

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 (ИУ)»	
КАФЕЛРА «I	Ірограммное обеспечение ЭВМ и информационные технологии (ИУ7)»	

ОТЧЕТ Лабораторная работа №4

по курсу «Моделирование»

на тему: «Моделирование работы системы массового обслуживания»

Студент	ИУ7-73Б (Группа)	(Подпись, дата)	К.Э. Ковалец (И. О. Фамилия)
Преподав	атель	(Подпись, дата)	<u>И.В. Рудаков</u> (И. О. Фамилия)

Содержание

1	Зад	цание						•			•			•	•		•		•	•
2	Teo	ретич	еска	ія час	сть															4
	2.1	Равно	омер	ное ра	спреде	лен	ие.									•				4
	2.2	Распр	оедел	ение 3	Эрланг	a .												•		4
	2.3	Прин	ципь	и упра	ВЛЯЮШ	цей	проі	pa	MM	Ы										٢
		2.3.1	По	шагов	ый под	ĮХОД	ц													ļ
		2.3.2	Со	бытий	ный пр	ЭИНІ	цип	•							•					٦
3	Рез	зультат	ты р	аботі	ы															6
	3.1	Листи	инги	прогр	аммы	•										•				(
	3 2	Лемон	нстр	ания т	аботы	про	orna	мм	Ы											(

1 Задание

Промоделировать систему, состоящую из генератора, памяти и обслуживающего аппарата. Генератор подает сообщения, распределенные по равномерному закону, они приходят в память и выбираются на обработку по закону из ЛР1 (Эрланга). Количество заявок конечно и задано. Предусмотреть случай, когда обработанная заявка возвращается обратно в очередь. Определить оптимальную длину очереди, при которой не будет потерянных сообщений. Реализовать двумя способами: используя пошаговый и событийный подходы.

2 Теоретическая часть

2.1 Равномерное распределение

Функция равномерного распределения:

$$F(x) = \begin{cases} 0, x < a, \\ \frac{x - a}{b - a}, x \in [a, b], \\ 0, x > b. \end{cases}$$
 (2.1)

Функция плотности равномерного распределения:

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b], \\ 0, & else. \end{cases}$$
 (2.2)

2.2 Распределение Эрланга

Функция распределения Эрланга:

$$F_k(x) = 1 - e^{-\lambda \cdot x} \cdot \sum_{i=1}^{k-1} \frac{(\lambda \cdot x)^i}{i!}.$$
 (2.3)

Функция плотности распределения Эрланга:

$$f_k(x) = \frac{\lambda \cdot (\lambda \cdot x)^{k-1}}{(k-1)!} \cdot e^{-\lambda \cdot x}.$$
 (2.4)

В данных формулах λ и k — положительные параметры распределения $(\lambda\geqslant 0;k=1,2,\ldots);\ x\geqslant 0.$

2.3 Принципы управляющей программы

2.3.1 Пошаговый подход

Заключается в последовательном анализе состояний всех блоков системы в момент $t+\Delta t$ по заданному состоянию в момент 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 системные события должны имитироваться на данный момент времени. Основной недостаток: значительные затраты машинных ресурсов, а при недостаточном малых Δt появляется опасность пропуска события.

2.3.2 Событийный принцип

Характерное свойство модели системы обработки информации: состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщения, окончания решения задачи, возникновения аварийных сигналов и т. д. При использовании событийного принципа состояния всех боков системы анализируется лишь в момент появления какого-либо события. Момент наступления следующего события определяется минимальным значением из списка будущих событий, представляющий собой совокупность моментов ближайшего изменения состояния каждого из блоков. Момент наступления следующего события определяется минимальным значением из списка событий.

3 Результаты работы

3.1 Листинги программы

В листинге 3.1 представлена функция управления системой массового обслуживания с помощью пошагового принципа.

Листинг 3.1 — Пошаговый принцип

```
def stepModel(generator, processor, countTasks, repeatProbability, step):
1
          tasksDone = 0
3
          timeCurrent = step
          timeGenerated = generator.generate()
          timeGeneratedPrev = 0
          timeProcessed = 0
6
7
          curQueueLen = 0
8
          maxQueueLen = 0
9
          free = True
10
12
          while tasksDone < countTasks:</pre>
13
              # Генератор
14
              if timeCurrent > timeGenerated:
15
                   curQueueLen += 1
16
17
                   if curQueueLen > maxQueueLen:
18
                       maxQueueLen = curQueueLen
19
20
                   timeGeneratedPrev = timeGenerated
21
                   timeGenerated += generator.generate()
22
23
              # Обработчик
24
              if timeCurrent > timeProcessed:
25
                   if curQueueLen > 0:
26
27
                       wasFree = free
28
                       if free:
29
30
                           free = False
                       else:
31
                           tasksDone += 1
32
33
                           curQueueLen -= 1
34
                           if randint(1, 100) <= repeatProbability:</pre>
35
                                curQueueLen += 1
36
37
                       if wasFree:
38
```

Продолжение листинга 3.1

```
timeProcessed = timeGeneratedPrev + processor.generate()
39
40
                       else:
                           timeProcessed += processor.generate()
41
                  else:
42
                      free = True
43
44
              timeCurrent += step
45
46
47
          return maxQueueLen
```

В листинге 3.2 представлена функция управления системой массового обслуживания с помощью событийного принципа.

Листинг 3.2 — Событийный принцип

```
def eventModel(generator, processor, countTasks, repeatProbability):
2
          tasksDone = 0
          curQueueLen = 0
3
          maxQueueLen = 0
5
          free = True
          processFlag = False
6
          events = [[generator.generate(), "g"]]
7
8
          while tasksDone < countTasks:</pre>
9
              event = events.pop(0)
10
11
              # Генератор
12
              if event[1] == "g":
13
                  curQueueLen += 1
14
15
                   if curQueueLen > maxQueueLen:
16
                       maxQueueLen = curQueueLen
17
18
                  addEvent(events, [event[0] + generator.generate(), "g"])
19
20
                   if free:
21
                       processFlag = True
22
23
              # Обработчик
24
              elif event[1] == "p":
25
                  tasksDone += 1
26
27
                  if randint(1, 100) <= repeatProbability:</pre>
28
                       curQueueLen += 1
29
30
                  processFlag = True
31
```

Продолжение листинга 3.2

```
32
              if processFlag:
33
                   if curQueueLen > 0:
34
                       curQueueLen -= 1
35
                       addEvent(events, [event[0] + processor.generate(), "p"])
36
37
                   else:
38
                       free = True
39
40
                  processFlag = False
41
42
          return maxQueueLen
43
44
45
     def addEvent(events: list, event: list):
46
          i = 0
47
          while i < len(events) and events[i][0] < event[0]:</pre>
48
              i += 1
49
50
          if 0 < i < len(events):</pre>
51
              events.insert(i - 1, event)
52
          else:
53
              events.insert(i, event)
54
```

3.2 Демонстрация работы программы

На рисунках 3.1 - 3.4 представлены примеры работы программы.

	Лабораторна	ая работа №4										
	ГЕНЕ	PATOP										
Равномерный закон распределения												
	a b											
	-5	5										
	ОБСЛУЖИВАЮ)ЩИЙ АППАРАТ										
	Закон распред	еления Эрланга										
	k	λ										
	3	2										
	ПАРАІ	МЕТРЫ										
	Количество заявок	1000										
	Вероятность возврата заявки	0										
	Временной шаг	0.01										
	PE3Y	ПЬТАТ										
	Максимальная	длина очереди										
	Пошаговый подход	Событийный подход										
	2	585										
	Per	шить										
	О ПРОГ	PAMME										
	Информация	о программе										

Рисунок 3.1 – Результат работы при вероятности возврата заявки 0%

Лабораторн	ая работа №4								
ГЕНЕ	РАТОР								
Равномерный закон распределения									
a	b								
-5	5								
ОБСЛУЖИВАК	ОЩИЙ АППАРАТ								
Закон распред	еления Эрланга								
k	λ								
3	2								
ПАРА	МЕТРЫ								
Количество заявок	1000								
Вероятность возврата заявки	10								
Временной шаг	0.01								
РЕЗУ	ЛЬТАТ								
Максимальная	д длина очереди								
Пошаговый подход	Событийный подход								
73	496								
Pe	шить								
О ПРО	ГРАММЕ								
Информаци	я о программе								

Рисунок 3.2 — Результат работы при вероятности возврата заявки 10%

	Лабораторна	яя работа №4	
	ГЕНЕЯ	PATOP	
	Равномерный зак	он распределения	
	a	b	
-5		5	
	ОБСЛУЖИВАЮ	ЩИЙ АППАРАТ	
	Закон распреде	еления Эрланга	
	k	λ	
3		2	
	ПАРАМ	МЕТРЫ	
К	оличество заявок	1000	
-	ность возврата заявки	50	
	Временной шаг	0.01	
	РЕЗУЈ	ТЬТАТ	
	Максимальная	длина очереди	
Г	Іошаговый подход	Событийный подход	
221		674	
	Рец	ПИТР]
			1
	О ПРОГ	PAMME	
	Информация	о программе]

Рисунок 3.3 — Результат работы при вероятности возврата заявки 50%

		Лабораторна	ая работа №4							
		ГЕНЕЯ	PATOP							
Равномерный закон распределения										
		а	b							
	-5		5							
		ОБСЛУЖИВАЮ	ЩИЙ АППАРАТ							
		Закон распреде	еления Эрланга							
		k	λ							
	3		2							
		ПАРАМ	МЕТРЫ							
	ı	Количество заявок	1000							
	Вероя	этность возврата заявки	100							
		Временной шаг	0.01							
		РЕЗУЈ	ПЬТАТ							
		Максимальная	длина очереди							
		Пошаговый подход	Событийный подход							
	220		125							
		Рец	ШИТЬ							
		О ПРОГ	PAMME							
		Информация	о программе							

Рисунок 3.4 — Результат работы при вероятности возврата заявки 100%