Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
«Обработка очередей»

Студент, группа **ИУ7-33Б**

Рядинский К.В.,

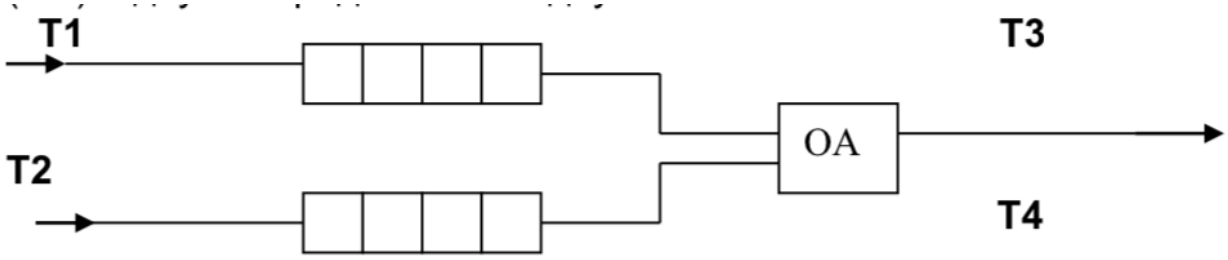
2020 г.

Условие задачи

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

Техническое задание

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 **T1** и **T2**, 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 **T3** и **T4**, 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.

Возможные аварийные случаи

1. Неправильный ввод пункта меню (должны быть только числа 0, 1, 2, 3)

Алгоритм

1. Выбор пункта меню
    1. Моделирование на списке
    2. Моделирование на массиве
    3. Сравнение операций добавления и удаления
2. Вывод результатов

Предварительный расчет моделирования

Так как среднее время прихода больше среднего времени обслуживания, то время моделирования будет определяться временем прихода заявок первого типа.

Так нам требуется продолжать моделирование, пока из ОА не выйдут 1000 элементов из первой очереди

1. Ожидаемое время моделирования — 1000 * ((4 + 1) / 2) = 3000
2. Время обработки 1000 заявок — 1000 * ((0 + 4) /2) = 2000
3. Тогда у нас остается 1000 ед времени, отсюда кол-во обработанных элементов из второй очереди будет примерно 1000 / ((0 + 1) / 2) = 500

Тесты

Ввод	Вывод
Ввод буквы в меню (a)	Неправильный ввод

Ввод несуществующего пункта меню (5)	Такого пункта меню нет

Структуры данных

Очередь, реализованная списком.

```
typedef struct queue_s
{
    int len;
    queue_node_t *head;
    queue_node_t *tail;
} lqueue_t;
```

Очередь, реализованная массивом.

```
typedef struct aqueue_s
{
    int len;
    int capacity;

    data_t *arr;
} aqueue_t;
```

Структура элемента очереди.

```
typedef struct data_s
{
    queue_time   last_entrance;
    queue_time   last_service;

    queue_time   entrance;
    queue_time   service;

    queue_type_t type;
} data_t;
```

Функции

Int add_queue(queue, elem) — добавление элемента в очередь

queue_elem pop_queue(queue) — удаление элемента из очереди

Int model(aq, lq, res) — моделирование

Void clean_queue(queue) — очистка очереди

Queue init_queue(void) — инициализация очереди

Оценка эффективности

Добавление в список (такты)	Добавление в массив (такты)
309	46

Удаление из списка (такты)	Удаление из массива (такты)
252	1500

Время моделирования (Среднее)

Моделирование на списке (нс)	Моделирование на массиве (нс)
4883000	3516000

Потребление памяти (Среднее потребление)

Моделирование на списке (Б)	Моделирование на массиве (Б)
102816	78320

Контрольные вопросы

1. Что такое очередь?

Очередь - структура данных, для которой выполняется правило FIFO, то есть первым зашёл - первым вышел. Вход находится с одной стороны очереди, выход - с другой.

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

При хранении массивом: кол-во элементов * размер одного элемента очереди. Память выделяется на стеке при компиляции, если массив статический. Либо память выделяется в куче, если массив динамический.
При хранении списком: кол-во элементов * (размер одного элемента очереди + указатель на следующий элемент). Память выделяется в куче для каждого элемента отдельно.

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

При хранении массивом память не освобождается, а просто меняется указатель на конец очереди. При хранении списком, память под удаляемый кусок освобождается.

4. Что происходит с элементами очереди при ее просмотре?
Эти элементы удаляются из очереди.

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

Зная максимальный размер очереди, лучше всего использовать статический массив. Не зная максимальный размер, стоит использовать связанный список, так как такую очередь можно будет переполнить только если закончится оперативная память.

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

При использовании линейного списка тратится больше времени на обработку операций с очередью, а так же может возникнуть фрагментация памяти. При реализации статическим кольцевым массивом, очередь всегда ограничена по размеру, но операции выполняются быстрее, нежели на списке.

7. Что такое фрагментация памяти?

Фрагментация памяти - разбиение памяти на куски, которые лежат не рядом друг с другом. Можно сказать, что это чередование свободных и занятых кусков памяти.

8. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на эффективное выделение и корректное освобождение динамической памяти. Помимо этого стоит обратить внимание на корректность реализации кольцевого массива, чтобы не произошло записи в невыделенную область памяти. Еще стоит обратить внимание на возникновение фрагментации памяти.

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

При запросе памяти, ОС находит подходящий блок памяти и записывает его в «таблицу» занятой памяти. При освобождении, ОС удаляет этот блок памяти из «таблицы» занятой пользователями памяти.

Вывод

Использование связанных списков невыгодно при реализации очереди. Списки проигрывают как по памяти, так и по времени обработки (проигрывает по времени примерно в 1.4 раза и потребляют памяти примерно в 1.4 больше). Но, когда заранее неизвестен максимальный размер очереди, то можно использовать связанные списки, так как в отличии от статического массива, списки ограничены в размерах только размером оперативной памяти. Стоит отметить, что при проведении тестов ни разу не наблюдалась фрагментация памяти, но даже при этом, список проигрывает во времени обработки массиву.