|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 №** | **4** |

**Название:**

«Обслуживающий аппарат»

**Дисциплина:** Моделирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-76Б |  |  | А. А. Петрова |
|  | (Группа) |  |  | (И.О. Фамилия) |
|  |  |  |  |  | |  |
| Преподаватель |  |  |  | И. В. Рудаков |
|  |  |  |  | (И.О. Фамилия) |

2022 г.

**Задание**

Промоделировать систему принципом «Δt» и событийным, состоящую из источника информации, буферной памяти и обслуживающего аппарата. Источник информации подает сообщения по равномерному закону распределения, а на обработку сообщения выбираются по нормальному закону распределения (10 вариант). С заданной вероятностью часть обработанных сообщений снова поступает в очередь (буферную память). Определить максимальный размер буферной памяти, при котором не будет потерь сообщений.

**Математическая формализация**

На рисунке ниже представлена схема моделируемой системы.

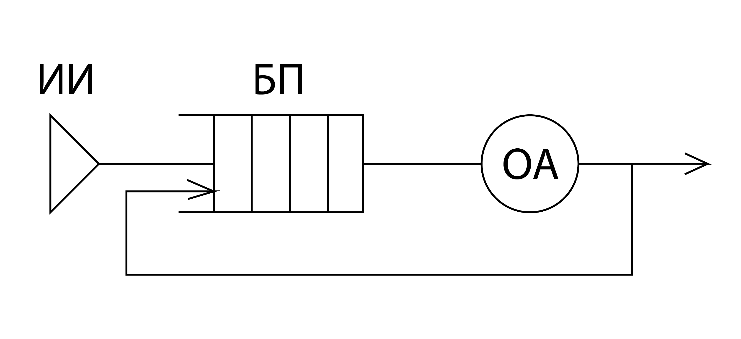


Рисунок 1: моделируемая система

Где ИИ – это источник информации, БП – буферная память, а ОА – обслуживающий аппарат.

**Принцип «Δt»**

Этот принцип заключается в последовательном анализе состояний всех блоков в момент времени 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 этого принципа – это значительные затраты вычислительных ресурсов, а при недостаточно малом появляется опасность пропуска отдельных событий в системе, исключающее возможность получения правильных результатов при моделировании.

**Событийный принцип**

Характерное свойство моделируемых систем обработки информации – это то, что состояние отдельных устройств изменяется в дискретные моменты времени, совпадающие с моментами поступления сообщений в систему, окончания реализации задания (процесса), возникновения прерываний и аварийных сигналов. Следовательно, моделирование и продвижение текущего времени в системе удобно проводить, используя событийный принцип.

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

**Реализация**

В листингах ниже представлена реализация принципа «Δt» и событийного принципа.

Листинг 1: принцип «Δt»

|  |
| --- |
| **def** time\_based\_modelling**(**self**,** request\_count**,** dt**):**  generator **=** self**.**generator  processor **=** self**.**processor  gen\_period **=** generator**.**get\_time**()**  proc\_period **=** gen\_period  current\_time **=** 0  **while** processor**.**processed\_requests **<** request\_count**:**  **if** gen\_period **<=** current\_time**:**  generator**.**request**()**  gen\_period **+=** generator**.**get\_time**()**  **if** current\_time **>=** proc\_period**:**  processor**.**process**()**  **if** processor**.**queue\_size **>** 0**:**  proc\_period **+=** processor**.**get\_time**()**  **else:**  proc\_period **=** gen\_period **+** processor**.**get\_time**()**  current\_time **+=** dt  **return** **(**processor**.**processed\_requests**,** processor**.**reentered\_requests**,**  processor**.**max\_queue\_size**,** **round(**current\_time**,** 3**))** |

Листинг 2: событийный принцип

|  |
| --- |
| **def** event\_based\_system**(**self**,** request\_count**):**  generator **=** self**.**generator  processor **=** self**.**processor  gen\_period **=** generator**.**get\_time**()**  proc\_period **=** gen\_period **+** processor**.**get\_time**()**  **while** processor**.**processed\_requests **<** request\_count**:**  **if** gen\_period **<=** proc\_period**:**  generator**.**request**()**  gen\_period **+=** generator**.**get\_time**()**  **if** gen\_period **>=** proc\_period**:**  processor**.**process**()**  **if** processor**.**queue\_size **>** 0**:**  proc\_period **+=** processor**.**get\_time**()**  **else:**  proc\_period **=** gen\_period **+** processor**.**get\_time**()**  **return** **(**processor**.**processed\_requests**,** processor**.**reentered\_requests**,**  processor**.**max\_queue\_size**,** **round(**proc\_period**,** 3**))** |

**Результаты работы**

Для всех тестов использовались следующие параметры распределений:

* равномерное: a = 0, b = 10;
* нормальное: m = 0, σ = 1.

Количество запросов: 10000.

Δt = 0.01.

Ниже представлены результаты работы программы в зависимости от вероятности (P) повторного попадания заявки в очередь.

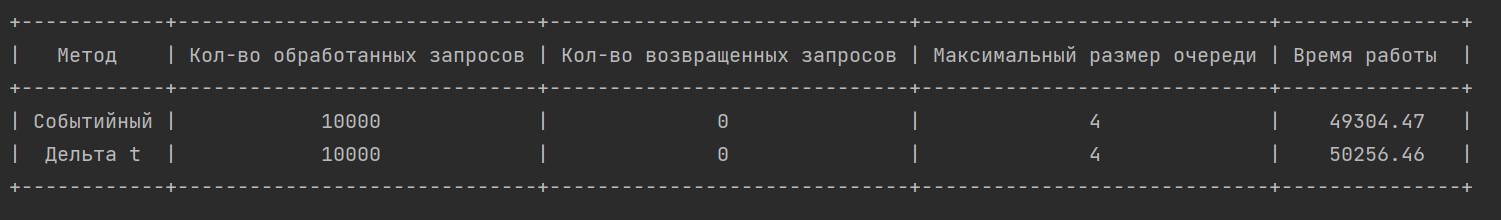
****

Рисунок 2: работа программы при P = 0%

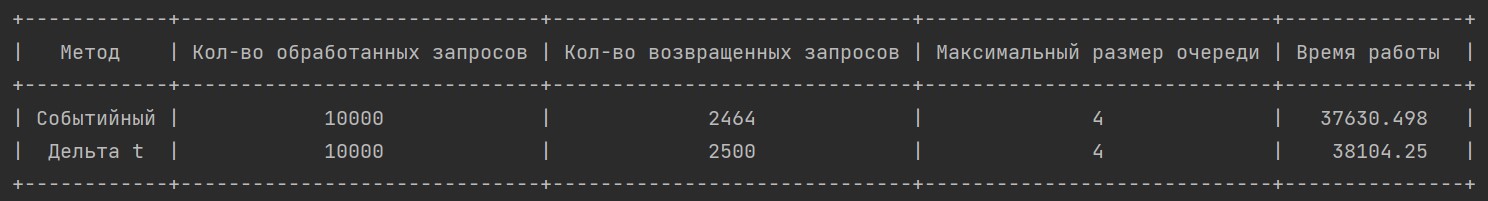
**

Рисунок 3: работа программы при P = 25%

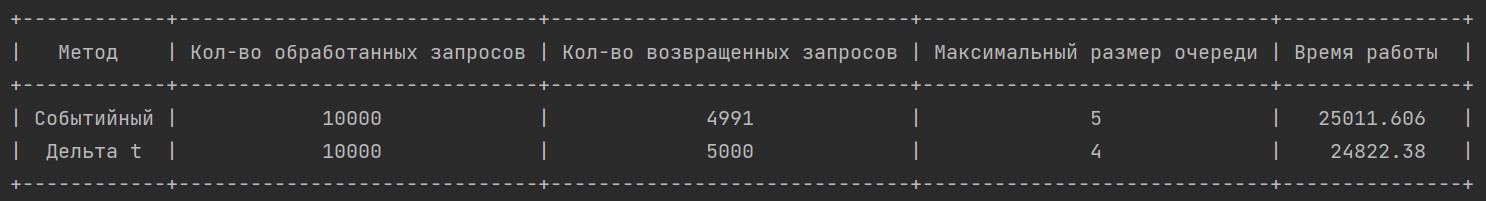


Рисунок 4: работа программы при P = 50%

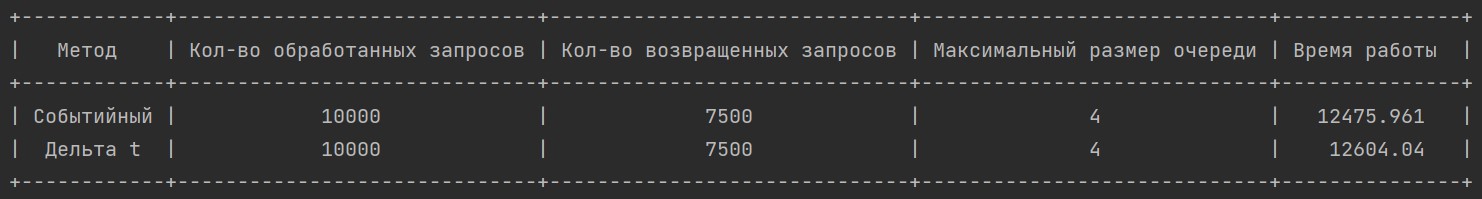


Рисунок 5: работа программы при P = 75%