



Министерство науки и высшего образования 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»

Кафедра «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«Обработка очередей»

по курсу «Типы и структуры данных»

Студент: Шимшир Эмирджан Османович

Группа: ИУ7-33Б

Студент

Шимшир Э.О.

подпись, дата

фамилия, и.о.

Преподаватель

Барышникова М.Ю.

подпись, дата

фамилия, и.о.

Оценка _____

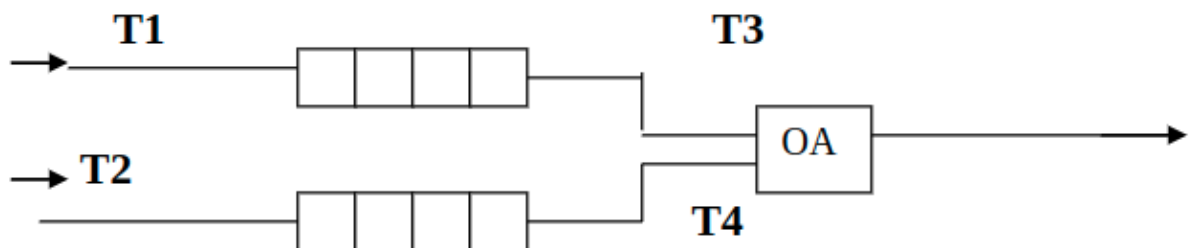
Условие задачи

Разработать программу работы со стеком, реализующую операции Система массового обслуживания состоит из обслуживающих аппаратов (ОА) и очередей заявок двух типов, различающихся временем прихода и обработки. Заявки поступают в очереди по случайному закону с различными интервалами времени (в зависимости от варианта задания), равномерно распределенными от начального значения (иногда от нуля) до максимального количества единиц времени.

В ОА заявки поступают из «головы» очереди по одной и обслуживаются за указанные в задании времена, распределенные равномерно от минимального до максимального значений (все времена – вещественного типа).

Требуется смоделировать процесс обслуживания первых 1000 заявок первого типа, выдавая после обслуживания каждой 100 заявок первого типа информацию о текущей и средней длине каждой очереди и о среднем времени пребывания заявок каждого типа в очереди. В конце процесса необходимо выдать на экран общее время моделирования, время простоя ОА, количество вошедших в систему и вышедших из нее заявок первого и второго типов.

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 T_1 и T_2 , 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 равномерно за времена T_3 и T_4 , 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 немедленно поступает на обслуживание; обработка заявки 2-го типа прерывается и она возвращается в "хвост" своей очереди (система с абсолютным приоритетом и повторным обслуживанием).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок 1-го типа 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, среднем времени пребывания заявок в очереди, количестве «выброшенных» заявок второго типа. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Входные данные:

1. Целое число, представляющее собой пункт меню:

целое число в диапазоне от 0 до 10

2. Дополнительный ввод: поле типа `int` в зависимости от требования

Выходные данные:

1. Результат выполнения команды

2. Сообщение об ошибке (при ее возникновении)

Описание меню и функций программы

```
emir_shimshir@mbp-emir ~/D/g/B/lab_05 (lab_05)> ./app.exe
Программа для работы с очередью
Шимшир Эмирджан ИУ7-33Б

(необходимо вводить только целые числа)

Пункты меню:
Массив
1. Добавить элемент
2. Удалить элемент
3. Распечатать очередь
Список
4. Добавить элемент
5. Удалить элемент
6. Распечатать очередь
7. Распечатать распечатать массив освободившихся адрессов
Выполнить задание
8. Для массива
9. Для списка
10. Замеры времени и памяти
0. Выход из программы

Введите пункт меню: █
```

Очередь в виде массива:

1. Добавить элемент
2. Удалить элемент
3. Распечатать очередь

Очередь в виде списка:

4. Добавить элемент
5. Удалить элемент
6. Распечатать очередь
7. Распечатать массив освободившихся адрессов

Выполнить задание

8. Для массива
9. Для списка
10. Замеры времени и памяти
0. Выйти из программы

Аварийные ситуации:

1. Неверно введен пункт меню
(не число или число меньше 0 или больше 10)
2. Достигнут максимально возможный размер очереди (как для массива, так и для списка)
(добавлено более 1 тысячи элементов)
3. Неверно введено число элементов для теста по времени
(не число или число меньше 1 или больше 1000)

Описание внутренних структур данных

Структура для хранения элемента очереди, реализованной в виде списка:

```
typedef struct node_t
{
    int elem;

    struct node_t *next;
} node_t;
```

int elem — значение текущего элемента в очереди

*struct node_t *next* — указатель на следующий элемент очереди

Структура для хранения указателя на голову очереди (*node_t *head*) и хвост (*node_t *tail*), а также размера очереди, реализованной в виде списка:

```
typedef struct list_t
{
    node_t *head;
    node_t *tail;

    int size;
} list_t;
```

Структура для хранения очереди, реализованной в виде массива:

```
#define MAX_ELEMS_COUNT_IN 10000
#define MAX_ELEMS 1000

typedef struct arr_t
{
    int arr[MAX_ELEMS_COUNT_IN];
    int begin;
    int end;
    int size;
} arr_t;
```

int arr[] — массив элементов очереди

int begin — голова очереди

int end — хвост очереди

int size — размер очереди

MAX_ELEMS_COUNT_IN - максимальное количество элементов в списке

MAX_ELEMS – максимальное число элементов для измерения времени и обработки

Взаимодействие с программой

1. Выводится меню программы
2. Пользователь вводит номер любой команды, которой соответствует свое назначение
3. Ввод осуществляется, пока не будет совершена ошибка при вводе (аварийная ситуация) или пока не будет введен 0 (означает выход из программы)

Набор тестов

	Название теста	Пользовательский ввод	Результат
1	Некорректный ввод пункта меню	test	Ошибка: пункты меню это числа от 0 до 10
2	Некорректный ввод количества элементов для замеров памяти и времени (не число)	test	Ошибка: неверно введено количество элементов
3	Некорректный ввод количества элементов для замеров памяти и времени (число меньше 1 или больше 1000)	-1 или 1234	Ошибка: неверно введено количество элементов
4	Добавить элемент в очередь (массив и список)	команда 1 или 4	Элемент добавлен в очередь
5	Удалить элемент из очереди (массив и список)	команда	Элемент удален из очереди

		2 или 5	
6	Распечатать очередь (массив и список)	команда 3 или 6	Распечатанная очередь
7	Переполнение очереди (массив и список)	Попытка добавить элемент в очередь, если уже добавлено 1000 элементов	Ошибка: очередь переполнена
8	Очередь пуста (массив и список)	Попытка распечатать или удалить элемент из очереди, если в ней ничего нет	Очередь пустая
9	Распечатать массив освободившихся адресов (список)	команда 7	Печатается массив освободившихся адресов
10	Распечатать пустой массив адресов (список)	Попытка распечатать массив освободившихся адресов, если он пуст	Массив освободившихся адресов пуст
11	Выполнить задачу (массив)	команда 8	Вывод результатов выполнения задания со всем временем

12	Выполнить задачу (список)	команда 9	Вывод результатов выполнения задания со всем временами
13	Вывод замеров времени и памяти	команда 10 элементов 10	Результат замеров времени добавления и удаления элементов и замеров памяти
14	Выход из программы	команда 0	Выход из программы, очистка консоли

Оценка эффективности

Результаты измерения времени и памяти при различном размере списков. Для каждого случая проводилось 5 экспериментов. При проведении экспериментов программа была скомпилирована без оптимизаций (-O0), внешние задачи отсутствовали.

Добавление элементов (в тиках)

Размер	Массив	Список
10	575	954
100	4754	10914
500	16183	39020
1000	39033	95781

Удаление элементов (в тиках)

Размер	Массив	Список
10	274	762
100	2008	4762
500	6942	17244
1000	13751	34534

Замеры памяти (в байтах)

Размер	Массив	Список
10	4012	160
100	4012	1600
500	4012	8000
1000	4012	16000

Тестирование задания

Время моделирования заявок

T1: от 1 до 5 T2: от 0 до 3

T3: от 0 до 4 T4: от 0 до 1

Теоретические вычисления:

Время моделирования = 3000 е.в.

(среднее время прихода заявки 1 типа или среднее время обработки заявки 1 типа) * (количество) – Выбирается большее время.

Число заявок 1 типа, вошедших = 1000, вышедших = 1000

(По условию)

Число заявок 2 типа, вошедших: 2000, вышедших = 2000

(время моделирования (3000 е.в.) / на максимальное из среднего времени прихода заявки 2 типа и обработки ОА (1.5 е.в. или 0.5 е.в.))

Практические вычисления:

```
Общее время моделирования = 3044.607422
Погрешность моделирования = 1.486914

Заявок вошло в 1ую очередь = 1000
Заявок 1ой очереди вышло = 1000
Среднее время обработки заявки в 1ой очереди (ожидаемое) = 3.000000

Заявок вошло во 2ую очередь = 2039
Заявок 2ой очереди вышло = 1654
Среднее время обработки заявки в 2ой очереди (ожидаемое) = 1.500000
Заявок 2го типа вернулось обратно в конец очереди = 1276

Погрешность ввода заявок в 1ую очередь 1.465129 процентов
Погрешность ввода заявок во 2ую очередь 0.456301 процентов

Время простоя = 1.551514

Введите пункт меню:
```

Ответы на контрольные вопросы

1. Что такое очередь?

Очередь — структура данных, для которой выполняется правило First In First Out, 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 реализации кольцевым массивом

$(\text{кол-во элементов}) * \text{sizeof}(\text{элемента})$

Если массив статический, то память выделяется на стеке при компиляции, если массив динамический, то память выделяется в куче

При реализации списком

$\text{sizeof}(\text{элемента}) + 8 \text{ байт (для указателя)}$

Память выделяется в куче, для каждого элемента отдельно

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

При использовании массива память не освобождается, лишь сдвигается указатель, а при реализации списком — переходит указатель и освобожденный элемент очищается из памяти

4. Что происходит с элементами очереди при ее просмотре?

Просматриваемый элемент удаляется из очереди

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

Все зависит от того, чем ограничено выполнение задачи

Если временем, то лучше использовать реализацию на массиве, так как операции добавления и удаления на нем выполняются быстрее

Если памятью, то список лучше, так он ограничен лишь объемом оперативной памяти, в то время как массив ограничен размером стека

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

При важности скорости работы программы — стоит выбрать массив из-за быстрого выполнения операций, но если известно количество элементов должно войти в очередь, так как он не ограничен по памяти

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

При использовании кольцевого массива 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.

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?

На корректное освобождение памяти

10. Что происходит с элементами очереди при ее просмотре?

При запросе памяти, операционная система находит подходящий блок памяти и записывает его в «настоящее» начало массива таблицы занятой памяти. При освобождении, ОС удаляет этот блок памяти из «настоящего» начала массива таблицы занятой пользователем памяти.

Вывод

При реализации очереди стоит учитывать два основных критерия – память и время.

Если реализовывать очередь на массиве, то он будет ограничен по памяти размером стека, в то время как список ограничен объемом оперативной памяти.

Если реализовать очередь на списке, то операции добавления и удаления элементов будут на нем происходить дольше из-за постоянной необходимости выделения памяти под следующий элемент списка

Массив быстрее списка

при удалении примерно в 3 раза

при добавлении примерно в 2 раза