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)

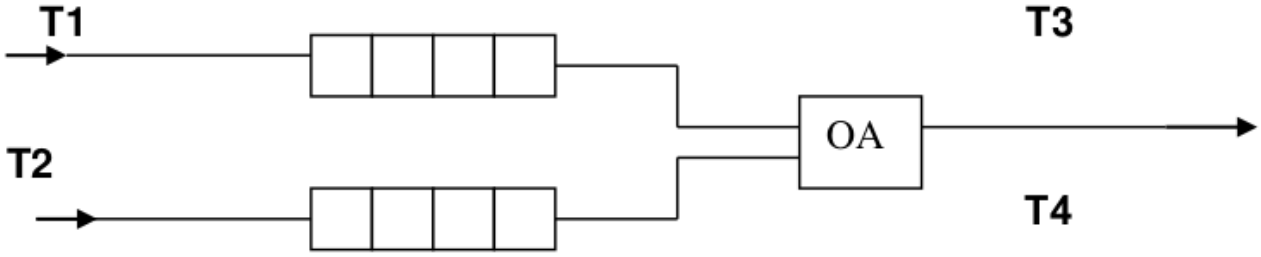
ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
«РАБОТА С ОЧЕРЕДЬЮ»

Студент, группа **Буланый К., ИУ7-36Б**

2019 г.

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 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.

Техническое задание

Входные данные:

1. **Целое число, представляющее собой номер команды:** целое число в диапазоне от **0** до **3**.
2. **Командно-зависимые данные:**
   - целочисленные значения (количество элементов очереди, интервалы характеристик ОА).

Выходные данные:

1. Результат моделирования работы ОА – данные о рабочем времени автомата, ожидаемое рабочее время автомата, погрешнсоть, число

вошедших заявок, число вышедших заявок, число необрботанных заявок, число срабатываний автомата, время простоя автомата, количество переиспользованных адресов и адресов, взятых из новой памяти (в случае реализации на списке)).

2. Количественная характеристика сравнения вариантов реализации очереди.

Функция программы: программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Ввести данные обслуживающего аппарата и вывести статистику работы.
2. Вывести количественную характеристику выполнения операций над очередью.

Обращение к программе: запускается из терминала.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 4 или меньшее, чем 0.
   На выходе: сообщение «Wrong command.»
2. Некорректный ввод количества элементов очереди.
   На входе: отрицательное целое число, число, превышающее максимально допустимое число для количества элементов стека или буква.
   На выходе: сообщение «Negatives were input.»
3. Некорректный ввод характеристик обслуживающего аппарата очереди.
   На входе: буква или, любой другой нечисловой символ или отрицательное число.
   На выходе: сообщение «Wrong parameters.»
4. Некорректный ввод характеристик обслуживающего аппарата очереди.
   На входе: правая граница интервала больше левой.
   На выходе: сообщение «Min is greater than max.»

Структуры данных

Хранение заявки:

```
typedef struct
{
   void *data;
} task_t;
```

Поля структуры:

- ***void data* –** каие-либо данные.

Реализация очереди на основе массива:

```
typedef struct
{
    size_t cap;
    size_t size;
    size_t front;
    size_t rear;
    task_t *arr;
} arrq_t;
```

Поля структуры:

- ***size_t* cap, size, rear, front –** максимальный допустимый размер, текущий размер, хвост и голова очереди;
- ***task_t* *arr –** указатель на массив заявок;

Реализация элемента очереди:

```
typedef struct qnode qnode_t;

struct qnode
{
    task_t task;
    qnode_t *next;
};
```

Поля структуры:

- ***task_t* task –** заявка ОА;
- ***qnode_t* *next –** указатель на следующий элемент очереди;

Реализация очереди на основе линейного односвязного списка:

```
typedef struct
{
    size_t cap;
    size_t size;
    qnode_t *front;
    qnode_t *rear;
} listq_t;
```

Поля структуры:

- ***size_t* cap, size –** максимальный допустимый размер и текущий размер очереди;
- ***qnode_t* *front, *rear –** указатели на голову и хвост очереди;

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с двумя реализациями очереди – на основе массива или на основе линейного односвязного списка.
3. При выборе команды вывода количественной характеристики выполнения операций над очередью, выводится среднее значение добавления/удаления элементов из очереди на основе 1000 добавлений/удалений.
4. При выборе команды вывода статистики модели ОА, выводится статистика ОА после обработки каждой 100 заявки, а также общие данные.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод комманды	5	Wrong command.
2	Некорректный ввод количества элементов очереди	-1	Wrong command.
3	Некорректный ввод количества элементов очереди	A	Wrong parameters.
4	Некорректный ввод характеристики ОА	Ввод интервала прихода: A 8	Wrong parameters.
5	Некорректный ввод характеристики ОА	Ввод интервала прихода: 0 B	Wrong parameters.
6	Некорректный ввод характеристики ОА	Ввод интервала прихода: 8 7	Min is greater than max.
7	Корректный ввод характеристик ОА	10000; 0 6; 0 1; 5	Вывод информации ОА и количественная характеристика моделирования
9	Вывод количественной	Выбор команды	**Время**

	характеристики выполнения операций над очередью		

Оценка эффективности

Время добавления элемента (в тактах процессора):

Массив	Список
145	244

Время удаления элемента (в тактах процессора):

Массив	Список
146	503

Объём занимаемой памяти (в байтах):

Количество элементов	Массив	Список
10	112	192
100	832	1632
1000	8032	16032

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 кольцевым 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 кольцевым 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 кольцевой 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

Как и в прошлой ЛР список проигрывает как и в производительности, так и в объеме занимаемой памяти. Однако ровно также списко выгоден, если неизвестен максимальный размер очереди.