## 1 Алгоритмы имитационной модели

В начале проведения симуляции управление находится у ведущей программы $\pi_0$. В ходе её выполнения управление может передаваться программным процессам $\pi_a$, $\pi_{bs}$ и $\pi_{es}$. Алгоритмы данных процессов приведены ниже.

Алгоритм программного процесса $\pi_a$ — **«Поступление требования в систему»** — включает следующие действия:

1. Определение момента $t = t_{now} + t_a$ поступления требования в систему, где $t_{now}$ — текущий момент модельного времени.
2. Выход из процесса $\pi_a$ и возврат к ведущей программе.
3. Формирование требования и разбиение его на фрагменты. Этот процесс заключается в определении числа фрагментов, что является случайной величиной, создании для каждого фрагмента элемента коллекции, который будет содержать набор переменных, хранящих такие атрибуты фрагмента, как момент времени его появления в системе, идентификатор требования, к которому он относится, и его текущее состояние.
   *a*) Первому атрибуту присваивается значение текущего момента модельного времени $t_{now}$.
   *a*) Последнему атрибуту присваивается значение $-1$, означающее, что этот фрагмент находится в очереди.
4. Все фрагменты требования ставятся в очередь системы.
5. Переход на шаг 1.

Алгоритм программного процесса $\pi_{bs}$ — **«Начало обслуживания»** —включает следующие действия:

1. Определение номера свободного прибора обслуживания.
2. Изменение состояние этого прибора на значение «занят».
3. Атрибуту «текущее состояние» взятого из очереди фрагменту требования присваивается значение, равное номеру прибора.
4. Определение момента $t = t_{now} + t_{es}$ завершения обслуживания фрагмента требования, где $t_{now}$ — текущий момент модельного времени, а $t_{es}$ — длительность обслуживания данного фрагмента.
5. Выход из процесса $\pi_{bs}$ и возврат к ведущей программе.

Алгоритм программного процесса $\pi_{es}$ — **«Завершение обслуживания»** — включает следующие действия:

1. Изменение состояния завершившего обслуживание прибора на значение

«свободен».

2. Атрибуту «текущее состояние» обслуженного фрагмента требования присваивается значение, равное общему числу приборов, что означает его готовность к сборке.
3. Проверка: если все фрагменты данного требования были обслужены и ожидают сборки, то они покидают систему как единое требование.
4. Если очередь системы не пуста, то перейти к программному процессу $\pi_{bs}$ с шага 1. В противном случае, выход из процесса $\pi_{es}$ и возврат к ведущей программе.

Алгоритм **ведущей программы** $\pi_0$:

1. Определение начальных условий.
2. Если очередь системы не пуста и хотя бы один прибор обслуживания свободен, то передать управление программному процессу $\pi_{bs}$ с шага 1. Иначе выполнить шаг 3.
3. Из таблицы расписания событий выбрать событие с минимальным моментом активации.
4. Продвинуть текущий момент модельного времени $t_{now}$ до момента активации выбранного события.
5. Выполняется проверка условия $t_{now} \leq t_{max}$, где $t_{now}$ — общее время моделирования. Если неравенство не выполняется, то процесс моделирования завершается, и вычисляются характеристики модели. Конец алгоритма. В противном случае переход к шагу 6.
6. В соответствии с событием выбирается программный процесс, которому необходимо передать управление:
   *а*) Если выбранным событием является генерация нового требования, то управление получает процесс $\pi_a$.
   *б*) Если выбранным событием является завершение обслуживания фрагмента требования, то управление получает процесс $\pi_{es}$.

## 2 Вероятностно-временные характеристики имитационной модели

В ходе осуществления функционирования имитационной модели ведущая программа ведёт сбор статистических данных и вычисляет оценки некоторых характеристик системы.

Записывается суммарное время $t_n$, $n = 0, 1, 2, \ldots$, нахождения в системе ровно $n$ требований. Затем с помощью выражения

$$\hat{p}_n = \frac{t_n}{t_{max}}$$

вычисляются оценки стационарных вероятностей состояний системы.

Кроме того, после окончания обслуживания каждого требования, его общая длительность $\tau$ пребывания в системе записывается в суммирующую переменную, содержащую аналогичную характеристику для всех обслуженных требований. С помощью выражения

$$\hat{u} = \frac{\sum_{i=1}^{Q} \tau_i}{Q},$$

где $Q$ — количество обслуженных требований, вычисляется оценка среднего времени пребывания требования в системе.

## 3 Структура программы имитационной модели

По описанным ранее алгоритмам была разработана программа имитационной модели. Для реализации использовался язык программирования Python и модули *math*, *numpy* и *json*. Программа позволяет вычислять оценки следующих вероятностно-временных характеристик рассматриваемой системы массового обслуживания типа $M^{[x]}/M/C$ при заданных параметрах:

— стационарное распределение вероятностей состояний системы $\hat{p}$.

— среднее времени прибывания требования $\hat{u}$ в системе.

Программа состоит из двух модулей:

1. System.py — модуль, в котором содержится класс $Mx\_M\_C$, описывающий соответствующую систему. В классе определены следующие атрибуты:
   — *lambda_* — интенсивность входящего потока;
   — *servers_count* — число обслуживающих приборов в системе;
   — *mu* — интенсивность обслуживания на приборах системы;
   — *servers_states* — вектор состояния обслуживающих приборов системы;
   — *demands* — коллекция, содержащая требования, находящиеся в очереди системы;
   — *last_state* — номер предыдущего состояния, то есть числа требований, системы.

   В классе $Mx\_M\_C$ содержатся следующие методы:
   — *arrival_time* — возвращает момент времени, когда очередной требование появится в системе;
   — *service_time* — возвращает момент времени, когда система завершит обслуживание требования;
   — *pack_size* — возвращает число фрагментов, на которые разобьётся требование;
   — *export_states* — сохраняет данные о времени пребывания системы в каждом состоянии в соответствующей файл;
   — *export_demands* — сохраняет данные о фрагментах требований в системе в соответствующей файл;
   — *import_states* — загружает данные о времени пребывания системы в каждом состоянии из соответствующего файла;

- — *import_ demands* — загружает данные о фрагментах тербований в системе из соответствующего файла;
- — *current_ demands* — возвращает идентификаторы присутствующих в системе требований;
- — *update_ time_ states* — обновляет данные о времени пребывания системы в каждом состоянии.

2. main.py — основной модуль, содержащий точку входа в программу. Содержит исходные данные для моделирования:
   - — *t_ max* — общее время моделирования;
   - — lambda_ — интенсивность входящего потока;
   - — *mu* — интенсивность обслуживания приборами системы;
   - — *servers_ count* — число обслуживающих приборов;
   - — *b* — среднее число фрагментов, на которые разбивается требование;
   - — *t* — текущее значение модельного времени;
   - — *indicator* — индикатор, отражающий, произошло ли на данном моменте модельного времени какое-либо событие;
   - — *schedule* — таблица временных отметок активации событий;
   - — *ready_ packs_ count* — количество обслуженных требований;
   - — *sum_ packs_ life_ time* — суммарное время нахождения требований в системе;

   В теле модуля содержится функция *simulation*, которая осуществляет процесс симуляции, продвигая модельное время вперёд и обрабатывая возникающие события, а также собирает данные статистики.

## 4 Результаты имитационного моделирования

В ходе проведения экспериментов с реализованной имитационной моделью с целью нахождения оценок характеристик рассматриваемой системы массового обслуживания типа $M^{[x]}$ были получены примеры результатов моделирования.

На основе данных экспериментов можно сделать вывод, что разработанная модель применима для анализа системы массового обслуживания с делением и слиянием требований: интенсивность поступления требований в систему, интенсивностью их обслуживания, количеством обслуживающих приборов и средним числом фрагментов, на которые разделяется требование.

**Пример 1.**

Рассматривается система массового обслуживания $M^{[x]}/M/C$.

Параметры генерации требований следующие: длительности интервалов между поступающими требованиями имеют экспоненциальное распределение, интенсивность поступления $\lambda = 1/10$.

Параметры системы определены следующим образом:

— длительность обслуживания имеет экспоненциальное распределение,
— интенсивность обслуживания $\mu 1/281$,
— число обслуживающих приборов $= 150$,
— среднее число фрагментов разбиения требования $\overline{b} = 5$.

**Результаты моделирования системы массового обслуживания:**

— оценка среднего времени пребывания требования в системе $\hat{u} =$,
— оценка вероятностей стационарного распределения состояний системы $\hat{p} = ()$.

**Пример 2.**

Рассматривается система массового обслуживания $M^{[x]}/M/C$.

Параметры генерации требований следующие: длительности интервалов между поступающими требованиями имеют экспоненциальное распределение, интенсивность поступления $\lambda = 1/10$.

Параметры системы определены следующим образом:

— длительность обслуживания имеет экспоненциальное распределение,
— интенсивность обслуживания $\mu 1/281$,
— число обслуживающих приборов $= 150$,
— среднее число фрагментов разбиения требования $\overline{b} = 5$.

**Результаты моделирования системы массового обслуживания:**

— оценка среднего времени пребывания требования в системе $\hat{u} =$,

— оценка вероятностей стационарного распределения состояний системы $\hat{p} = ()$.