МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

**Метрология, стандартизация и спецификация**

Отчет

по лабораторной работе №3

ПРОВЕРИЛ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жевнерчук Д. В.

ВЫПОЛНИЛИ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чугунков В. С.

Байрамов Р. Ш.

Теркалов А. С.

23-ИВТ-5

Нижний Новгород

2025 г.

Задание

Создать систему массового обслуживания (СМО) по обработке заявок, используя язык Python.

Система имеет 1 обработчик, имеющий 2 состояния “свободен” (нет заявки в обработке) и “занят” (обрабатывается заявка). Он способен обрабатывать лишь одну заявку за раз. При поступлении новых заявок во время состояния “занят” эти заявки помещаются в очередь. Заявки поступают в реальном времени.

Пользователь задает минимальный и максимальный промежутки времени, через которые в систему могут поступать новые заявки. Так же пользователь задает максимальное и минимальное время, которое заявки могут обрабатываться.

Система функционирует определенное количество времени, заранее определенно пользователем, после чего обработка заявок прекращается, а обрабатываемая в момент прекращения заявка считается незавершенной.

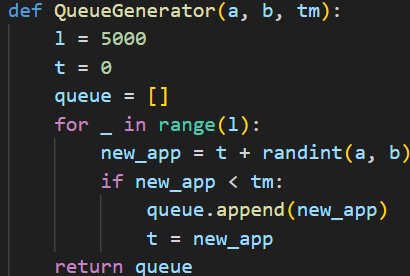
Реализация

Система будет состоять из 3 основных функций:

1. **MassServiceSystem(receiption\_queue, processing\_queue, a, b, c, d). Основная функция, запускающая работу системы. В ней происходит распределение новых заявок и управление очередью заявок, путем вызова функций ApplicationReceiption и ApplicationProcessing. В receiption\_queue хранятся еще непоступившие заявки (см. ниже), processing\_queue - очередь поступивших заявок. Переменные a и b - минимальный и максимальный промежуток времени между поступлением заявок, c и d - минимальное и максимальное время обработки заявок. Подробный разбор кода последует после объяснения основных переменных и функций.**

**Поступление заявок в реальном времени симулирется с помощью списка receiption\_queue с временными метками поступления заявок. Эти метки формируются в функции QueueGenerator.**

**QueueGenerator(a, b, tm). Генерирует случайное временных меток заявок по заданному диапазону [a, b]. Переменная tm - максимальное время работы системы, задействуется чтобы не сгенерировать заявку, которая поступит после завершения работы. Промежутки между временными метками (перерывы между поступлением заявок) случайно генерируются из заданного пользователем диапозона [a, b].**

****

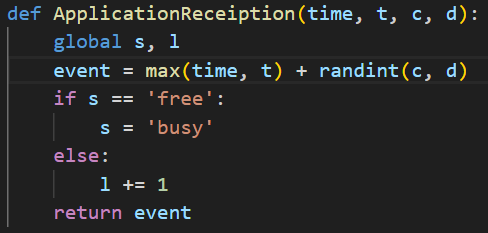
****

**В processing\_queue хранятся временные метки завершения обработки поступивших в очередь заявок. Эти метки формируются в функции ApplicationReceiption (см. ниже). В начальный момент времени time=0 пуста.**

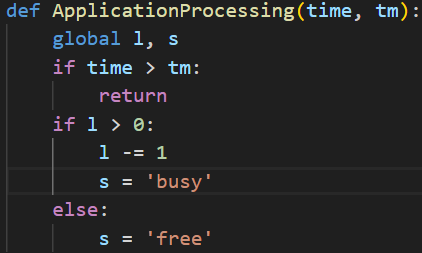
**2. ApplicationReceiption(time, t, c, d). Функция приема новой заявки. В зависимости от состояния обработчика (переменная s).**

**Если обработчик свободен (s=’free’) отправляет заявку на обработку (меняет s с free на busy), если обработчик занят - отправляет в очередь, и увеличивает ее длину в переменной l.**

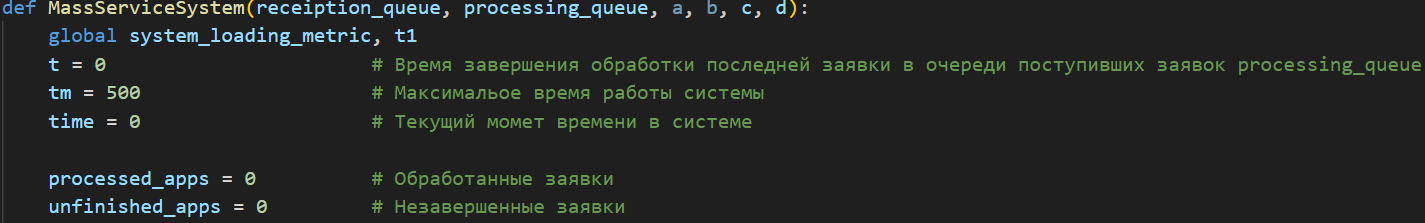
**Переменная time - время поступления заявки (а также текущий момент времени в системе), t - время завершения обработки последней заявки в очереди поступивших заявок processing\_queue (необходима для вычисления времени обработки новопоступившей завки). При любом состояни s высчитывается временная метка завершения обработки заявки, эта метка помещается в переменную event, которая впоследствии отправляется в конец очереди processing\_queue (не в этой функции, см. далее). Промежутки между временными метками (время обработки заявки) случайно генерируются из заданного пользователем диапозона [c, d].**

****

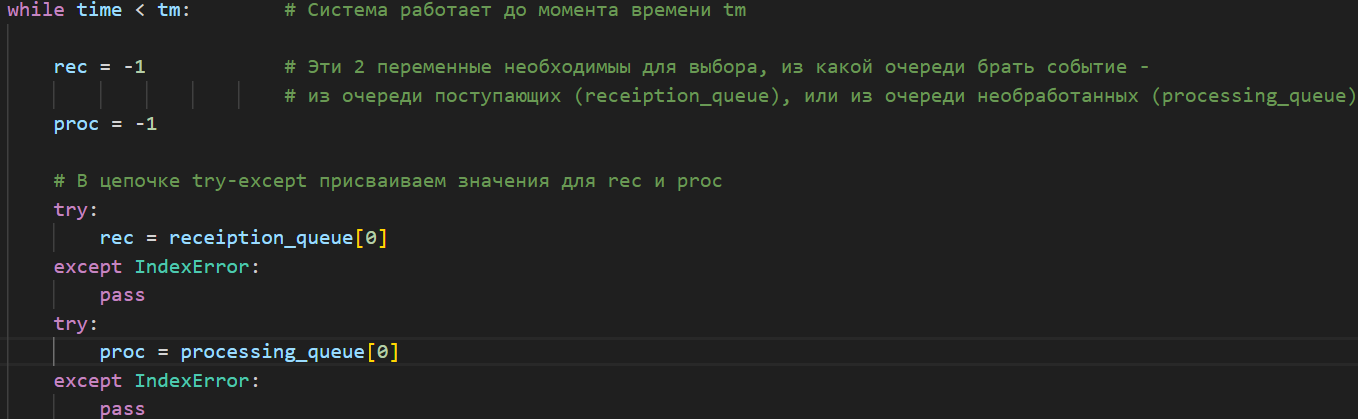
**3. ApplicationProcessing(). Обрабатывает заявки. Если заявка пришла из очереди, эта заявка из очереди удаляется и длина l уменьшается. Если после этого очередь становится пустой, меняет состояние обработчика с busy на free.**

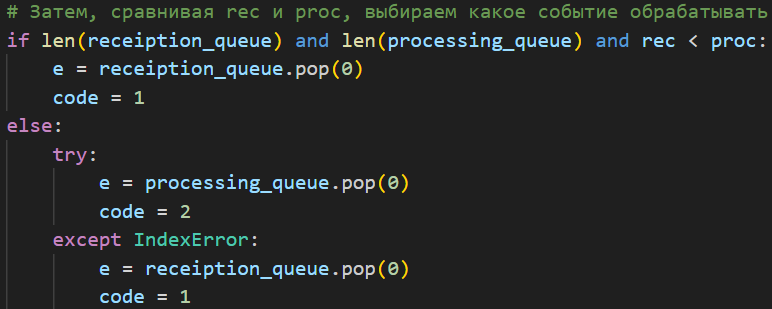
****

**Теперь, введя основные функции и переменные, разберем функцию MassServiceSystem(receiption\_queue, processing\_queue, a, b, c, d). В начале задаются вышеописаные пременные, а также счетчик обработанных (processed\_apps) и незавершенных заявок (unfinished\_apps).**

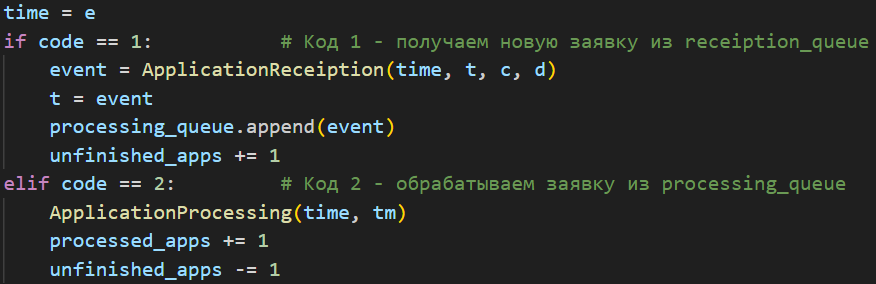
****

**Далее запускается цикл обработки заявок. Сначала из очередь поступающих (receiption\_queue) и уже поступивших заявок (processing\_queue) выбирается наименьшая (событие, которое произойдет рагьше всего). Для этого вводятся переменные rec и proc. Если метка взята из receiption\_queue, переменная code=1, если из processing\_queue, то code=2**

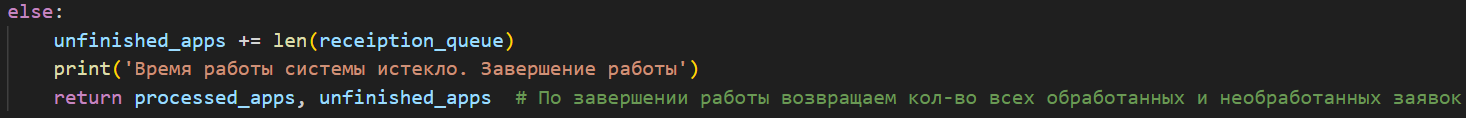
****

****

**Далее обновляем time, и в зависимости от значения code запускаем либо ApplicationReceiption (code=1), либо ApplicationProcessing (code=2). Обновляем счетчики заявок и при необходимости обновляем переменную t и добавляем новую временную метку в processing\_queue.**

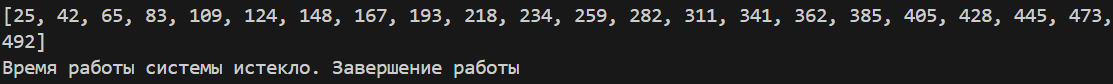
****

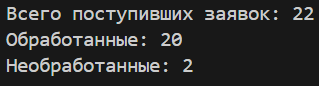
**По завершении работы СМО возвращаются счетчиками обработанными и необработанными заявками.**

****

**Пример работы**

****

****

****