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»

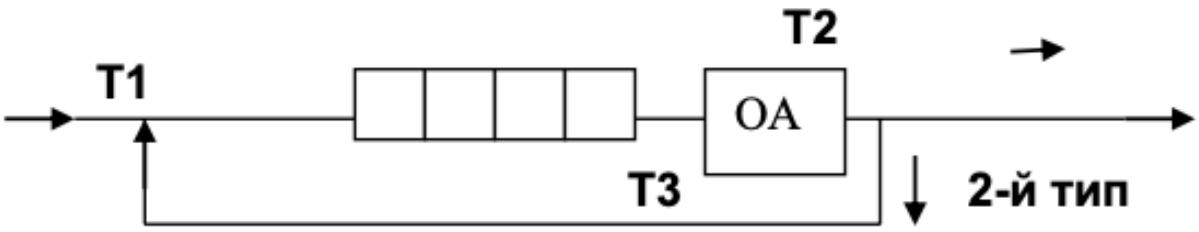
Студент, группа **Артемьев И.О., ИУ7-33Б**

2020 г.

Описание условия задачи

Смоделировать процесс обслуживания первых 1000 заявок **1-го типа.** Выдавать после обслуживания каждых 100 заявок **1-го типа** информацию о текущей и средней длине очереди, количестве вошедших и вышедших заявок и о среднем времени пребывания заявок в очереди. В конце процесса выдать общее время моделирования, время простоя аппарата, количество вошедших в систему и вышедших из нее заявок первого типа и количество обращений заявок второго типа. По требованию пользователя выдать на экран адреса элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Система массового обслуживания состоит из обслуживающего аппарата (ОА) и очереди заявок двух типов.



Заявки 1-го типа поступают в "хвост" очереди по случайному закону с интервалом времени **T1,** равномерно распределенным от **0 до 5** единиц времени (е.в.). В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время **T2** от **0 до 4** е.в., после чего покидают систему.

Единственная заявка 2-го типа постоянно обращается в системе, обслуживаясь в ОА равновероятно за время **T3** от **0 до 4** е.в. и возвращаясь в очередь не далее 4-й позиции от "головы". В начале процесса заявка 2-го типа входит в ОА, оставляя пустую очередь. (Все времена – **вещественного** типа)

Техническое задание

Входные данные:

1. **Целое число, представляющее собой номер команды:** целое число в диапазоне от **0** до **4**.
2. **Командно-зависимые данные:**
   - целочисленные значения (количество элементов очереди).

Выходные данные:

1. Время для добавления и удаление элемента из очереди в разных реализациях.
2. Память, затраченная на количество элементов в разных реализациях.

3. Фрагментация памяти.
4. Результат моделирования работы ОА – данные о времени прихода, ожидаемое время прихода, погрешность, данные о рабочем времени автомата, ожидаемое рабочее время автомата, погрешность, число вошедших заявок, число вышедших заявок, число срабатываний автомата, время простоя автомата, количество обращений заявок второго типа.

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

1. Выйти.
2. Вывести время для добавления и удаление элемента из очереди в разных реализациях.
3. Вывести память, затраченную на введенное количество элементов в разных реализациях.
4. Отследить фрагментацию памяти.
5. Смоделировать процесс обслуживания первых 1000 заявок.

Обращение к программе: запускается из терминала командой ./app.exe .

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 4 или меньшее, чем 0.
   На выходе: сообщение «Ошибка: неверный пункт меню»
2. Некорректный ввод количества элементов очереди.
   На входе: отрицательное целое число, число, превышающее максимально допустимое число для количества элементов очереди или буква.
   На выходе: сообщение «Ошибка: введено неверное количество элементов»

Структуры данных

Реализация очереди на основе массива:

```
typedef struct
{
   int *data;
   /*
```

```
   *   first element point
   *   second element point
   */
   size_t felp, lelp;
} vector_q;
```

Поля структуры:

- ***data–*** указатель на массив элементов;
- ***felp, lelp* –** индексы начального и конечного элемента очереди;

Реализация очереди на основе линейного односвязного списка:

```
typedef struct node_t
{
   int data;

   struct node_t *next;
} node_t;
```

Поля структуры:

- ***data* –** значение элемента узла;
- ****next* –** указатель на следующий узел списка;

Реализация массива свободных областей:

```
typedef struct
{
   node_t *free_array_points[MAX_LEN_ARRAY_FREE];

   int len;
} free_points_t;
```

Поля структуры:

- ***free_array_points[MAX_LEN_ARRAY_FREE] –** указатель на массив адресов; **MAX_LEN_ARRAY_FREE** = 10
- ***len* –** длина массива;

Алгоритм

1. Пользователь вводит номер команды из меню.

2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия 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 ОА, выводится статистика ОА после обработки каждой 100 заявки, а также общие данные, описанные в секции “Выходные данные”.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод комманды	5	Ошибка: неверный пункт меню
2	Некорректный ввод комманды	a	Ошибка: неверный пункт меню
3	Некорректный ввод количества элементов очереди	0	Ошибка: введено неверное количество элементов
4	Некорректный ввод количества элементов очереди	a	Ошибка: введено неверное количество элементов

Оценка эффективности

Измерения эффективности реализаций очереди будут производиться в единицах измерения – тактах процессора. При записи результатов использовалось среднее количество тактов, полученное по результатам 1000 измерений.

Время добавления элемента (в тактах процессора):

Массив	Список
42	213

Время удаления элемента (в тактах процессора):

Массив	Список
22	333

Объём занимаемой памяти (в байтах):

Количество элементов	Массив	Список
10	40	200
100	400	2000
1000	4000	20000

Моделирование работы автомата

Время прихода: 2523.195078 (**ожидаемое время прихода**: 2500.000000, **погрешность**: 0.927803%)
Время работы автомата: 1969.482311 (**ожидаемое время работы автомата**: 2000.000000, **погрешность**: 1.525884%)
Число вошедших заявок: 1002 (для первого типа)
Число вышедших заявок: 1000 (для первого типа)
Число срабатываний автомата: 1000 (для первого типа)
Время простоя автомата: 552.000092
Количество обращений заявок второго типа: 655

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?

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. В каком случае лучше реализовать очередь посредством указателей, а в каком – массивом?

Если заранее известно кол-во элементов и оно не более чем половина выделенного места под статический массив, то массивом, иначе списком.

7. 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.

8. 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.

9. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на переполнение очереди (при реализации очереди в виде массива) и на фрагментацию (при реализации очереди в виде списка)

10. 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 (при 1000 элементах 20000 байт, а у вектора при том же размере 4000), так и по времени обработки (добавление за 213 тиков и удаление за 333 тика, а у вектора добавление за 42 тика и удаление за 22 тика). 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.