**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет**
**имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ
## по курсу «Моделирование»

«Моделирование работы информационного центра»

Студент: ИУ7-73Б _____ М. Д. Маслова
(группа) (подпись, дата) (И. О. Фамилия)

Преподаватель: _____ И. В. Рудаков
(подпись, дата) (И. О. Фамилия)

*2022 г.*

# СОДЕРЖАНИЕ

## 1 Задание

В театр приходят зрители группами по $3 \pm 2$ человека через интервалы времени $5 \pm 4$ секунды. Вход в театр осуществляется через две двери, в каждой из которых осуществляется проверка билетов и досмотр вещей, что занимает $30 \pm 15$ секунд на каждого человека. При этом $5\%$ из пришедших групп приобрели VIP-билеты и могут пройти необходимые этапы, войдя через отдельную третью дверь.

После проверки билетов и вещей каждый зритель сдает верхнюю одежду в гардероб, в котором принимают одежду $4$ гардеробщика. Каждый зритель может получить номерок за $7 \pm 3$ секунды. Отдельного гардероба для VIP-зрителей не предусмотрено.

Известно, что зрители пришедшие группой на входе встают в одну очередь, выбирая кратчайшую. В гардеробе отдельно каждый зритель, не зависимо от группы с которой он пришел, выбирает кратчайшую очередь.

Смоделировать процесс входа $600$ зрителей в театр. Определить время до начала представления, за которое необходимо начать пропуск зрителей.

# 2 Теоретическая часть

## 2.1 Схемы модели

На рисунке 2.1 представлена структурная схема модели.

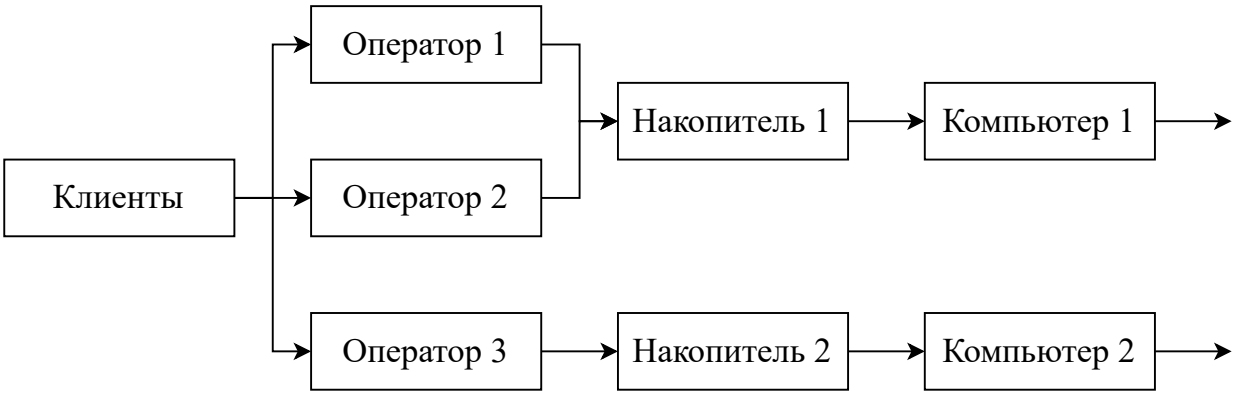


Рисунок 2.1 – Структурная схема модели

В процессе взаимодействия клиентов с информационным центром возможно два режима работы:

- режим нормального обслуживания, когда клиент выбирает одного из свободных операторов, отдавая предпочтение тому, у кого максимальная производительность;
- режим отказа клиенту в обслуживании, когда все операторы заняты.

На рисунке 2.2 представлена схема модели в терминах систем массового обслуживания (СМО).

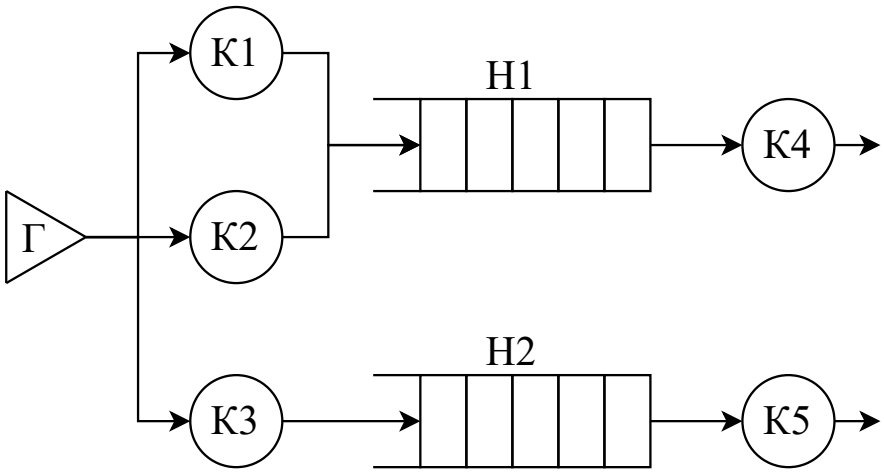


Рисунок 2.2 – Схема модели в терминах СМО

## 2.2 Переменные и уравнение имитационной модели

Эндогенные переменные:

– время обработки задания $i$-ым оператором;
– время решения задания на $j$-ом компьютере.

Экзогенные переменные:

– $n_0$ — число обслуженных клиентов;
– $n_1$ — число клиентов, получивших отказ.

Вероятность отказа рассчитывается по формуле 2.1, которая описывает уравнение модели:

$$P_{\text{отказа}} = \frac{n_0}{n_0 + n_1}. \tag{2.1}$$

# 3 Практическая часть

## 3.1 Текст программы

На листинге 3.1-3.2 представлены реализации генератора и канала обслуживания соответственно.

Листинг 3.1 — Реализация генератора

```
class Generator:
    def __init__(self, distribution, receivers):
        self.distribution = distribution
        self.receivers = receivers
        self.nextEvent = Event(-1, self)

    def GenerateNextEvent(self, curTime):
        self.nextEvent.Time = curTime + self.distribution.Generate()

    def TransmitRequest(self):
        for receiver in self.receivers:
            if receiver.TakeRequest(self.nextEvent.time):
                return True
        return False
```

Листинг 3.2 — Реализация канала обслуживания

```
class Processor(Generator):
    def __init__(self, generator: Generator, memory: Memory):
        super().__init__(generator.distribution, generator.receivers)
        self.nextEvent.eventBlock = self
        self.memory = memory
        self.aviable = True

    def TakeRequest(self, curTime) -> bool:
        if self.aviable:
            self.aviable = False
            self.GenerateNextEvent(curTime)
            return True
        return self.memory.InsertRequest()

    def EndProcess(self, curTime):
        self.TransmitRequest()
        if not self.memory.IsEmpty():
            self.memory.RemoveRequest()
            self.GenerateNextEvent(curTime)
        else:
            self.aviable = True
            self.NextEvent.Time = -1
```

На листинге 3.3 реализация моделирования работы информационного центра.

Листинг 3.3 — Реализация моделирования работы информационного центра

```
class EventModel:
    def __init__(self, generator: Generator, operators: list[Processor]
                 , computers: list[Processor], requestsNum=1000):
        self.generator = generator
        self.operators = operators
        self.computers = computers
        self.blocks = [self.generator] + self.operators + self.computers
        self.requestsNum = requestsNum

    def run(self):
        self.generator.GenerateNextEvent(0)
        events = [block.NextEvent for block in self.blocks]

        generatedRequests = 1
        denials = 0

        while generatedRequests < self.requestsNum:
            curTime = events[0].Time
            for event in events[1:]:
                if not event.Time < 0 and event.Time < curTime:
                    curTime = event.Time

            for block in self.blocks:
                if abs(block.NextEvent.Time - curTime) < EPS:
                    if not isinstance(block.NextEvent.EventBlock,
                     ↪  Processor):
                        generatedRequests += 1
                        if not block.TransmitRequest():
                            denials += 1
                        block.GenerateNextEvent(curTime)
                    else:
                        block.EndProcess(curTime)

        return denials / generatedRequests
```

## 3.2 Полученный результат

На рисунке 3.1 приведен примеры работы программы на значениях, данных в условии.

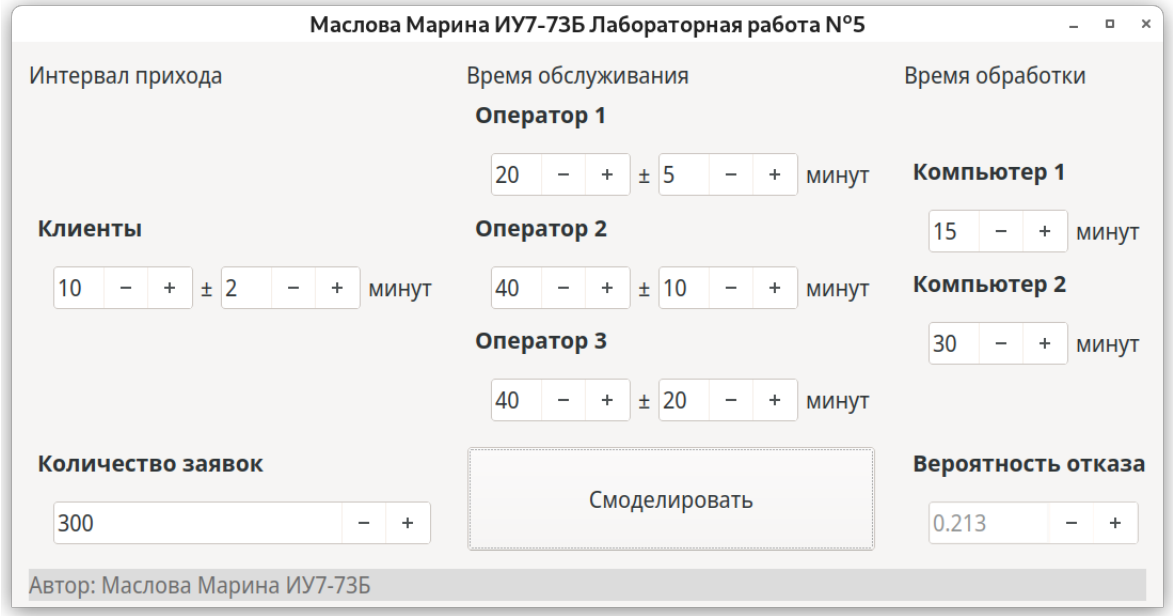


Рисунок 3.1 – Пример работы программы