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 дисциплине "Моделирование"

Тема Моделирование работы супермаркета

Студент Малышев И. А.

Группа ИУ7-71Б

Оценка (баллы) _____

Преподаватель: Рудаков И. В.

Москва — 2022 г.

1 | Задание

Реализовать программу для моделирования следующей системы: в супермаркете покупатели приходят к кассам с заданным интервалом времени. У каждой кассы формируется своя очередь. Клиент выбирает очередь с минимальной длинной. Кассиры обслуживают клиентов за заданный интервал времени. После того, как все товары отсканированы, клиенту необходимо оплатить товар. У каждого терминала оплаты формируется своя очередь. Клиент выбирает терминал с очередью наименьшей длины. Терминалы обслуживают клиентов за фиксированный интервал времени. Количество клиентов задается.

2 | Решение

2.1 Теоретическая часть

2.1.1 Структурная схема системы в терминах СМО

Структурная схема в терминах СМО представлена на рисунке 2.1.

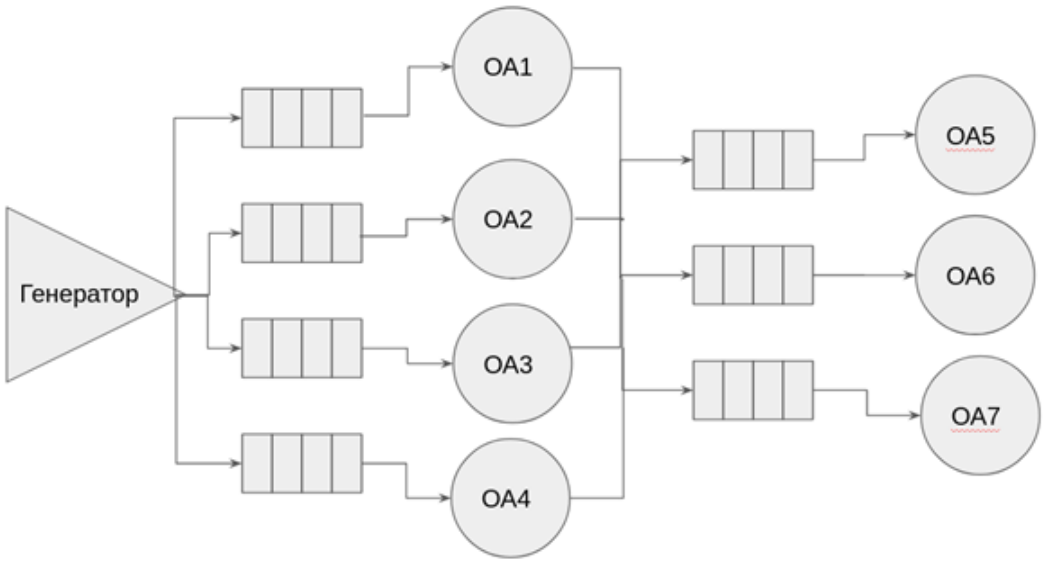


Рис. 2.1: Структурная схема системы в терминах СМО.

2.2 Листинг

Далее представлен фрагмент программы, выполняющий поставленное задание.

```
1 from numpy import random as nr
2
3 class Generator:
4   def __init__(self, generator, count):
5     self._generator = generator
6     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
    receiver_min = self.receivers[0];
    min = self.receivers[0].current_queue_size;
    for receiver in self.receivers:
      if receiver.current_queue_size < min:
        min = receiver.current_queue_size
        receiver_min = receiver
    receiver_min.receive_request()
    return receiver_min

class Processor(Generator):
  def __init__(self, generator, max_queue=-1):
    self._generator = generator
    self.current_queue_size = 0
    self.max_queue_size = max_queue
    self.max_size = 0
    self.processed_requests = 0
    self.received_requests  = 0
    self.next = 0
    self.receivers = []

  def process_request(self):
    if self.current_queue_size > 0:
      self.processed_requests += 1
      self.current_queue_size -= 1

    if len(self.receivers) != 0:
      receiver_min = self.receivers[0];
      min = self.receivers[0].current_queue_size;
      for receiver in self.receivers:
        if receiver.current_queue_size < min:
          min = receiver.current_queue_size
          receiver_min = receiver
      receiver_min.receive_request()
      receiver_min.next  = self.next + receiver_min.next_time()

  def receive_request(self):
    self.current_queue_size += 1
    self.received_requests += 1
    if self.max_size < self.current_queue_size:
      self.max_size = self.current_queue_size
    return True

```

```
57    def next_time(self):
58      return self._generator.generate()
59
60  class Modeller:
61    def __init__(self, generator, operators, computers):
62      self._generator = generator
63      self._operators = operators
64      self._computers = computers
65
66    def event_mode(self):
67      refusals = 0
68      processed = 0
69      generated_requests = self._generator.num_requests
70      generator = self._generator
71
72      generator.next = generator.next_time()
73      self._operators[0].next = self._operators[0].next_time()
74
75      blocks = [
76        generator
77      ] + self._computers + self._operators
78
79      num_requests = generator.num_requests
80      count = 0;
81      while count < num_requests:
82        my_str = 'iter '
83        for oper in self._operators:
84        my_str += str(oper.current_queue_size) + ' '
85        my_str += "|||||"
86        for oper in self._computers:
87        my_str += str(oper.current_queue_size) + ' '
88        my_str += "|||||"
89        for oper in self._computers:
90        my_str += str(oper.processed_requests) + ' '
91        my_str += "|||||"
92        for oper in self._operators:
93        my_str += str(oper.processed_requests) + ' '
94
95        print(my_str)
96        current_time = generator.next
97        for block in blocks:
98          if 0 < block.next < current_time:
99            current_time = block.next
100
101
102       for block in blocks:
103         if current_time == block.next:
104           if not isinstance(block, Processor):
105             next_generator = generator.generate_request()
106             if next_generator is not None:
```

```
107                 next_generator.next = \
108                 current_time + next_generator.next_time()
109                 processed += 1
110               else:
111                 refusals += 1
112               generator.next = current_time + generator.next_time()
113           else:
114             block.process_request()
115             if block.current_queue_size == 0:
116               block.next = 0
117             else:
118               block.next = current_time + block.next_time()
119         count = 0
120         for oper in self._computers:
121         count += oper.processed_requests
122 
123       max_queue = []
124       for oper in self._operators:
125         max_queue.append(oper.max_size)
126       for oper in self._computers:
127         max_queue.append(oper.max_size)
128       processed_arr = []
129       for oper in self._operators:
130         processed_arr.append(oper.processed_requests)
131       for oper in self._computers:
132         processed_arr.append(oper.processed_requests)
133       return {
134         "max_queue": max_queue,
135         "time": current_time,
136         "processed": count,
137         "proc_arr": processed_arr,
138         "pribilo": processed
139       }
```

2.3 Результаты работы

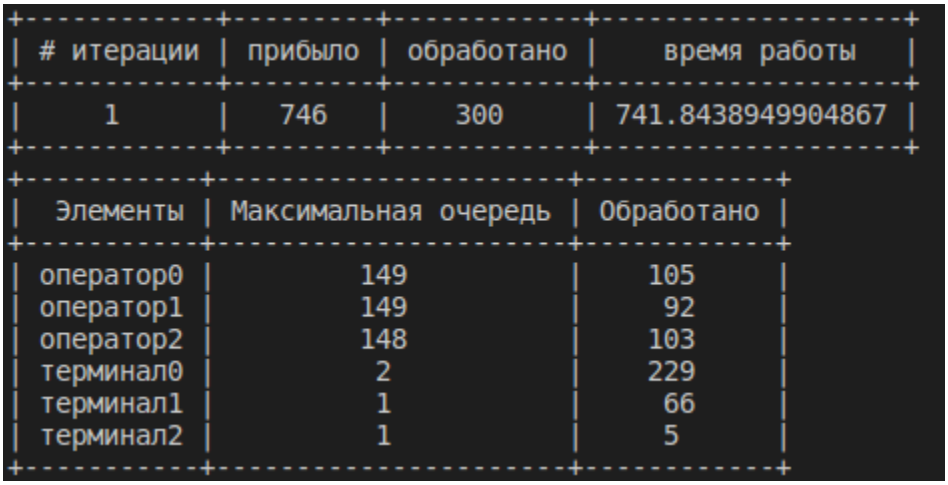
Промоделируем работу системы для 300 клиентов, 3-х касс и 3-х терминалов. Клиенты приходят с интервалом 0-2 минуты, кассир обрабатывает клиента за 1-7 минут, а терминал за 0-2 минуты.

Входные параметры:

- Количество клиентов: 300
- Время обслуживания кассира: 4
- Погрешность обслуживания кассира: 3

- Время обслуживания терминала: 1
- Погрешность обслуживания терминала: 1
- Время прихода клиента: 1
- Погрешность времени прихода клиента: 1

Результаты работы программы представленны на рисунке 2.2



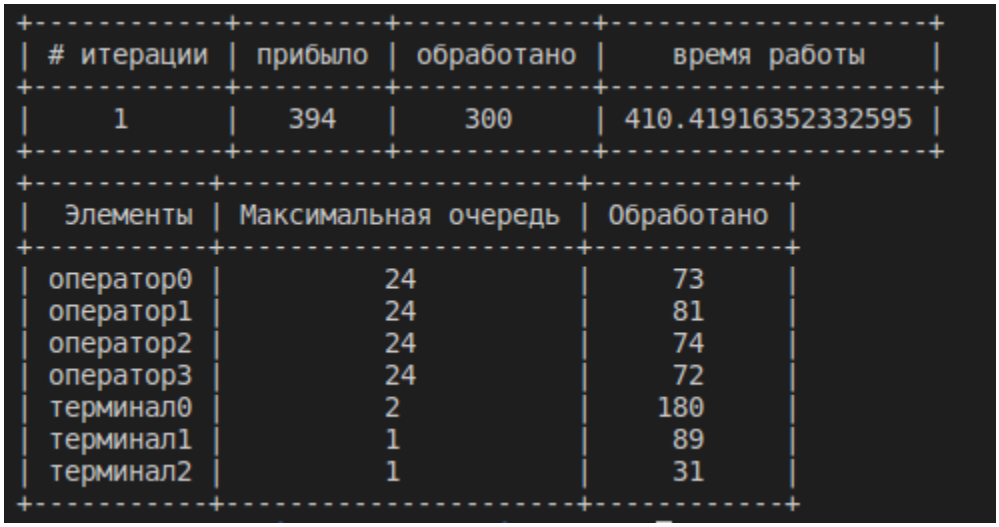
# итерации	прибыло	обработано	время работы
1	746	300	741.8438949904867

Элементы	Максимальная очередь	Обработано
оператор0	149	105
оператор1	149	92
оператор2	148	103
терминал0	2	229
терминал1	1	66
терминал2	1	5

Рис. 2.2: Работа программы с тремя операторами и тремя терминалами.

Как видно из результатов, операторы не успеваю обработать всех прибывших клиентов, поэтом образуются длинные очереди.

Добавим еще одного оператора и запустим с теми же параметрами. Результаты работы программы представлены на рисунке 2.3



# итерации	прибыло	обработано	время работы
1	394	300	410.41916352332595

Элементы	Максимальная очередь	Обработано
оператор0	24	73
оператор1	24	81
оператор2	24	74
оператор3	24	72
терминал0	2	180
терминал1	1	89
терминал2	1	31

Рис. 2.3: Работа программы с четыремя операторами и тремя терминалами.

Ситуация становится лучше, но очереди все еще довольно большие.

Добавим еще двух операторов. Результаты работы программы представлена на рисунке 2.4

# итерации	прибыло	обработано	время работы
1	305	300	303.07971799421796

Элементы	Максимальная очередь	Обработано
оператор0	3	61
оператор1	2	61
оператор2	2	59
оператор3	2	50
оператор4	2	41
оператор5	2	28
терминал0	2	158
терминал1	2	102
терминал2	1	40

Рис. 2.4: Результаты работы для 6ти операторов и 3х терминалов.

Как видно из рисунка 4 теперь очереди составляют не больше 3х человек, что приемлимо. 300 человек обслуживаются за 5 часов.

Также для уменьшения очередей можно улучшить производительность работы операторов. Путь оператор тратит на обслуживание клиента 1-5 минут. А всего операторов 4. Результат работы программы преставлен на рисунке 2.5

# итерации	прибыло	обработано	время работы
1	313	300	308.28419596172245

Элементы	Максимальная очередь	Обработано
оператор0	6	78
оператор1	6	80
оператор2	5	72
оператор3	5	71
терминал0	2	164
терминал1	2	96
терминал2	1	40

Рис. 2.5: Работа программы при 4х операторах и трех терминалах с улучшенной эффективностью операторов.