**Отчет по лабораторной работе 5**

**По предмету «Типы и структуры данных»**

Студент ИУ7-32Б

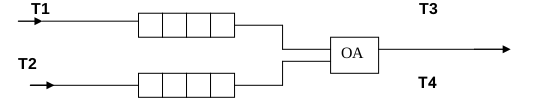
Герасименко Екатерина

**Цель работы:** отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценка эффективности программы (при различной реализации) по времени и по используемому объему памяти.

**Условие задачи:**

Система массового обслуживания состоит из обслуживающего аппарата

(ОА) и двух очередей заявок двух типов.



Заявки 1-го и 2-го типов поступают в "хвосты" своих очередей по

случайному закону с интервалами времени Т1 и Т2, равномерно

распределенными от 1 до 5 и от 0 до 3 единиц времени (е.в.) соответственно.В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за времена Т3 и Т4, распределенные от 0 до 4 е.в. и от 0 до 1 е.в. соответственно, после чего покидают систему. (Все времена – вещественного типа) В начале процесса в системе заявок нет.

Заявка 2-го типа может войти в ОА, если в системе нет заявок 1-го типа.

Если в момент обслуживания заявки 2-го типа в пустую очередь входит заявка 1-го типа, то она ждет первого освобождения ОА и далее поступает на обслуживание (система с относительным приоритетом ).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа.

Выдать на экран после обслуживания каждых 100 заявок 1-го типа

информацию о текущей и средней длине каждой очереди, количестве

вошедших и вышедших заявок и о среднем времени пребывания заявок в

очереди. В конце процесса выдать общее время моделирования и количество вошедших в систему и вышедших из нее заявок обоих типов. По требованию пользователя выдать на экран адреса элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

**Входные данные:**

Интервалы времени Т1, Т2, Т3, Т4

**Выходные данные:**

Информация по выполненному процесу

**Функция программы:**

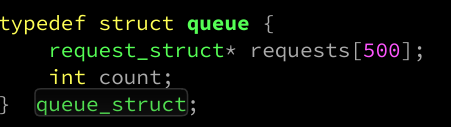
Реализация системы обработки заявок из двух очередей, с относительным приоритетом.

**Обращение к программе осуществляется через консоль.**

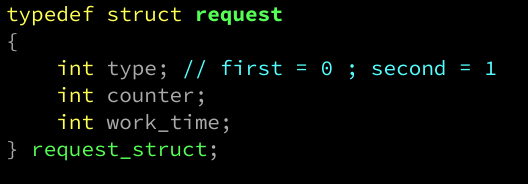
**Структуры данных:**

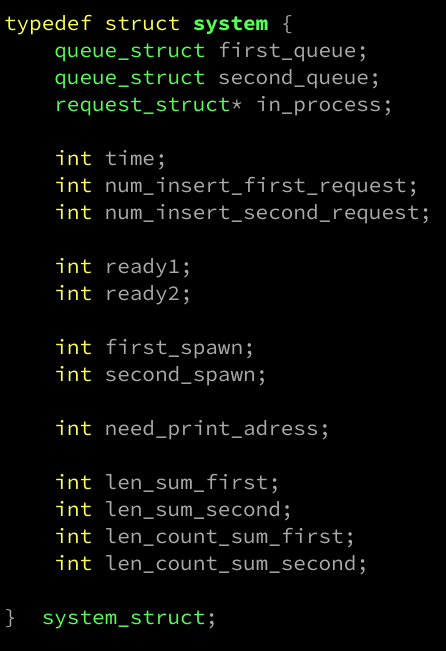
Для массива:

* Очередь

****

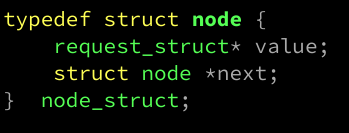
* Заявка

****

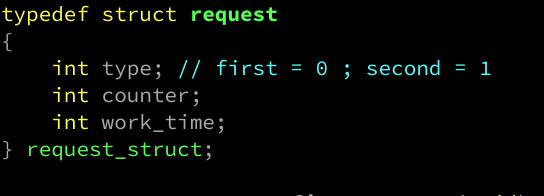
****

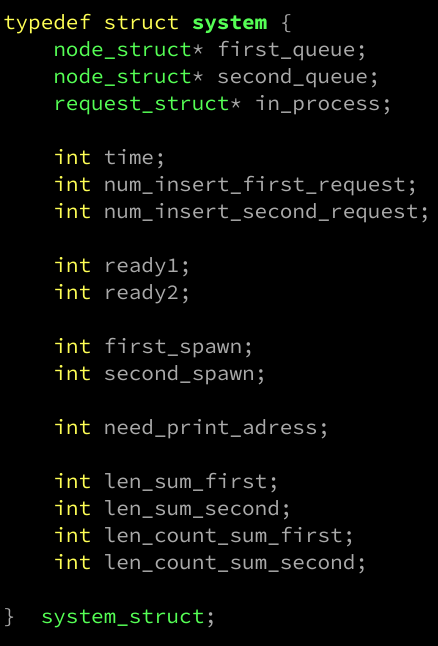
Для списка:

* Списковая ячейка

****

* **Заявка**

****

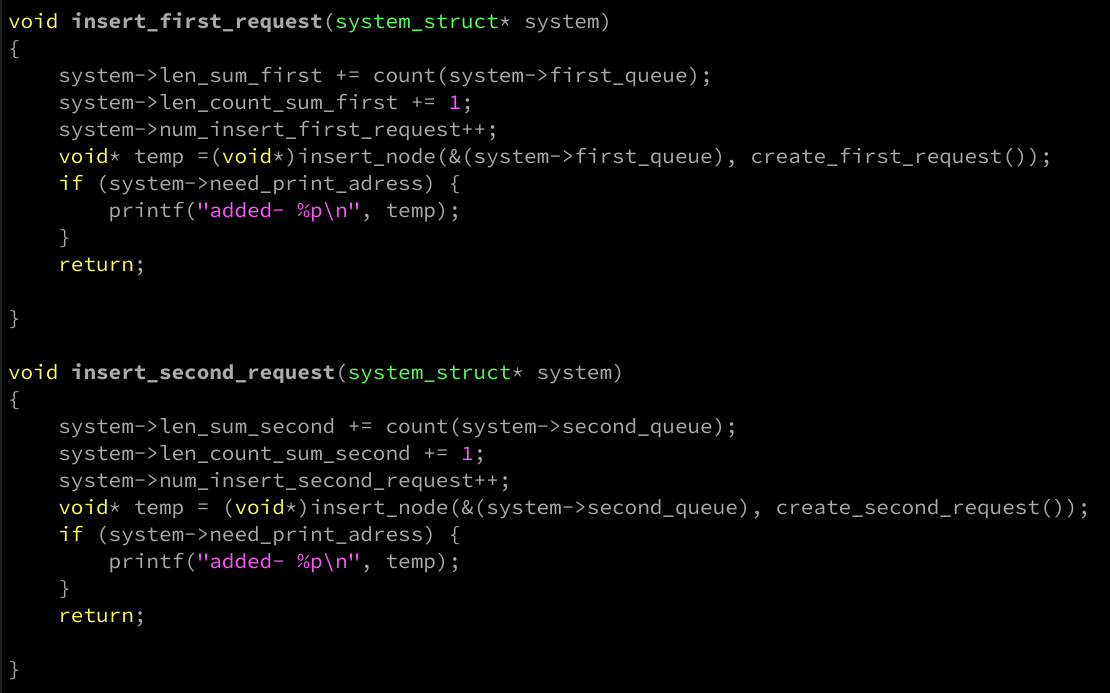
****

**Функции:**

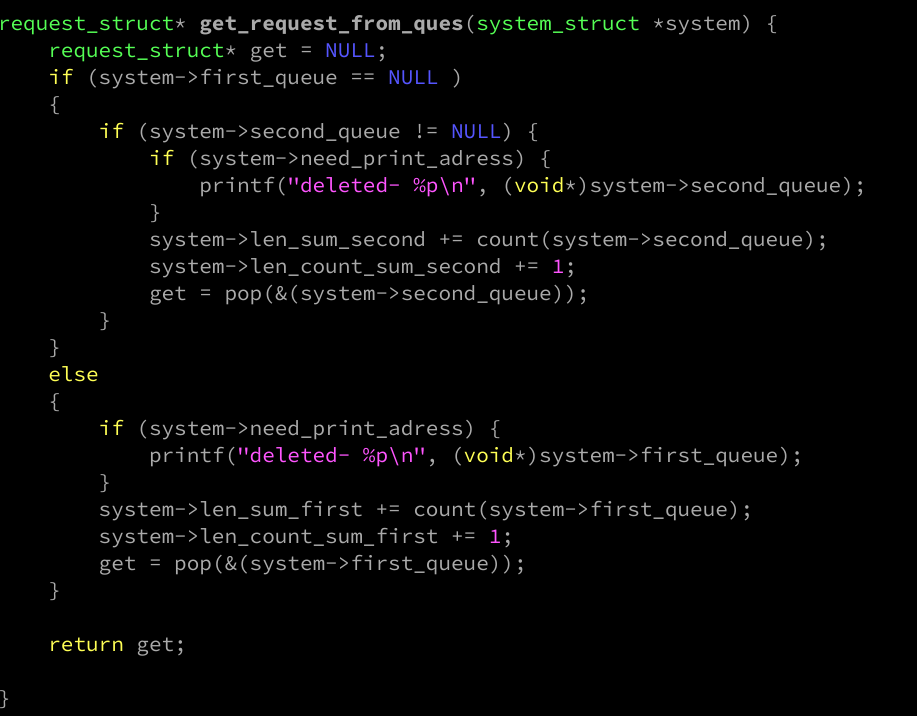
* Создание заявки первого/второго типа

****

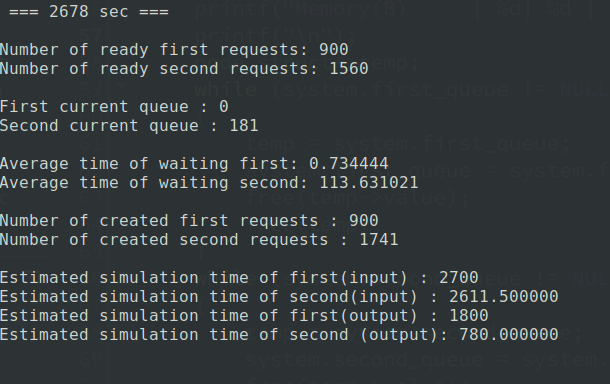
* Добавление заявки первого второго типа в очередь

****

* Получает заявку из очереди с учетом приоритета

****

**Промежуточный вывод информации:**

****

**Расчетное время моделирования**

В данной задаче время прихода больше времени обработки, следовательно очереди не будет, и время моделирования будет определяться временем прихода заявок.

Следовательно :

Среднее время прихода в первой очереди = (5 + 1) / 2 = 3

Среднее время прихода в второй очереди = (3 + 0) / 2 = 1.5

Среднее время обработки в первой очереди = (5 + 1) / 2 = 2

Среднее время обработки в второй очереди = (3 + 0) / 2 = 0.5

Созданное количество заявок в первой очереди 1000, во второй 2000.

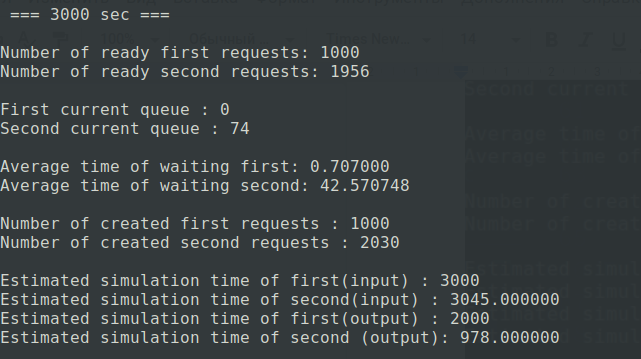
Estimated simulation time of first(input) = 3 \* 1000 = 3000

Estimated simulation time of second(input) = 1.5 \* 2000 = 3000

Estimated simulation time of first(output) = 2 \* 1000 = 2000

Estimated simulation time of second (output) = 0.5 \* 2000 = 1000

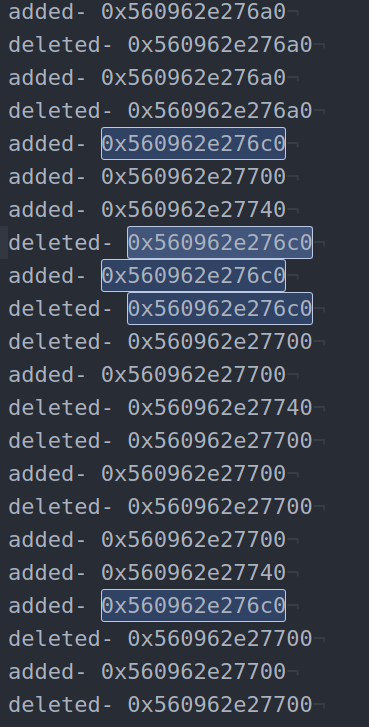
Следовательно все заявки второго типа будут обработаны.

****

Итоговое время моделирования 3000 sec. Следовательно разница между итоговым временем и расчетным временем моделирования менее 3%.

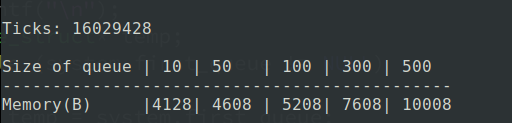
**Фрагментация памяти:**

Фрагментация памяти присутствует.

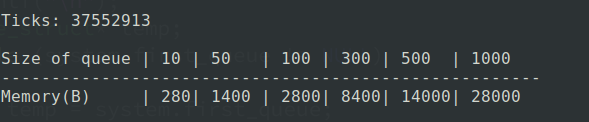
****

**Анализ эффективности (по памяти и времени):**

Для массива:



Для списка:



**Вывод:**

Список проигрывает по времени, но выигрывает по памяти на небольших размерностях (в данном случае до 300 элементов). Массив в данной реализации выигрывает по времени и начинает выигрывать по памяти начиная с 300 элементов.

**Контрольные вопросы:**

1. **Что такое очередь?**

Очередь – последовательной список переменной длины. Включение элементов идёт с «хвоста» списка, исключение – с «головы» списка. Принцип работы: первым вошел – первым вышел.

2. **Каким образом, и какой объем памяти выделяется под хранение очереди при различной ее реализации?**

При реализации списком, память выделяется динамически по мере необходимости.

При реализации массивом, память выделяется сразу под все элементы, которые затем хранятся в массиве.

3. **Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?**

При реализации списком, считывается первый с головы (текущий) элемент, происходит смещение головы, а тот элемент освобождается.

При реализации очереди массивом, считывается текущий элемент, остальные элементы сдвигаются на 1 элемент в сторону текущего элемента.

4. **Что происходит с элементами очереди при ее просмотре?**

При просмотре очереди текущий элемент из нее удаляется.

5. **Каким образом эффективнее реализовывать очередь. От чего это зависит?**

Выбор способа зависит от приоритетов: время или память.

При реализации списком легче добавить и удалить элемент, но при этом может возникнуть фрагментация памяти. При реализации массивом при удалении необходимо сдвигать все его элементы, что, при больших размерах, может быть очень затратно по времени.

7. **Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций?**

При реализации очереди массивом не возникает фрагментации памяти, затрачивается время на сдвиг элементов. При реализации списком может возникнуть фрагментация памяти.

8. **Что такое фрагментация памяти?**

Фрагментация – чередование занятых и свободных участков памяти при последовательных запросах на добавление и удаление. Свободные участки могут быть слишком малы, чтобы хранить в них нужную информацию.

9. **На что необходимо обратить внимание при тестировании программы?**

Необходимо обратить внимание на корректное освобождение памяти при удалении элемента из очереди.

10. **Каким образом физически выделяется и освобождается память при динамических запросах?**

Программа запрашивает блок памяти необходимого размера. ОС находит подходящий блок, записывает его адрес и размер в таблицу адресов, а затем возвращает данный адрес в программу. При запросе на освобождение указанного блока программы, ОС убирает его из таблицы адресов, адрес считается освобожденным. При попытке доступа к освобожденной памяти могут возникнуть ошибки.