Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учр
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 **ИУ7-35Б** **Назиров И.В.,**

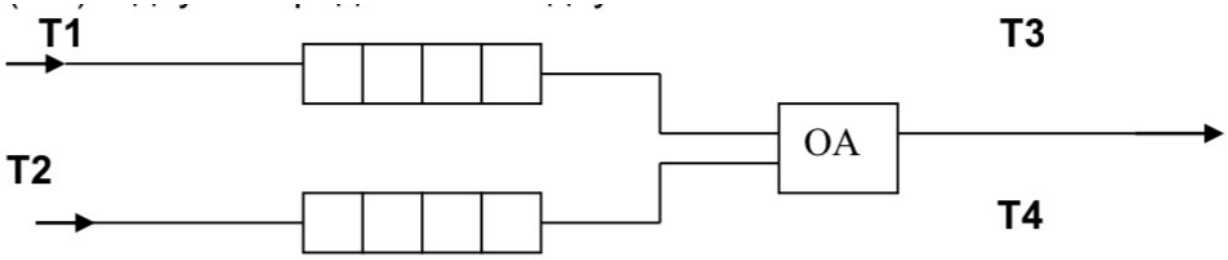
2021 г.

Условие задачи

Отработка навыков работы с типом данных «очередь», представленным в вид
одномерного массива и односвязного линейного списка. Сравнительный
анализ реализации алгоритмов включения и исключения элементов из очере
при использовании двух указанных структур данных. Оценка эффективности
программы (при различной реализации) по времени и по используемому объе
памяти

Техническое задание

Система массового обслуживания состоит из обслуживающего аппарата (ОА)
двух очередей заявок двух типов. Заявки 1-го и 2-го типов поступают в "хвос
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**
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
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:

Номер выполнения команды;
Возможность ввода времени(время в условных единицах).

Пример:
//ввод команды для очереди
>>1

Выходные данные

Время моделирования.

Время простоя.

Количество заявок вошедших в очередь 1 и 2 типа.

Количество заявок вышедших из очереди 1 и 2 типа.

Количество выкинутых заявок 2 типа из ОА.

Среднее время очереди 1 и 2 типа.

Время работы машины.

Адреса;Т

Для каждых обработанных 100 заявок:

- Текущая длина очереди 1 и 2 типа.
- Среднее время очереди 1 и 2 типа.

Возможные аварийные случаи

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

Алгоритм

1. Выбор пункта меню
    1. Моделирование на массиве
    2. Моделирование на списке
    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 node_el {
    double data;
    struct node_el *next;
} node;

typedef struct {
    node *head, *tail;
    size_t size;
} st_list_que;
```

Очередь, реализованная массивом.

```
typedef struct {
    double arr[MAX_SIZE_OF_QUE];
    size_t size;
} st_arr_que;
```

Функции

double arr_que_push(st_arr_que *arr_que, double value) —
добавление элемента в очередь массив

double arr_que_pop(st_arr_que *arr_que) -
удаление элемента из очередь массив

void arr_que_clear(st_arr_que *arr_que) -
очищение элементов очереди массива

void arr_que_init(st_arr_que *arr_que) -
инициализация очереди массива

double list_que_push(st_list_que *list_que, double value)
добавление элемента в очередь списка

double list_que_pop(st_list_que *list_que)
удаление элемента из очередь списка

void clear_list_que(st_list_que *list_que)
Очищение элементов очереди списка

void list_que_init(st_list_que *list_que)
инциализация очереди списка

Оценка эффективности

T1 = 1 до 5; T2 = 0 до 3;
T3 = 0 до 4; T4 = 0 до 1;

Количество вызовов 1000

Добавление в список (такты)	Добавление в массив (такты)
68	14

Удаление из списка (такты)	Удаление из массива (такты)
29	1952

Время моделирования (Среднее)

Моделирование на списке	Моделирование на массиве
2953.38	2973.73

Теоретическая ожидание времени моделирования

```
Theory results:
All time of modulation = 3000.000000
Wait time = -0.000000
in 1 type  = 1000.000000
in 2 type  = 2000.000000
```

Контрольные вопросы

1. Что такое FIFO и LIFO?

Очередь - структура данных, для которой выполняется правило FIFO, то есть первым зашёл - первым вышел. Вход находится с одной стороны очереди, выход - с другой. На стек действует правило LIFO — последним пришел, перв вышел.

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. От чего зависит эффективность физической реализации очереди?

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.Для чего нужен алгоритм «близнецов».

Метод близнецов (или buddy system) является схемой выделения памяти, сочетающ в себе возможность слияния буферов и распределитель по степени числа 21.
В основе метода лежит создание буферов малого размера путем деления пополам буферов и слияния смежных буферов по мере возможности. При разделении буфер два каждая половина называется близнецом (buddy) второй.

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.

10. **Какие дисциплины выделения памяти вы знаете?**

динамический и статический

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

1. Система входа - можете ли вы войти в систему после ввода правильных учетных данных.

2. Механизм работы. Выполняется ли работа в нескольких режимах плавно? Возвращаеться ли код ошибке при вводе неверных данных.

Оценка эффективности памяти

Список	Массив
47472	8000

Сравнение методов

По результатам выполнения программы, видно, что реализация очереди списком дает прирост по времени работы при 10000 элементов, т.к. в массиве при удалении происходит сдвиг всего массива на позицию влево. Если бы реализации удаления элементов в массиве была другой(удаление не 1 элементу), то возможно выиграш по времени был бы при реализации массивом. Но, когда заранее неизвестен максимальный размер очереди, то можно использовать связанный 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 (на моём ПК), но даже при этом, список проигрывает во времени обработки кольцевому массиву.

Вывод

Использование связанных списков невыгодно при реализации очереди. Списк
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). Н
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 массив