*Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования*

|  |  |
| --- | --- |
| **Описание: Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

Факультет [Информатика и системы управления](https://vk.com/search?c%5bname%5d=0&c%5bsection%5d=people&c%5buni_country%5d=1&c%5buni_city%5d=1&c%5buniversity%5d=250&c%5bfaculty%5d=1033)

Кафедра Программное обеспечение ЭВМ и информационные технологии

**Отчёт**

**по лабораторной работе №5  
по курсу «Моделирование»  
 Тема: «Моделирование работы обслуживающего аппарата»  
Вариант 5**

Студент: Марабян К.В.

Группа: ИУ7-75

Преподаватель: Рудаков И.В.

Москва, 2019

**Задание**

Смоделировать систему, состоящую из генератора, очереди и ОА.   
Генератор вырабатывает заявки по равномерному закону. Время обслуживания распределено по экспоненциальному закону (в соответствии с вариантом). Определить оптимальную длину очереди, т.е. ту длину, при которой ни одно сообщение не исчезает. Должна быть возможность возвращения заявки в очередь после ее обработки с заданной вероятностью. Использовать 2 типа протяжки модельного времени: принцип dt и событийный принцип.

**Реализация**

**Принцип dt** заключается в последовательном анализе состояний всех блоков системы в момент t+dt. При этом новое состояние определяется в соответствии с их алгоритмическим описанием с учетом действующих случайных факторов, которые мы задаём распределениями вероятностей. В результаты анализа принимается решение о том какое общесистемное событие должно имитироваться на данный момент времени. Выбор точности моделирования и длительности времени моделирования подчиняется закономерности: время одного такта должно быть достаточным для обеспечения требуемой точности моделирования. Основным недостатком является значительные затраты модельного времени. А при недостаточно малом dt появляется опасность пропуска отдельных событий в системе, что приводит к нарушению адекватности всей модели.

При использовании **событийного принципа** есть специальный список событий. Состояния всех блоков системы анализируется лишь в момент наступлении какого-либо события. Момент наступления следующего события определяется минимальным из значений списка событий, представляющий собой совокупность моментов ближайшего изменения состояния. Недостаток проявляется в списке. Он может быть очень большим (долго просматривать список), что ведет к долгому вычислению модельного времени.

*Листинг 1. Принцип dt.*

|  |
| --- |
| *def* time\_based\_modelling(self):  generator = self.\_generator  processor = self.\_processor   time\_to\_appear = generator.generate()  time\_to\_process = time\_to\_appear + processor.generate()  curr\_time = 0  *while* self.\_processed\_requests < self.\_req\_count:  *if* curr\_time >= time\_to\_appear:  self.add\_to\_queue()  time\_to\_appear += generator.generate()  *if* curr\_time >= time\_to\_process:  self.process()  *if* self.\_queue\_size > 0:  time\_to\_process += processor.generate()  *else*:  time\_to\_process = time\_to\_appear + processor.generate()  curr\_time += self.dt   *return* (self.\_processed\_requests, self.\_reentered\_requests, self.\_max\_queue\_size, curr\_time) |

*Листинг 1. Событийный принцип.*

|  |
| --- |
| *def* event\_based\_modelling(self):  generator = self.\_generator  processor = self.\_processor   time\_to\_appear = generator.generate()  time\_to\_process = time\_to\_appear + processor.generate()  *while* self.\_processed\_requests < self.\_req\_count:  *if* time\_to\_appear <= time\_to\_process:  self.add\_to\_queue()  time\_to\_appear += generator.generate()  *if* time\_to\_appear >= time\_to\_process:  self.process()  *if* self.\_queue\_size > 0:  time\_to\_process += processor.generate()  *else*:  time\_to\_process = time\_to\_appear + processor.generate()   *return* (self.\_processed\_requests, self.\_reentered\_requests, self.\_max\_queue\_size, time\_to\_process) |

**Интерфейс и результаты работы программы**

