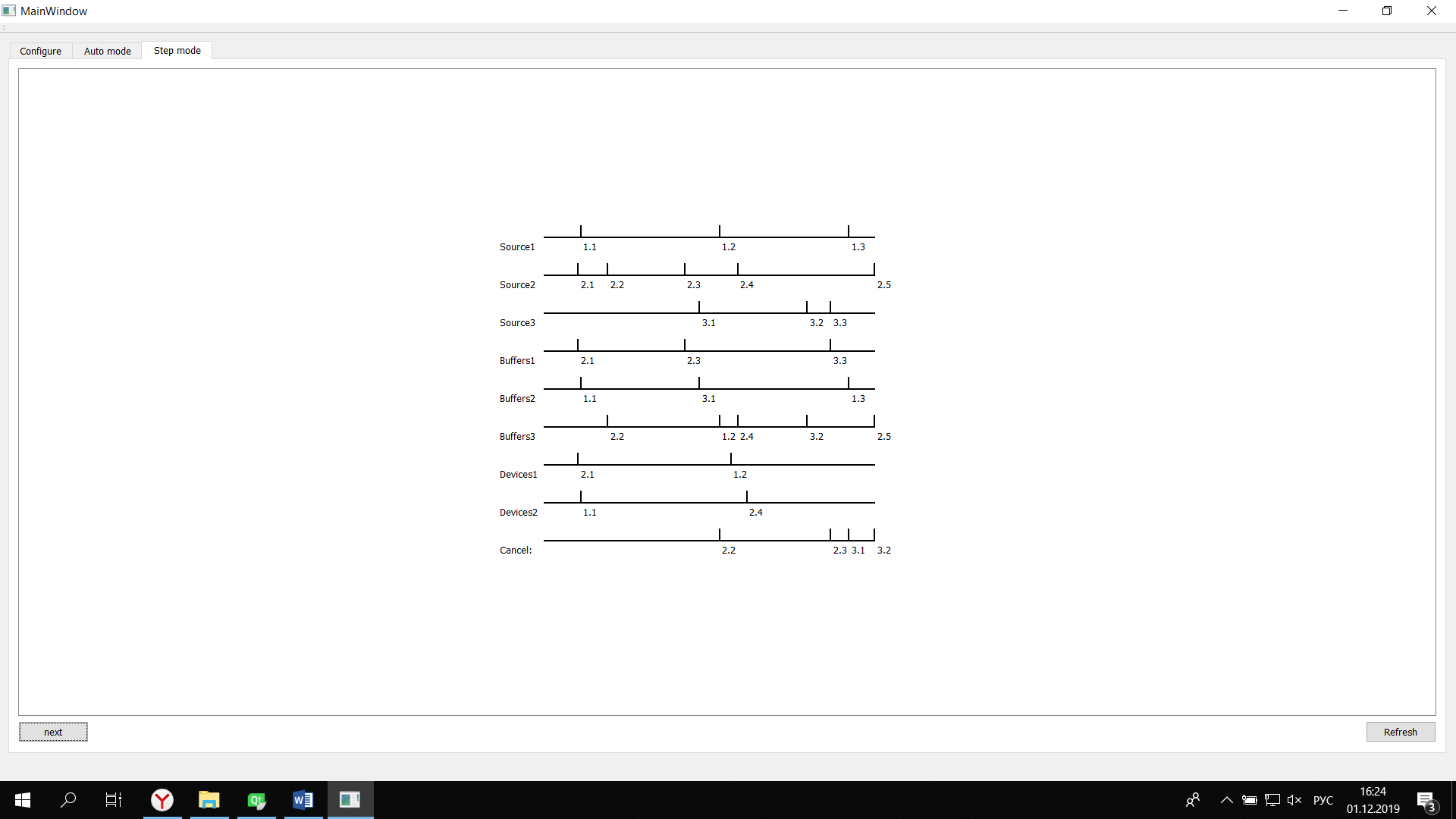
1 – не известен момент наступления очередного ОС

В нашем случае 0 у ОС «Поступление заявки» указывает на отсутствие очередной заявки от данного источника, а 0 у ОС «Освобождение прибора» означает, что не известен момент освобождения в будущем т.е. прибор уже свободен и находится в состоянии простоя.

Для определения следующего относительно текущего положения ОС необходимо в КАОС выбрать ОС с минимальным Тос среди всех ОС, у которых ПР=0. Именно эту функцию реализует блок БООС.

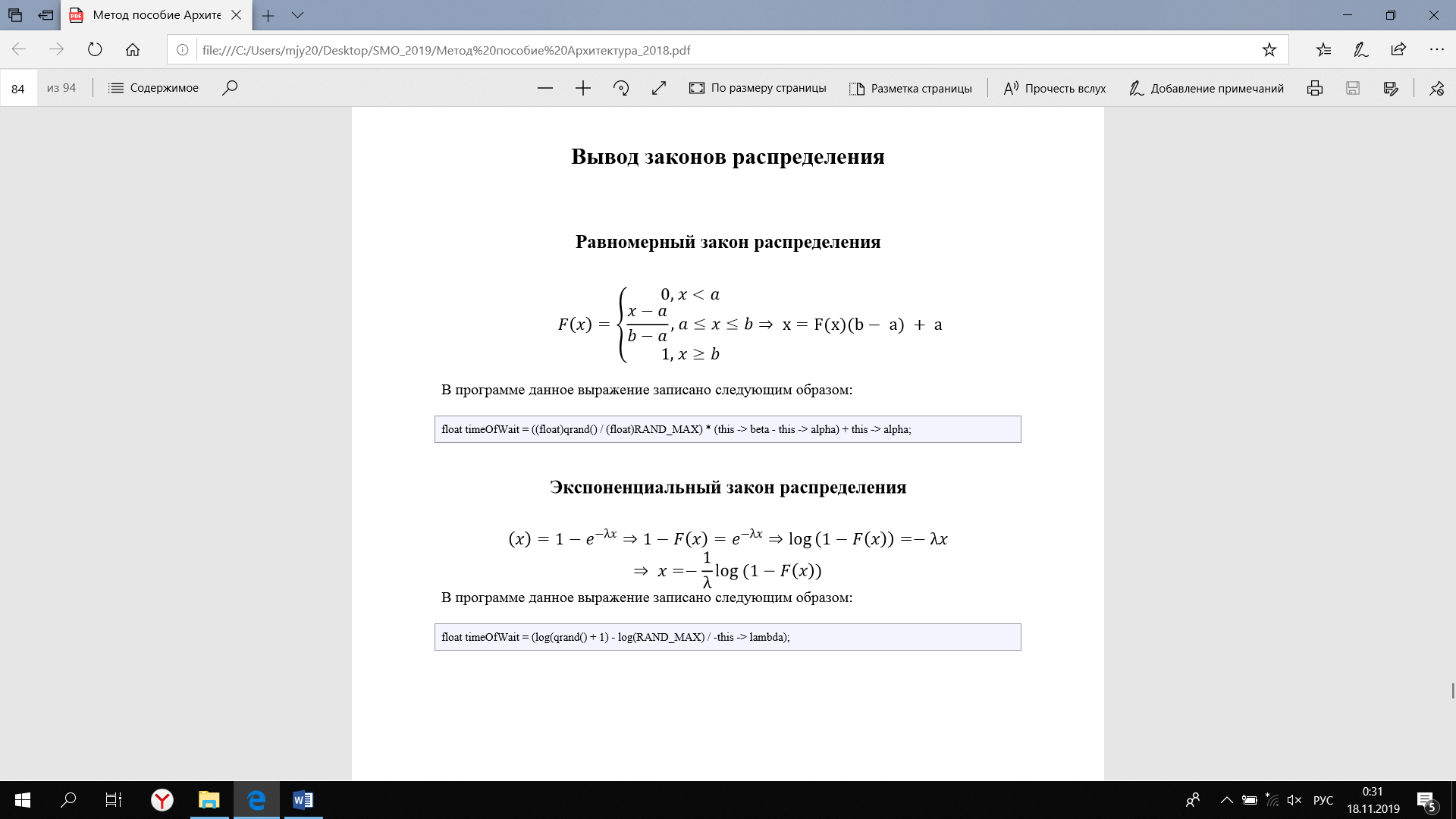
На основании результата БООС производится анализ типа ОС в блоке БАС, который подключает соответствующий БМС для модификации состояния системы.

# **Временная диаграмма**



# **Вывод законов распределения**

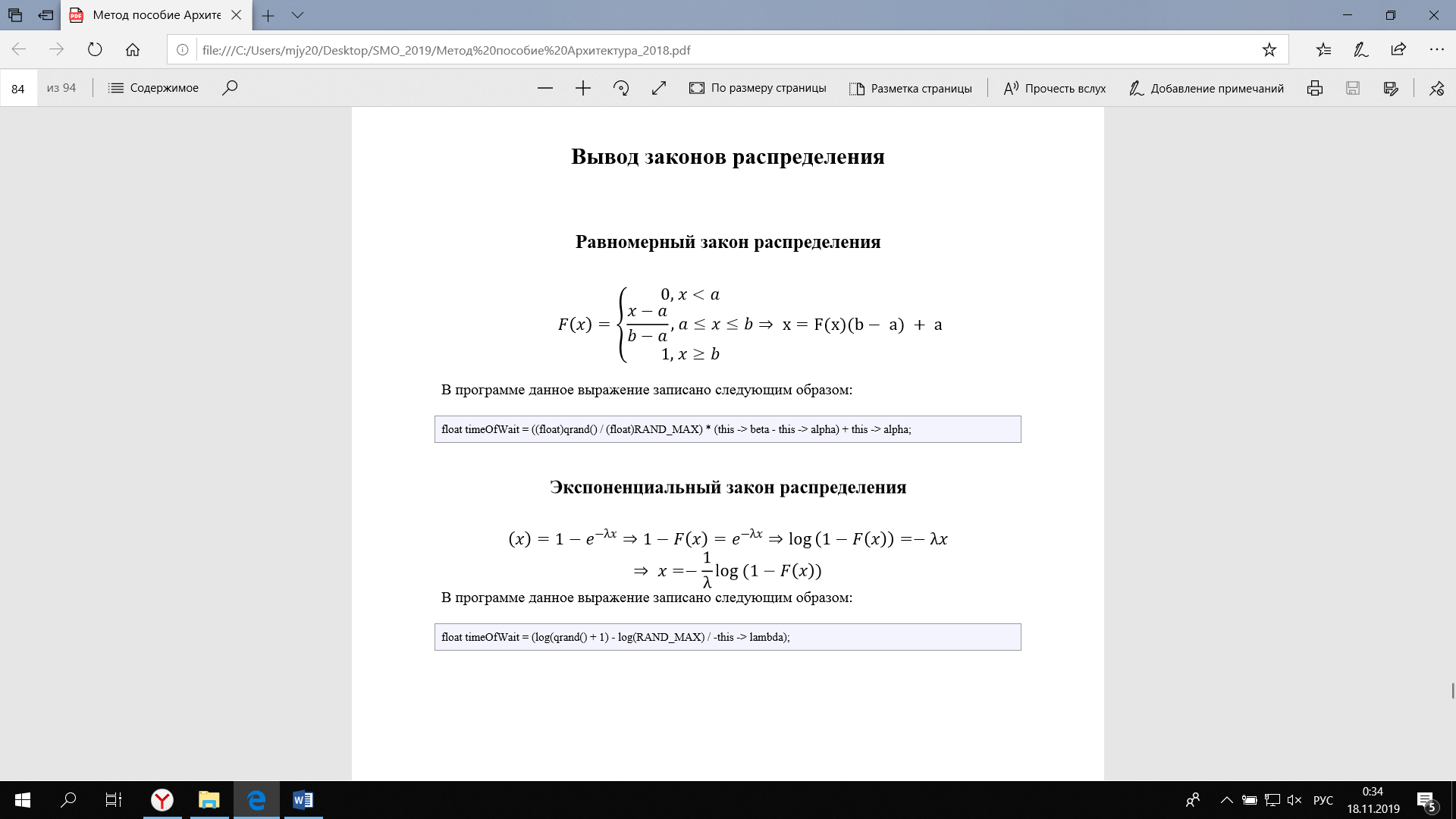
***Равномерный закон распределения:***



В программе данное выражение записано так:

**((float)rand()/(float)RAND\_MAX)\*(beta - alpha) + alpha;**

***Экспоненциальный закон распределения:***



В программе данное выражение записано так:

**((log(1 - ((float)rand()/(float)RAND\_MAX)))\*(-1/*this*->lambda));**

# **Ограничения и требуемые характеристики**

Вероятность отказа должна составлять не более 10%.

Загрузка приборов более 90%.

Время пребывания заявки в системе не ограничено, т.к. в зависимости от присланных данных, заявка может обрабатываться длительное время для получения верного результата.

# **Модульная структура**

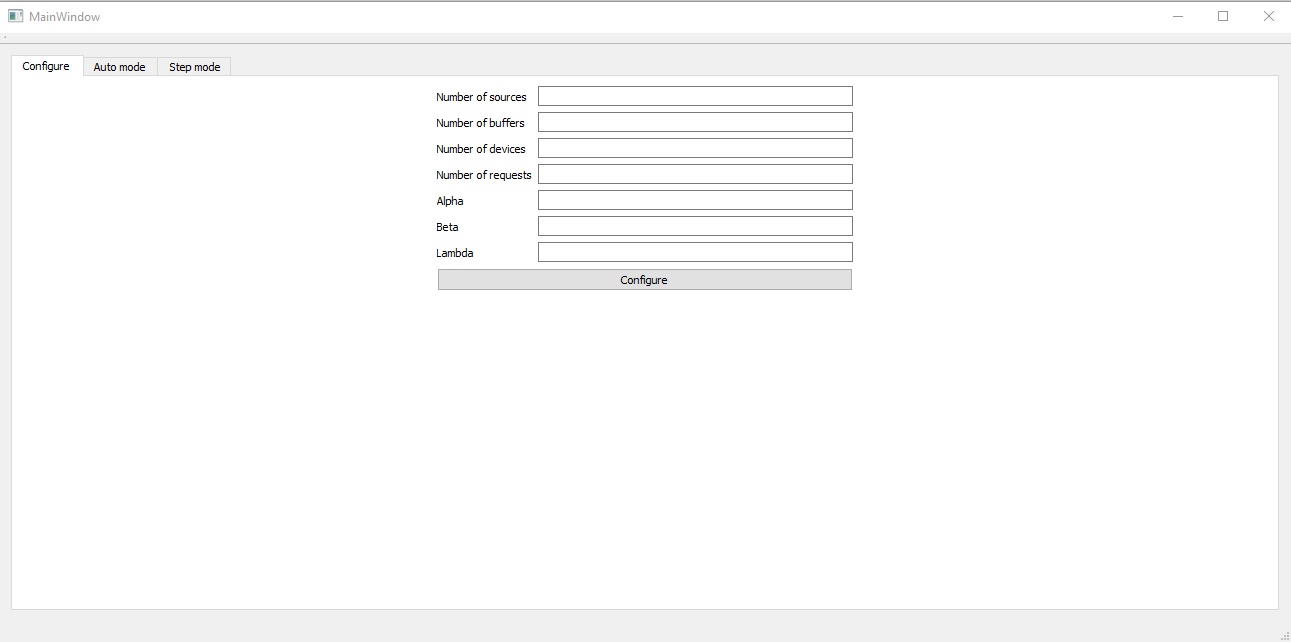
Разработка производилась в среде QT Creator 4.12.4, с использованием языка С++ и графической библиотеки QT.

Приложение является объектно-ориентированным и содержит следующие классы:

1. Request–содержит описание заявки. Реализует методы для получения этих описаний;
2. Source–реализует методы создания заявки и генерации времени заявки;
3. Buffer–реализует методы проверки свободного места в буфере, добавления заявки в буфер, выбора заявки из буфера;
4. Device–реализует методы проверки свободных приборов, загрузки заявки на прибор, удаления заявки из прибора;
5. Controller–реализует основной цикл работы системы;
6. Analytics – реализует сбор аналитики для пошагового и автоматического отображения;
7. Main Window–реализует графический интерфейс.

# **Описание работы программы**

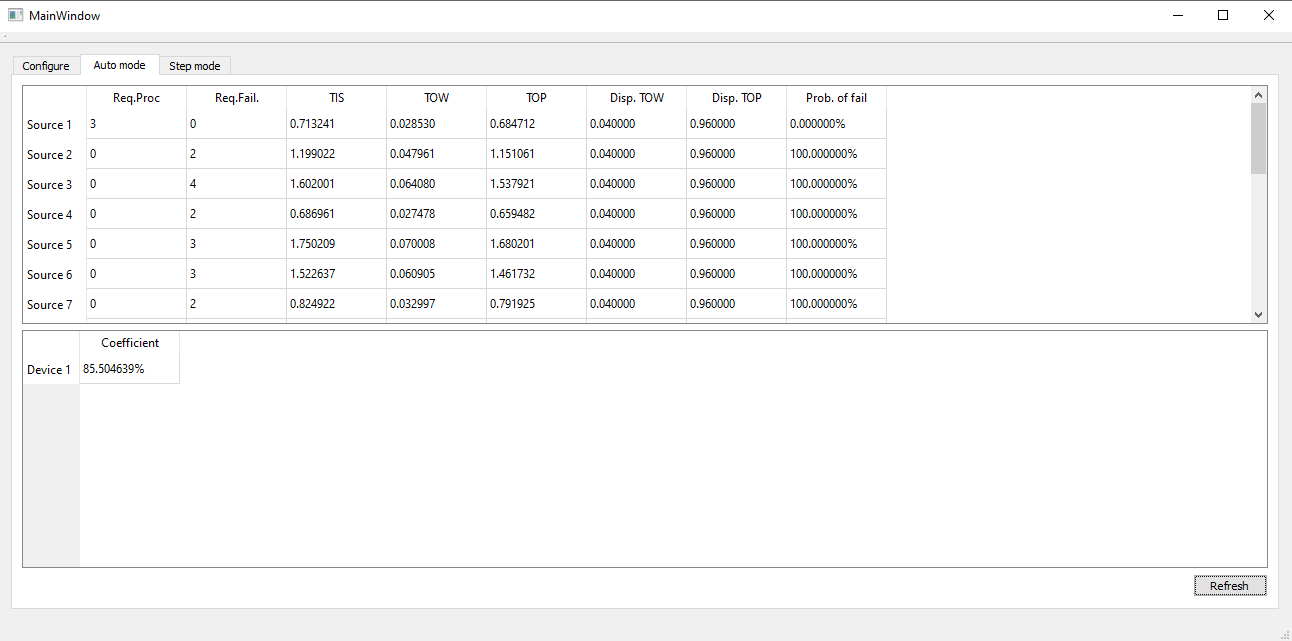
После запуска программы появляется следующее окно:



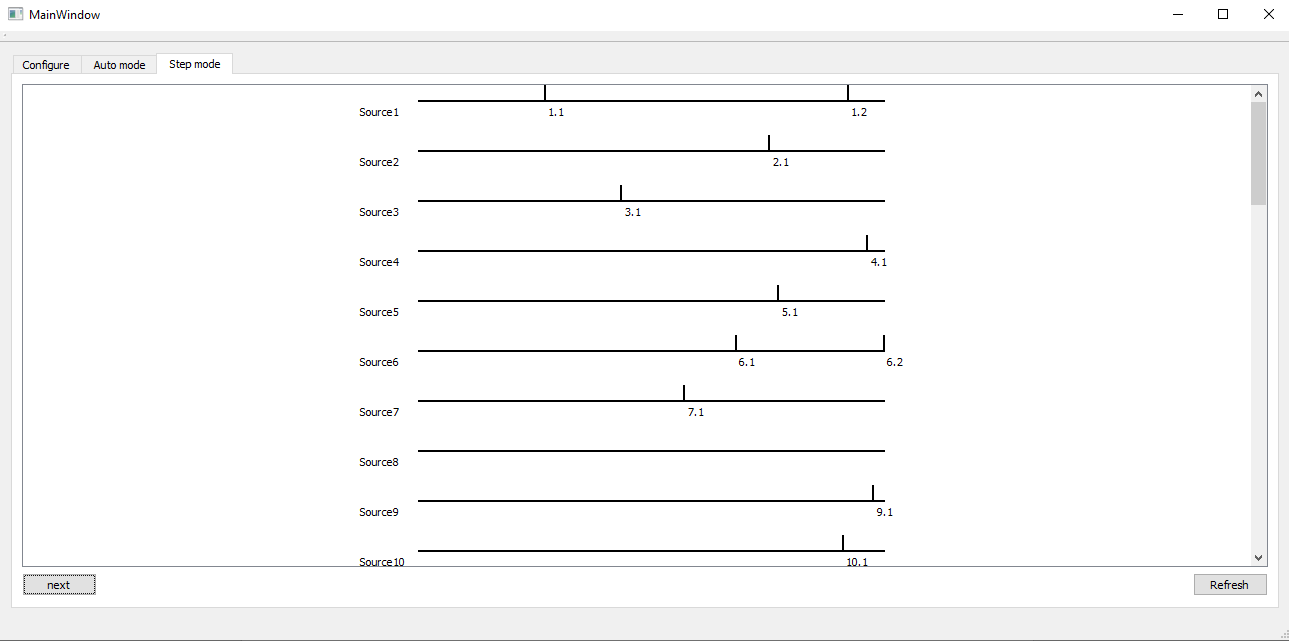
Мы находимся в главной части программы. Тут необходимо задать параметры для СМО и нажать кнопку «Configure» для того, чтобы сохранить параметры. Сверху мы видим вкладки «Auto mode» и «Step mode». Мы можем на них нажать и наблюдать результаты работы системы.

Рассмотрим пример отображения Auto mode:

При нажатии на кнопку Refresh у нас отображается последний результат работы.



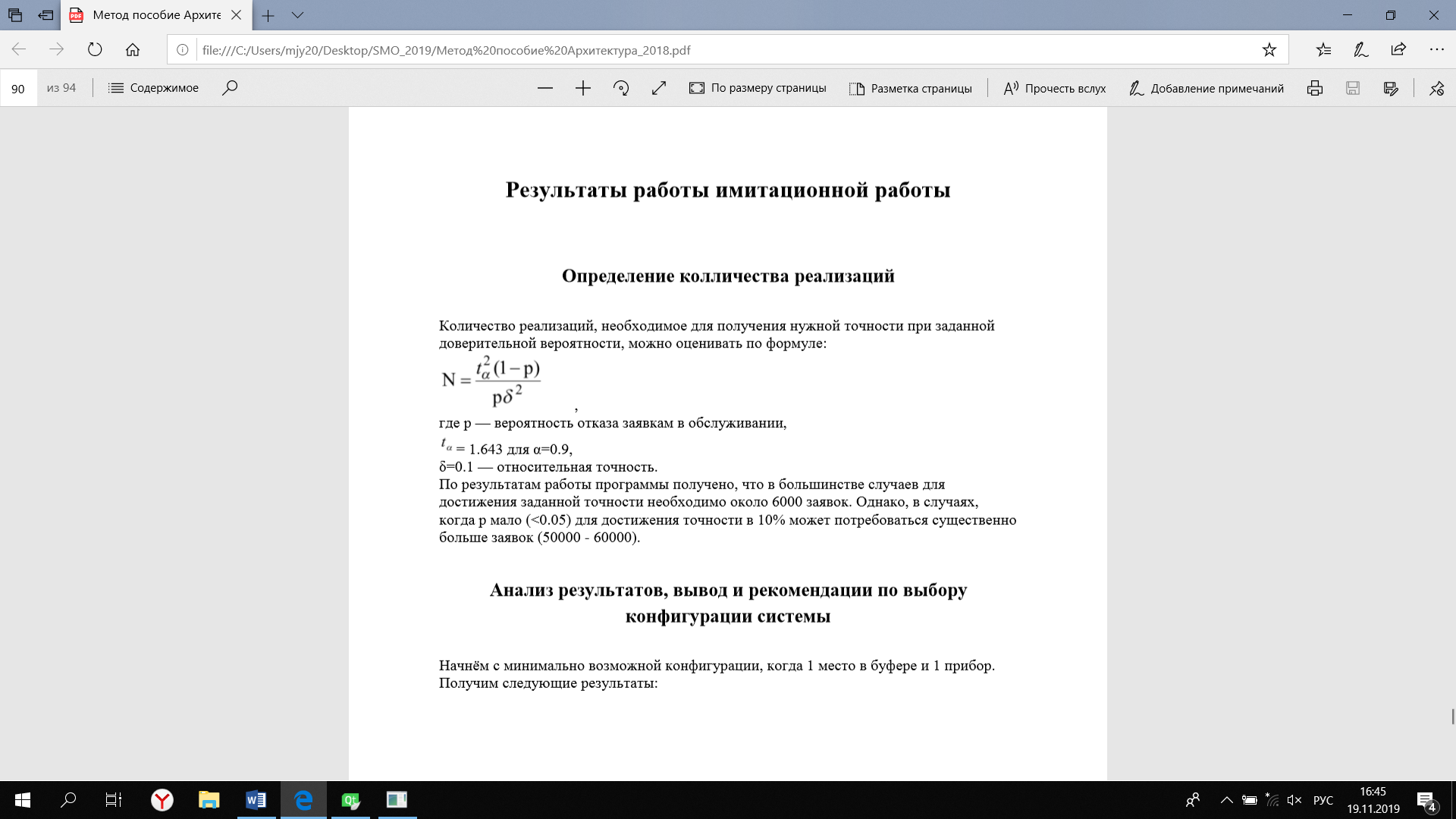
Пример отображения Step mode:



# **Результаты работы**

***Определение количества реализаций***

Количество реализаций, необходимое для получения нужной точности при заданной доверительной вероятности, можно оценивать по формуле:

, где p—вероятность отказа заявкам в обслуживании,

=1.643 дляα=0.9,

δ=0.1—относительная точность.

По результатам работы программы получено, что в большинстве случаев для достижения заданной точности необходимо около 6000 заявок. Однако, в случаях, когда p мало(<0.05) для достижения точности в 10% может потребоваться существенно больше заявок(50000-60000).

# **Анализ результатов**

Т. к. целью моделирования является выбор конфигурации системы, требующей наименьшее количество ресурсов и обрабатывающей максимальный поток информации, то начнем с проверки конфигурации с макс. числом источников и минимальным числом приборов и мин. размером буфера.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число источников | Размер буфера | Число приборов | Загруженность в % | Alpha | Beta | Lambda | P отк | T в сист. |
| 20 | 20 | 1 | 72 | 0.5 | 2.1 | 2.3 | 95% | 1.03 |
| 15 | 15 | 2 | 83 | 0.5 | 2.1 | 2.3 | 56% | 0.16 |
| 10 | 8 | 3 | 95 | 0.5 | 2.1 | 2.3 | 8.6% | 0.064 |

Из таблицы видно, что в последнем случае мы получили необходимую загруженность приборов, но вероятность отказов не удовлетворяет нашим условиям. Уменьшим интервал генерирования заявки источниками.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Число источников | Размер буфера | Число приборов | Загруженность в % | Alpha | Beta | Lambda | P отк | T в сист. |
| 10 | 8 | 3 | 88 | 1.1 | 2.1 | 2.3 | 0% | 0.03 |
| 10 | 8 | 3 | 91 | 1.1 | 1.9 | 2.3 | 0% | 0.06 |

Последний случай удовлетворяет всем условиям.

Если загруженность плохая, можно попробовать изменить три вещи:

* Увеличить количество источников;
* Уменьшить интервал генерирования заявки источниками;
* Заменить эти приборы на меньшее количество приборов с большей производительностью.

# **Вывод**

В ходе курсовой работы была написана система массового обслуживания на языке C++ с использованием графической библиотеки Qt. С помощью данной программы была проанализирована реальная система и подобрана максимально выгодная комплектация данной системы.