



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 системы массового обслуживания»

Студент:	<u>ИУ7-73Б</u> (группа)	_____ (подпись, дата)	<u>М. Д. Маслова</u> (И. О. Фамилия)
Преподаватель:		_____ (подпись, дата)	<u>И. В. Рудаков</u> (И. О. Фамилия)

2022 г.

СОДЕРЖАНИЕ

1	Задание	4
2	Теоретическая часть	5
2.1	Используемые распределения	5
2.1.1	Равномерное распределение	5
2.1.2	Нормальное распределение	5
2.2	Описание принципов	6
2.2.1	Пошаговый принцип	6
2.2.2	Событийный принцип	6
3	Практическая часть	7
3.1	Текст программы	7
3.2	Полученный результат	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:

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

Обозначение: $X \sim R[a, b]$.

Момент времени t_i может быть вычислен по следующей формуле:

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

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:

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

Обозначение: $X \sim N(m, \sigma^2)$.

Момент времени t_i может быть вычислен по следующей формуле:

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

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, исключая 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

```
1  class StepModel:
2
3      def run(self):
4          self.processor.set_aviable(True)
5          processed_requests = 0
6          total_requests = self.requests_num
7          generator_time = self.generator.next_time()
8          processor_time = -1
9          empty_generated = None
10
11         current_time = self.step
12         while processed_requests < total_requests:
13             if generator_time < current_time:
14                 if self.processor.is_aviable() and
15                     ↪ self.memory.is_empty():
16                     empty_generated = True
17                     self.memory.insert_request()
18                     saved_time = generator_time
19                     generator_time = saved_time + self.generator.next_time()
20
21             if 0 < processor_time < current_time:
22                 processed_requests += 1
23                 if random.randint(0, 100) < self.repeat_percent:
24                     self.memory.insert_request()
25                     self.processor.set_aviable(True)
26
27             if self.processor.is_aviable():
28                 if not self.memory.is_empty():
29                     self.memory.remove_request()
30                     self.processor.set_aviable(False)
31                     processor_time = ((saved_time if empty_generated
32                                     else processor_time)
33                                     + self.processor.process_time())
34             else:
35                 processor_time = -1
36
37             empty_generated = False
38             current_time += self.step
39
40         return self.memory.max_len
```

На листинге 3.2 представлена реализация событийного принципа.

Листинг 3.2 — Реализация событийного принципа

```
1  class EventModel:
2
3      def run(self):
4          self.processor.set_aviable(True)
5
6          processed_requests = 0
7          total_requests = self.requests_num
8          events = FutureEvents()
9
10         events.add(Event(self.generator.next_time(),
11             ↪ EventType.GENERATOR))
12
13         while processed_requests < total_requests:
14             cur_event = events.next()
15
16             if cur_event.event_type == EventType.GENERATOR:
17                 self.memory.insert_request()
18                 events.add(Event(cur_event.time +
19                     ↪ self.generator.next_time(),
20                         EventType.GENERATOR))
21
22             if cur_event.event_type == EventType.PROCESSOR:
23                 processed_requests += 1
24                 if random.randint(0, 100) < self.repeat_percent:
25                     self.memory.insert_request()
26                     self.processor.set_aviable(True)
27
28             if self.processor.is_aviable():
29                 if not self.memory.is_empty():
30                     self.memory.remove_request()
31                     self.processor.set_aviable(False)
32                     events.add(Event(cur_event.time +
33                         self.processor.process_time(),
34                             EventType.PROCESSOR))
35
36         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 Марина ИУ7-73Б Лабораторная работа №4

Параметры генератора		Другие параметры	
a	1.00	b	5.00
Параметры обслуживающего аппарата		Результат моделирования	
m	1.00	Вероятность повторной обработки	0.00
σ	0.10	Число заявок	1000
		Δt	1.00
		Пошаговый принцип	1
		Событийный принцип	1
Выполнить			
Автор: Маслова Марина ИУ7-73Б			

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

Маслова Марина ИУ7-73Б Лабораторная работа №4

Параметры генератора		Другие параметры	
a	1.00	b	5.00
Параметры обслуживающего аппарата		Результат моделирования	
m	1.00	Вероятность повторной обработки	0.25
σ	0.10	Число заявок	1000
		Δt	1.00
		Пошаговый принцип	3
		Событийный принцип	2
Выполнить			
Автор: Маслова Марина ИУ7-73Б			

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

Маслова Марина ИУ7-73Б Лабораторная работа №4

Параметры генератора		Другие параметры	
a	1.00	b	5.00
Параметры обслуживающего аппарата		Результат моделирования	
m	1.00	Вероятность повторной обработки	0.50
σ	0.10	Число заявок	1000
		Δt	1.00
		Пошаговый принцип	5
		Событийный принцип	6
Выполнить			
Автор: Маслова Марина ИУ7-73Б			

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

Маслова Марина ИУ7-73Б Лабораторная работа №4

Параметры генератора				Другие параметры				
a	1.00	-	+	b	5.00	-	+	
				Вероятность повторной обработки				
				0.75 - +				
				Число заявок		1000	-	+
				Δt		1.00	-	+
Параметры обслуживающего аппарата				Результат моделирования				
m	1.00	-	+	σ	0.10	-	+	
				Пошаговый принцип				
				79 - +				
				Событийный принцип				
				72 - +				
Выполнить								
Автор: Маслова Марина ИУ7-73Б								

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

Маслова Марина ИУ7-73Б Лабораторная работа №4

Параметры генератора				Другие параметры				
a	1.00	-	+	b	5.00	-	+	
				Вероятность повторной обработки				
				0.99 - +				
				Число заявок		1000	-	+
				Δt		1.00	-	+
Параметры обслуживающего аппарата				Результат моделирования				
m	1.00	-	+	σ	0.10	-	+	
				Пошаговый принцип				
				319 - +				
				Событийный принцип				
				319 - +				
Выполнить								
Автор: Маслова Марина ИУ7-73Б								

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