МГТУ им. Н.Э. Баумана

**Лабораторная Работа № 5**

**Обработка очередей**

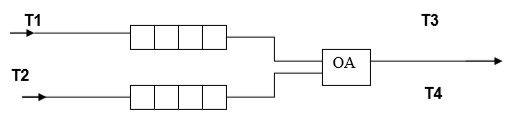
**Цель работы:** отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценка эффективности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

Студент: Нгуен Ань Тхы

Группа : ИУ7 И - 36Б

1) Описание условия 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 любого типа может войти в ОА, если:

а) она вошла в пустую систему;

б) перед ней обслуживалась заявка ее же типа;

в) перед ней из ОА вышла заявка другого типа, оставив за собой пустую очередь (система с чередующимся приоритетом). Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине каждой очереди, а в конце процесса - 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.

2) Описание ТЗ:

Исходные данные:

Интервалы времени T1, T2, T3, T4 (не отрицательные числа)

Выходные данные:

Информации соответствует каждому операциями:

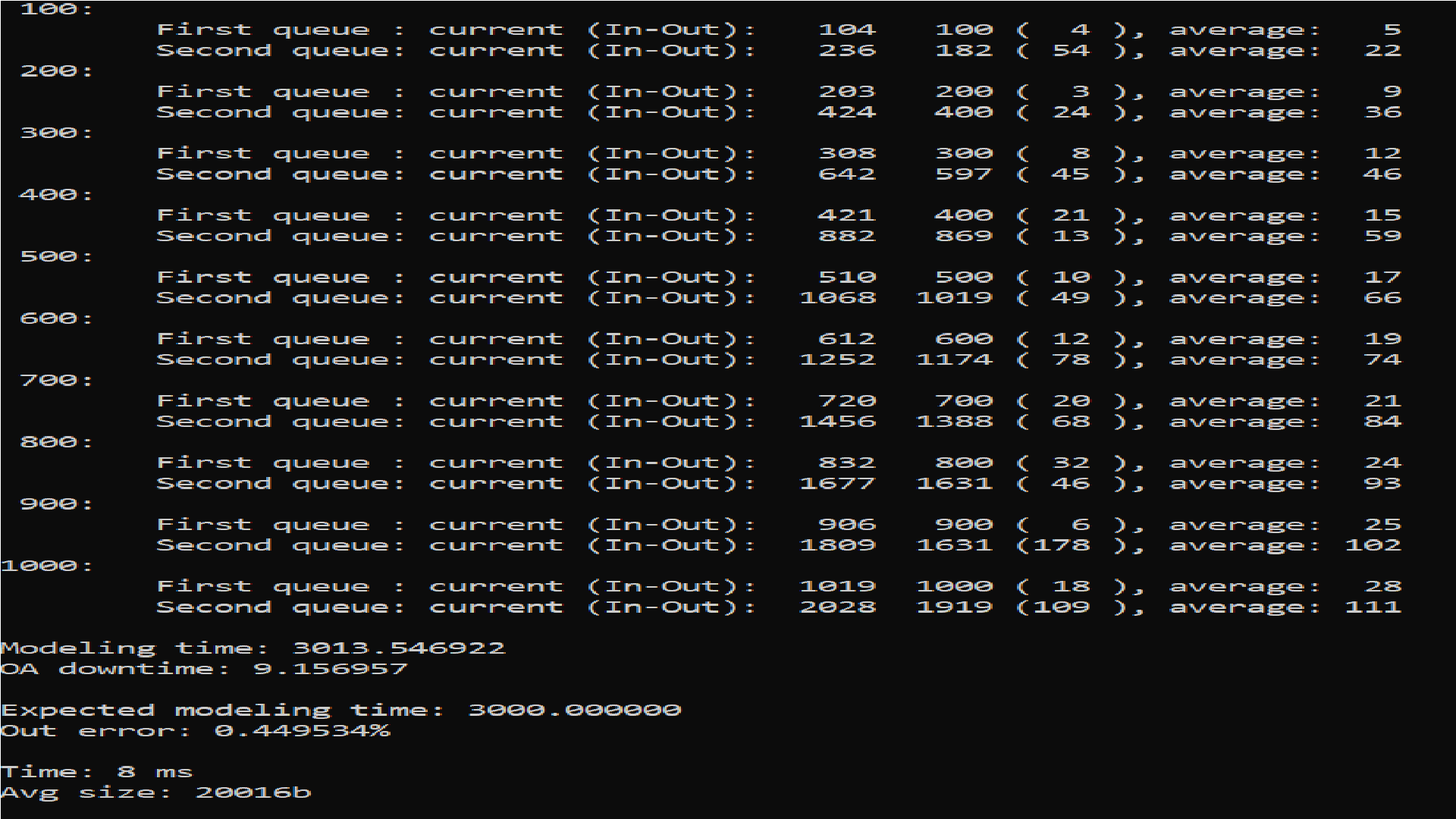
- выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине каждой очереди

- общее время моделирования и количество вошедших в систему

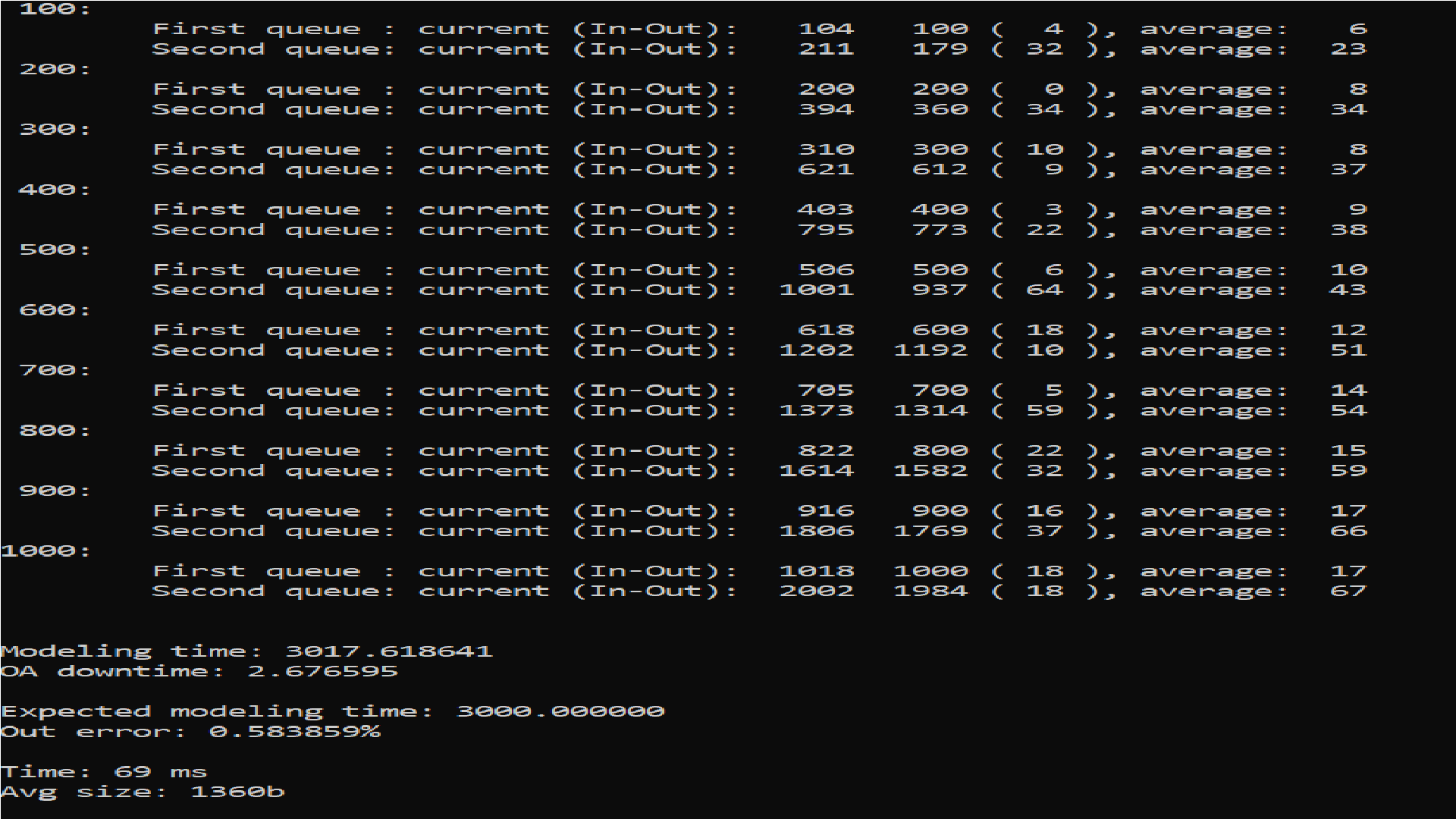
- адреса элементов очереди при удалении и добавлении элементов.

3) Пример тесты:

Стек с массивом:

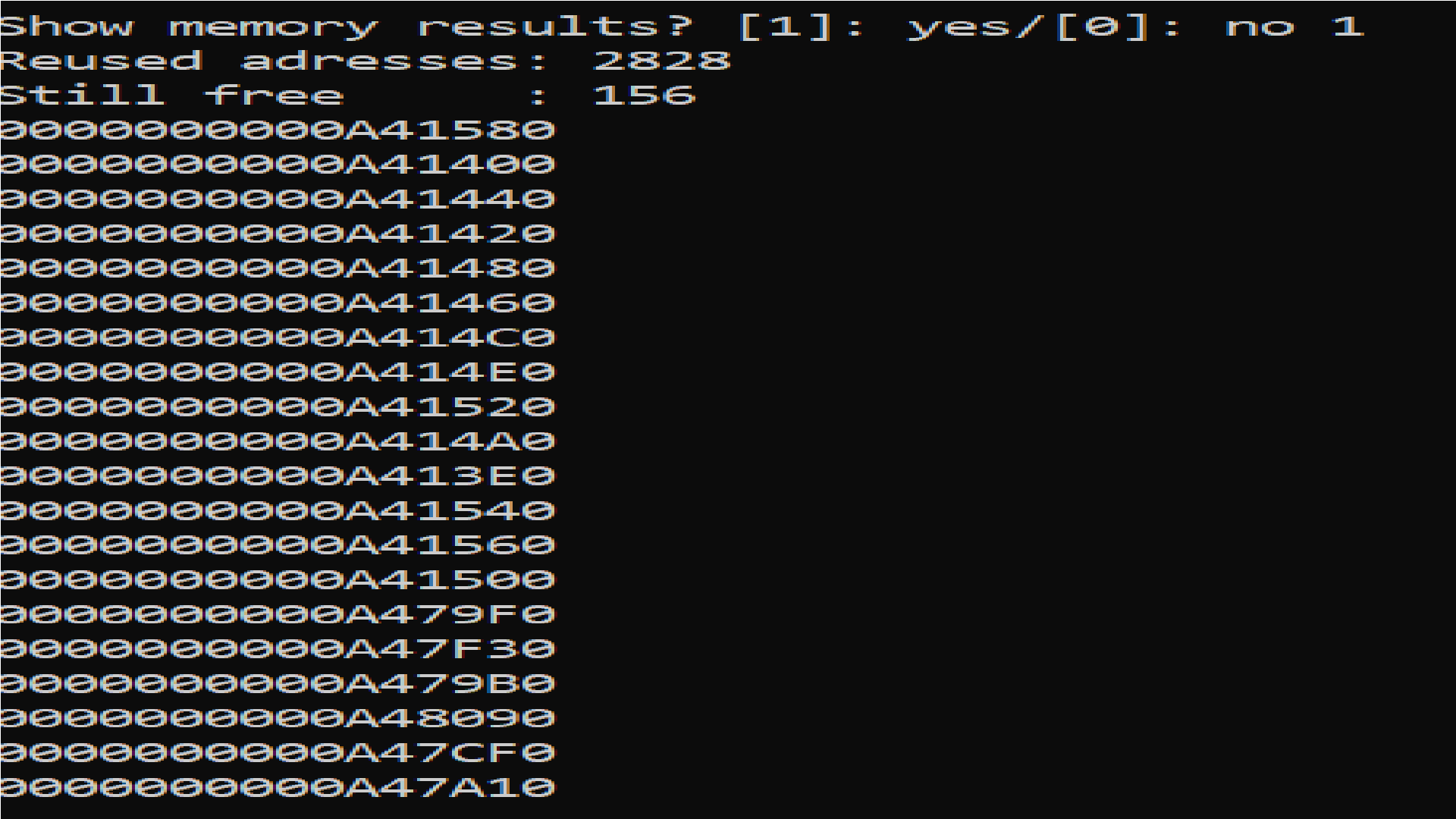


Стек со списком:



Вывод: эффективнее использовать массив. Дает заметный выигрыш по времени при больших размерах очереди.

Информация о адресе элементов очереди при удалении и добавлении элементов



4) 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. Чтобы этого избежать, можно использовать кольцевой массив, но будет сложнее реализовать операции работы с такой очередью. 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.