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 №6

по курсу «Моделирование»

на тему: «Моделирование работы билетно-кассового РЖД центра»

Студент ИУ7-72Б _____ Т. М. Сучкова
(Группа) (Подпись, дата) (И. О. Фамилия)

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

2022 г.

1 Задание

В билетно-кассовый РЖД центр приходят клиенты через интервал времени $0-3$ мин. Данный центр содержит 4 оператора: 2 терминала, причем один из терминалов — нового формата, другой — старого, и 2 кассы. Если все операторы заняты, клиент встает в очередь. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 3 ± 1; 4 ± 2; 5 ± 3; 8 ± 4 мин. Клиенты стремятся занять оператора, длина очереди к которому минимальна. Полученные запросы сдаются в приемный накопитель. Откуда выбираются на обработку. На сервере находятся компьютеры, которые обработывают данные запросы. На первый компьютер — запросы от 1-ого оператора, на второй — запросы от 2-ого, на третий — запросы от 3-его и 4-ого операторов. Время обработки запросов на 1-ом, 2-ом и 3-ем компьютерах равны соответственно 1, 2, 3 мин.

Смоделировать процесс обработки 600 запросов.

Также в отчете необходимо сделать следующее.

1. Построить структурную схему модели.
2. Построить СМО модель.

2 Теоретическая часть

Структурная схема модели приведена на рисунке 2.1.

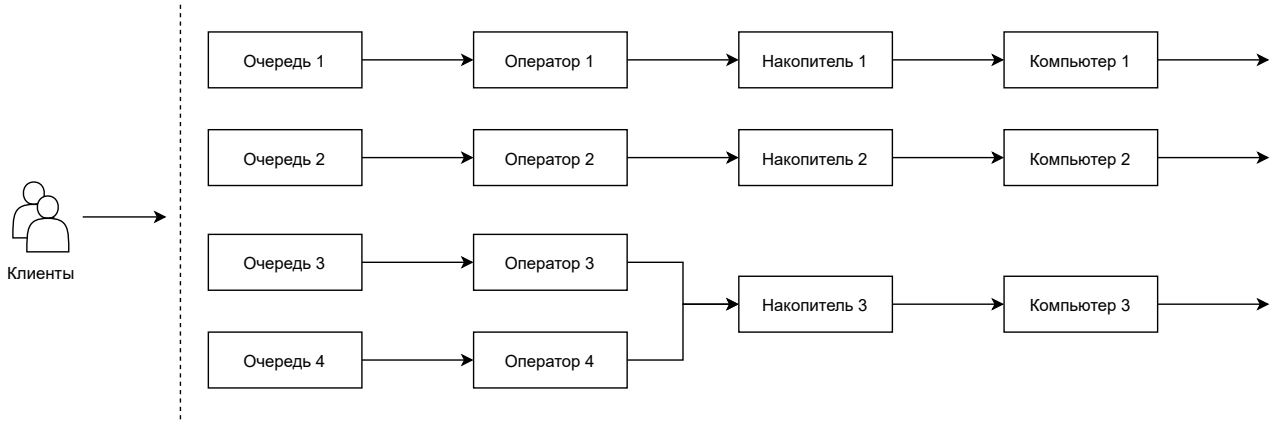


Рисунок 2.1 – Структурная схема модели

Схема модели в терминах СМО приведена на рисунке 2.2.

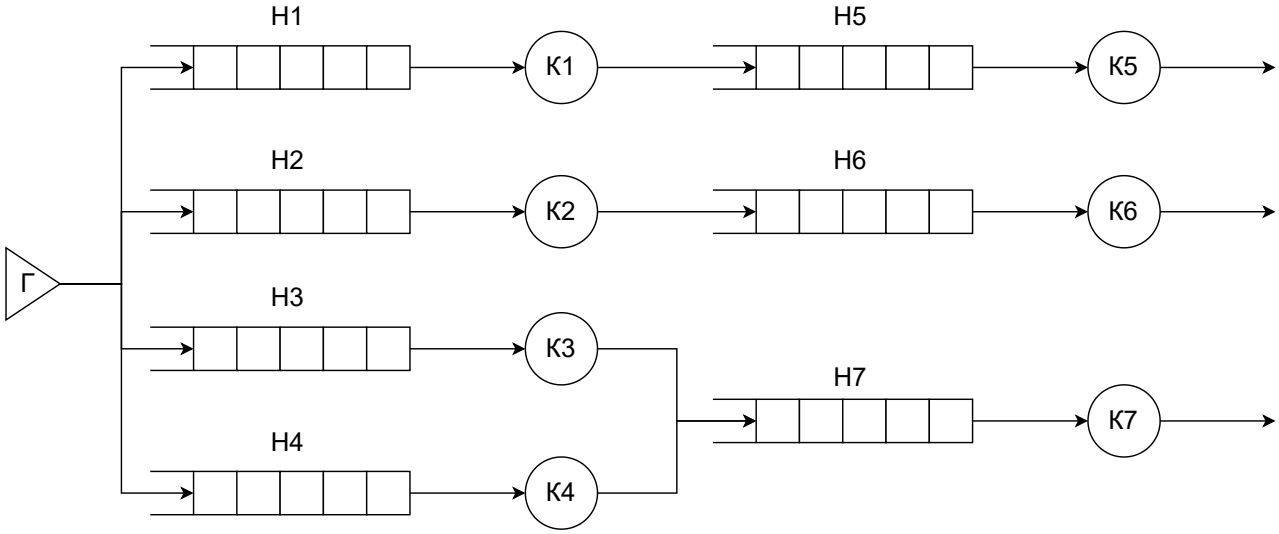


Рисунок 2.2 – Схема модели в терминах СМО

2.1 Переменные

Эндогенные переменные:

— время обработки задания i-ым оператором;

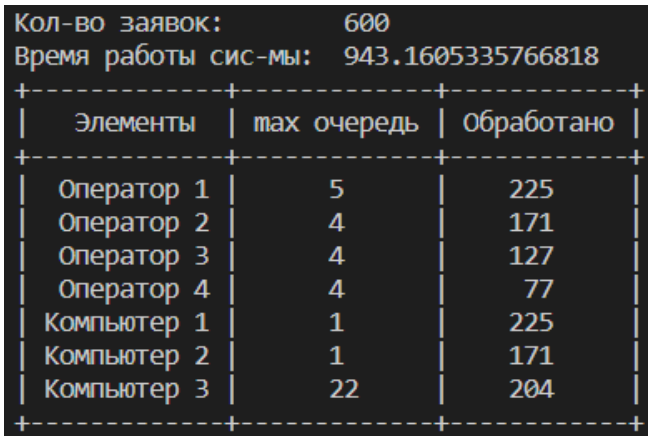
— время решения этого задания j-ым компьютером.

Экзогенные переменные:

— n_0 — число обслуженных клиентов.

3 Результат

Результаты работы программы представлены на рисунке 3.1.



```
Кол-во заявок:         600
Время работы сис-мы:   943.1605335766818
+-------------+-------------+------------+
|   Элементы  | max очередь | Обработано |
+-------------+-------------+------------+
|  Оператор 1 |      5      |    225     |
|  Оператор 2 |      4      |    171     |
|  Оператор 3 |      4      |    127     |
|  Оператор 4 |      4      |     77     |
| Компьютер 1 |      1      |    225     |
| Компьютер 2 |      1      |    171     |
| Компьютер 3 |      22     |    204     |
+-------------+-------------+------------+
```

Рисунок 3.1 – Пример работы программы

4 Код программы

В листингах 4.1–4.5 представлен основной код программы.

Листинг 4.1 – Класс распределения

```
import random as rand


class UniformDistribution:
    def __init__(self, a: float, b: float):
        self.a = a
        self.b = b

    def generate(self):
        return self.a + (self.b - self.a) * rand.random()
```

Листинг 4.2 – Класс генератора

```
class Generator:
    def __init__(self, generator, count: int):
        self._generator = generator
        self.receivers = []
        self.num_requests = count
        self.next = 0

    def next_time(self):
        return self._generator.generate()

    def generate_request(self):
        self.num_requests -= 1
        min_receiver = self.receivers[0]
        min_q_size = self.receivers[0].curr_q_size

        for i in range(1, len(self.receivers)):
            curr_r = self.receivers[i]
            if curr_r.curr_q_size < min_q_size:
                min_q_size = curr_r.curr_q_size
                min_receiver = curr_r

        min_receiver.receive_request()

        return min_receiver
```

Листинг 4.3 – Класс процессора

```
from Generator import Generator

class Processor(Generator):
    def __init__(self, generator):
        self._generator = generator
        self.curr_q_size = 0
        self.max_size = 0
        self.processed_requests = 0
        self.received_requests  = 0
        self.next = 0
        self.receivers = []

    # обрабатываем запрос, если они есть
    def process_request(self):
        if self.curr_q_size > 0:
            self.processed_requests += 1
            self.curr_q_size -= 1

        if len(self.receivers) != 0:
            min_receiver = self.receivers[0]
            min_q_size = self.receivers[0].curr_q_size

            for receiver in self.receivers:
                if receiver.curr_q_size < min_q_size:
                    min_q_size = receiver.curr_q_size
                    min_receiver = receiver

            min_receiver.receive_request()
            min_receiver.next  = self.next + min_receiver.next_time
               ()

    # добавляем реквест в очередь
    def receive_request(self):
        self.curr_q_size += 1
        self.received_requests += 1

        if self.max_size < self.curr_q_size:
            self.max_size = self.curr_q_size

        return True
```

```
40
41
42      def next_time(self):
43          return self._generator.generate()
```

Листинг 4.4 – Класс модели

```
1  from Generator import Generator
2  from Processor import Processor
3
4  class Model:
5      def __init__(self, generator: Generator, operators: list[
           Processor], computers: list[Processor]):
6          self._generator = generator
7          self._operators = operators
8          self._computers = computers
9
10     # n0 — число обслуженных клиентов
11     def start_event(self) -> dict[str, float]:
12         generator = self._generator
13         n0 = 0
14         n1 = 0
15
16         generator.receivers = self._operators.copy()
17         self._operators[0].receivers = [self._computers[0]]
18         self._operators[1].receivers = [self._computers[1]]
19         self._operators[2].receivers = [self._computers[2]]
20         self._operators[3].receivers = [self._computers[2]]
21
22         generator.next = generator.next_time()
23         self._operators[0].next = self._operators[0].next_time()
24
25         blocks = [ generator ] + self._operators + self._computers
26
27         n_done = 0
28         n_requests = generator.num_requests
29
30         while n_done < n_requests:
31             # находим наименьшее время
32             curr_t = generator.next
33             for block in blocks:
34                 if 0 < block.next < curr_t:
35                     curr_t = block.next
```

```
36
37              # для каждого из блоков
38              for block in blocks:
39                  # если событие наступило для этого блока
40                  if curr_t == block.next:
41                      if not isinstance(block, Processor):
42                          # для генератора
43                          # проверяем, может ли оператор обработать
44                          next_generator = generator.generate_request
                                ()
45                          if next_generator is not None:
46                              next_generator.next = \
47                                  curr_t + next_generator.next_time()
48                              n0 += 1
49                          else:
50                              n1 += 1
51                          generator.next = curr_t + generator.
                                next_time()
52                      else:
53                          block.process_request()
54                          if block.curr_q_size == 0:
55                              block.next = 0
56                          else:
57                              block.next = curr_t + block.next_time()
58              n_done = 0
59              for item in self._computers:
60                  n_done += item.processed_requests
61
62          max_q_size = []
63
64          for item in self._operators:
65              max_q_size.append(item.max_size)
66          for item in self._computers:
67              max_q_size.append(item.max_size)
68
69          done_arr = []
70
71          for item in self._operators:
72              done_arr.append(item.processed_requests)
73          for item in self._computers:
74              done_arr.append(item.processed_requests)
```

```
75
76          return {"max_q_size": max_q_size,
77                  "time": curr_t,
78                  "done": n_done,
79                  "done_arr": done_arr,
80                  "arrived": n0
81                  }
```

Листинг 4.5 – Основная функция

```
1  from Model import Model
2  from Generator import Generator
3  from Distribute import UniformDistribution
4  from Processor import Processor
5
6  from prettytable import PrettyTable
7
8
9  def create_operators(distribution, t_proc: list[list[int]], n: int)
       -> list[Processor]:
10     operators = list()
11
12     for i in range(n):
13         operators.append(Processor(
14                     distribution(t_proc[i][0] - t_proc[i][1],
                           t_proc[i][0] + t_proc[i][1]))
15                     )
16     return operators
17
18 def create_computers(distribution, t_proc: list[int], n: int) ->
       list[Processor]:
19     comps = list()
20
21     for i in range(n):
22         comps.append(Processor(
23                     distribution(t_proc[i], t_proc[i])) )
24     return comps
25
26
27 def create_table(data: dict[str, float], operators_n: int,
       computers_n: int, clients_number: int):
28     table = PrettyTable()
29     table.add_column('Элементы', [('Оператор '+ str(i + 1)) for i
```

```
        in range(operators_n)] + [('Компьютер ' + str(i + 1)) for i
        in range(computers_n)])
30      table.add_column('max очередь', data['max_q_size'])
31      table.add_column('Обработано', data['done_arr'])
32
33      print("Кол-во заявок:       ", clients_number)
34      print("Время работы сис-мы:  " + str(data['time']))
35      print(table)
36
37
38  if __name__ == '__main__':
39      clients_number = 600
40
41      generator = Generator(
42          UniformDistribution(0, 3),
43          clients_number,
44      )
45
46      t_proc_op = list()
47      t_proc_op.append([3, 1]) # t_i, dt_i
48      t_proc_op.append([4, 2])
49      t_proc_op.append([5, 3])
50      t_proc_op.append([8, 4])
51
52      operators = create_operators(UniformDistribution, t_proc_op,
        len(t_proc_op))
53
54      t_proc_comp = [1, 2, 3]
55
56      computers = create_computers(UniformDistribution, t_proc_comp,
        len(t_proc_comp))
57
58      model = Model(generator, operators, computers)
59      res = model.start_event()
60
61      create_table(res, len(operators), len(computers),
        clients_number)
```