

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	Т «Информатика и системы управления»						
КАФЕДРА _	«Программное обеспечение ЭВМ и информационные технологии»						

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ по курсу «Моделирование»

«Моделирование работы системы массового обслуживания»

Студент:	_ИУ7-73Б_		М. Д. Маслова			
	(группа)	(подпись, дата)	(И. О. Фамилия)			
Преподаватель:			И. В. Рудаков			
		(подпись, дата)	(И. О. Фамилия)			

СОДЕРЖАНИЕ

1	Зада	ание .		. 4
2	Teo	ретиче	еская часть	. 5
	2.1	Испол	льзуемые распределения	. 5
		2.1.1	Равномерное распределение	. 5
		2.1.2	Нормальное распределение	. 5
	2.2	Описа	ание принципов	. 6
		2.2.1	Пошаговый принцип	. 6
		2.2.2	Событийный принцип	. 6
3	Пра	ктиче	еская часть	. 7
			программы	
	3 2	Попуц	ченный пезультат	8

1 Задание

Разработать программное обеспечение, предоставляющее возможность промоделировать систему, состоящую из генератора, буферной памяти и обслуживающего аппарата, пошаговым и событийным принципами. Генератор выдает сообщения по равномерному закону, обслуживающий аппарат обрабатывает их по нормальному закону. С определенной долей вероятности часть обработанных сообщений снова поступают в очередь. Определить размер буферной памяти, при котором не будет потерь сообщений.

2 Теоретическая часть

2.1 Используемые распределения

2.1.1 Равномерное распределение

Случайная величина X имеет *равномерное распределение* на отрезке [a, b], если ее плотность распределения f(x) равна:

$$p(x) = \begin{cases} \frac{1}{b-a}, & \text{если } a \le x \le b; \\ 0, & \text{иначе.} \end{cases}$$
 (2.1)

Обозначение: $X \sim R[a, b]$.

Момент времени t_i может быть вычислен по следующей формуле:

$$t_i = a + (b - a)R, (2.2)$$

где $R \in [0,1]$ — равномерно распределенная случайная величина в промежутке [0,1].

2.1.2 Нормальное распределение

Случайная величина X имеет *нормальное распределение* с параметрами m и σ , если ее плотность распределения f(x) равна:

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{-\frac{(x-m)^2}{2\sigma^2}}, \quad x \in \mathbb{R}, \sigma > 0.$$
 (2.3)

Обозначение: $X \sim N(m, \sigma^2)$.

Момент времени t_i может быть вычислен по следующей формуле:

$$t_i = \sigma \sqrt{\frac{12}{n}} \left(\sum_{i=1}^n R_i - \frac{n}{2} \right) + m, \tag{2.4}$$

где $R_i \in [0,1]$ — равномерно распределенные случайные величины в промежутке [0,1].

2.2 Описание принципов

2.2.1 Пошаговый принцип

Пошаговый принцип или принцип Δt заключается в последовательном анализе состояний всех блоков в момент времени $t+\Delta t$ по заданному состоянию блоков в момент времени t. При этом новое состояние блоков определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов. В результате этого анализа принимается решение о том, какие общесистемные события должны имитироваться программой на данный момент времени.

Основной недостаток принципа Δt заключается в значительных затратах вычислительных ресурсов, а при недостаточно малом Δt появляется опасность пропуска отдельных событий в системе, исключающая возможность получения правильных результатов при моделировании.

2.2.2 Событийный принцип

Состояния отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщений в систему, окончания реализации задания, поэтому моделирование и продвижение текущего времени в системе удобно проводить, используя событийных принцип.

При использовании данного принципа состояние всех блоков имитационной модели анализируется лишь в момент появления какого-либо события. Момент наступления следующего события определяется минимальными значениями из списка будущих событий, представляющего собой совокупность моментов ближайшего изменения состояний каждого из блока системы.

3 Практическая часть

3.1 Текст программы

На листинге 3.1 представлена реализация принципа Δt .

Листинг 3.1 — Реализация пошагового принципа

```
class StepModel:
1
2
         def run(self):
 3
              self.processor.set_aviable(True)
 4
              processed_requests = 0
 5
              total_requests = self.requests_num
 6
              generator_time = self.generator.next_time()
 7
8
              processor\_time = -1
 9
              empty_generated = None
10
              current_time = self.step
11
              while processed requests < total requests:</pre>
12
                  if generator_time < current_time:</pre>
13
                      if self.processor.is_aviable() and
14

    self.memory.is empty():

15
                           empty_generated = True
                      self.memory.insert_request()
16
                      saved_time = generator_time
17
18
                      generator_time = saved_time + self.generator.next_time()
19
                  if 0 < processor_time < current_time:</pre>
20
                      processed\_requests += 1
21
                      if random.randint(0, 100) < self.repeat_percent:</pre>
22
                           self.memory.insert_request()
23
                      self.processor.set_aviable(True)
24
25
                  if self.processor.is_aviable():
26
                      if not self.memory.is empty():
27
                           self.memory.remove_request()
28
                           self.processor.set_aviable(False)
29
                           processor_time = ((saved_time if empty_generated
30
                                                else processor_time)
31
                                                  + self.processor.process_time())
32
                      else:
33
                           processor\_time = -1
34
35
                  empty generated = False
36
                  current_time += self.step
37
38
39
              return self.memory.max len
```

На листинге 3.2 представлена реализация событийного принципа.

Листинг 3.2 — Реализация событийного принципа

```
class EventModel:
 1
 2
         def run(self):
 3
              self.processor.set aviable(True)
 5
             processed_requests = 0
 6
              total requests = self.requests num
 7
              events = FutureEvents()
 8
 9
              events.add(Event(self.generator.next time(),
10
                 EventType.GENERATOR))
11
             while processed_requests < total_requests:</pre>
12
                  cur_event = events.next()
13
14
                  if cur_event.event_type == EventType.GENERATOR:
15
                      self.memory.insert_request()
16
                      events.add(Event(cur_event.time +
17

→ self.generator.next_time(),
                                        EventType.GENERATOR))
18
19
                  if cur_event.event_type == EventType.PROCESSOR:
20
                      processed requests += 1
21
                      if random.randint(0, 100) < self.repeat_percent:</pre>
22
                          self.memory.insert_request()
23
                      self.processor.set aviable(True)
24
25
                  if self.processor.is_aviable():
26
                      if not self.memory.is empty():
                          self.memory.remove_request()
28
                          self.processor.set_aviable(False)
29
                          events.add(Event(cur_event.time +
30
                                             self.processor.process_time(),
31
                                             EventType.PROCESSOR))
32
33
34
              return self.memory.max_len
```

3.2 Полученный результат

На рисунках 3.1-3.5 приведены примеры работы программы для каждого принципа с вероятностями повторной обработки сообщения 0.0, 0.25, 0.50, 0.75, 0.99 соответственно.

Маслова М	Марина ИУ7	-73Б Лаборатор	ная рабо	ота N	l°4			-	0	×
Параметры генератора		Другие параметры								
- 100 · b 500		Вероятность повторной обработки				0.00	_		+	
a 1.00 - + b 5.00	- +	Число заявок	1000	_	+	Δt	1.00	_	ŀ	+
Параметры обслуживающего аппарата Результат моделирования										
m 1.00 - + σ 0.10	- +	Пошаговый принцип			1			_		+
m 1.00 - + σ 0.10	- +	Событийный г	іный принцип 1				-		+	
	E	Выполнить								
Автор: Маслова Марина ИУ7-73Б										

Рисунок 3.1 – Пример работы с указанной вероятностью повторной обработки равной 0

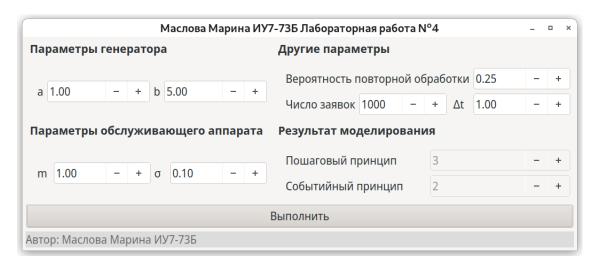


Рисунок 3.2 – Пример работы с указанной вероятностью повторной обработки равной 0.25

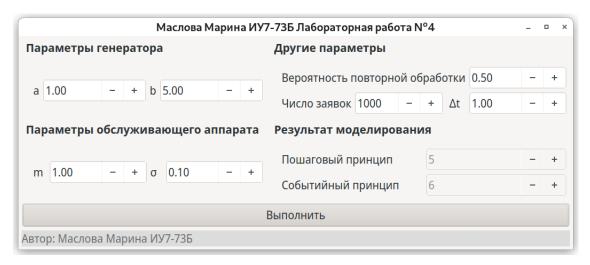


Рисунок 3.3 – Пример работы с указанной вероятностью повторной обработки равной 0.50

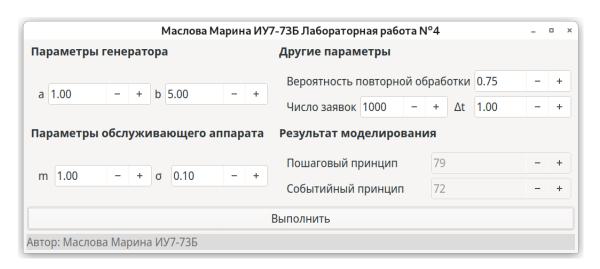


Рисунок 3.4 – Пример работы с указанной вероятностью повторной обработки равной 0.75

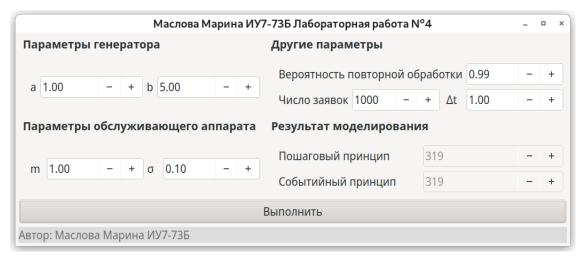


Рисунок 3.5 – Пример работы с указанной вероятностью повторной обработки равной 0.99