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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

«Обработка очередей»

Студент Маслова Марина Дмитриевна
фамилия, имя, отчество

Группа ИУ7-33Б

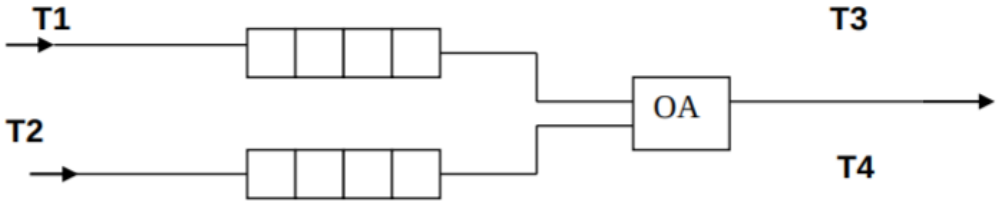
2020 г.

Оглавление

Техническое задание

Условие 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 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 любого типа может войти в ОА, если:

а) она вошла в пустую систему;

б) перед ней обслуживалась заявка ее же типа;

в) перед ней из ОА вышла заявка другого типа, оставив за собой пустую очередь (система с чередующимся приоритетом).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок 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. По требованию пользователя выдать на экран адреса элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти

Входные данные

Целое число, задающее выбор пункта меню.

Целое число, задающие количество добавляемых и удаляемых элементов.

Выходные данные

Адреса элементов очереди на массиве и списке. Результаты моделирования. Количественные характеристики оценки эффективности программы.

Задачи, реализуемые программой

1. Добавление элементов в очередь на массиве и списке.
2. Удаление элементов из очереди на массиве и списке.
3. Вывод адресов элементов очередей на массиве и списке на экран.
4. Моделирование

Возможные аварийные ситуации и ошибки пользователя

Некорректный ввод пункта меню.

Некорректный ввод целочисленных данных.

Переполнение очереди.

Попытка вывода адресов элементов пустой очереди или удаления элементов из нее.

Ошибка выделения памяти.

Описание внутренних структур данных

Для краткости записи используется беззнаковый тип **uint**:

```
typedef unsigned int uint;
```

Для хранения типа заявки используется перечисляемый тип:

```
typedef enum order_type
{
    FIRST,
    SECOND
} order_type_t;
```

Для хранение заявки используется структура:

```
typedef struct order
{
    order_type_t type;
    double coming;
    double processing;
} order_t;
```

Для хранения очереди на массиву используется структура:

```
typedef struct
{
     order_t array[MAX_ARR_LEN];
     uint length;
     uint in;
     uint out;
} arr_queue_t;
```

Для хранения узла очереди на списке используется структура:

```
typedef struct queue_element queue_element_t;
struct queue_element
{
    order_t data;
    queue_element_t *prev;
};
```

Для хранения очереди на списке используется структура:

```
typedef struct list_queue
{
    queue_element_t *in;
    queue_element_t *out;
    uint length;
} list_queue_t;
```

Описание функций

```
/**
 * Удаляет элемент очереди на массиве
 * queue – указатель на очередь;
 * order – указатель, куда записывается удаленный элемент;
 * Возвращает код ошибки
 */
int aq_pop(arr_queue_t *queue, order_t *const order);
```

```
/**
 * Добавляет элемент в очередь на массиве
 * queue – указатель на очередь;
 * order – добавляемый элемент;
 * Возвращает код ошибки
 */
int aq_push(arr_queue_t *queue, const order_t order);
/**
 * Выводит адреса элементов очереди на массиве
 * queue – указатель на стек;
 * Возвращает код ошибки
 */
int aq_print(arr_queue_t *queue);
/**
 * Удаляет элемент очереди на списке
 * queue – указатель на очередь;
 * order – указатель, куда записывается удаленный элемент;
 * Возвращает код ошибки
 */
int lq_pop(list_queue_t *queue, order_t *const order);
/**
 * Добавляет элемент в очередь на списке
 * queue – указатель на очередь;
 * order – добавляемый элемент;
 * Возвращает код ошибки
 */
int lq_push(list_queue_t *queue, const order_t order);
/**
 * Выводит адреса элементов очереди на списке
 * queue – указатель на вершину стека;
 * Возвращает код ошибки
 */
int lq_print(list_queue_t *queue);
/**
 * Генерирует вещественное число от min до max
 * Возвращает случайное число от min до max
 */
double generate_double(double min, double max)
```

```
/**
 * Моделирует систему массового обслуживания
 */
void modeling(void)
```

Описание алгоритма

При добавлении и удалении элемента из очереди на массиве индексы входа и выхода из очереди циклически сдвигаются, а необходимый элемент вставляется или удаляется из очереди.

При добавлении элемента в очередь на списке выделяется память под этот элемент, указатель на последний вошедший элемент и на вход в очередь переставляется на новый элемент. При удалении элемента из очереди освобождается память из-под этого элемента, а указатель на выход из очереди переставляется на предыдущий элемент.

Моделирование реализуется циклом пока количество обработанных заявок первого типа не будет равно 1000, каждая итерация цикла соответствует определенному моменту времени, на каждой итерации цикла происходит проверка на выход какой-либо заявки из ОА (тогда соответствующий счетчик заявок увеличивается на 1), проверка на приход заявок в одну из очередь и проверка на переход заявки из своей очереди в ОА. При совпадении текущего времени и времени перехода заявки в то или иное состояние, происходит добавление, перемещение или удаление заявки из системы.

Тесты

№	Что проверяется	Входные данные	Результат
1	Некорректный ввод пункта меню	jk	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
2	Некорректный ввод пункта меню (длинная строка)	hjkl	Некорректный ввод кода действия. Попробуйте ещё раз.

3	Некорректный ввод пункта меню (< 0)	-1	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
4	Некорректный ввод пункта меню (> 7)	9	Введенный код не соответствует ни одному действию. Попробуйте ещё раз.
5	Неверный ввод количества удаляемых элементов (не числовые данные, пункты 1, 2, 4, 5)	1 или 2 или 4 или 5 j	Введенные данные нецелочисленные!
6	Неверный ввод количества удаляемых элементов (неположительное число, пункты 1, 2, 4, 5)	1 или 2 или 4 или 5 -7	Введенное количество выходит за допустимый диапазон!
7	Неверный ввод количества удаляемых элементов (количество больше количества элементов в очереди, пункты 2 и 5)	2 или 5 4 (при трех элементах в очереди)	Введенное количество выходит за допустимый диапазон!
8	Неверный ввод количества добавляемых элементов элементов (количество больше количества элементов в очереди, пункты 2 и 5)	1 или 4 11 (при максимальном размере очереди 10)	Введенное количество выходит за допустимый диапазон!

Оценка эффективности

Добавление элементов (такты):

Количество элементов	Массив	Список
10	5	10
100	9	37
1000	38	187

Удаление элементов (такты):

Количество элементов	Массив	Список
10	6	8
100	8	11
1000	34	74

Моделирование (такты):

	Массив	Список
Время	1033160	1042619
Память	48000	94784

Используемая память (байт):

Количество элементов	Массив	Список
10	48000	320
100	48000	3200
1000	48000	32000

Оценка правильности моделирования

В системе обслуживания 2 очереди и один автомат.

Время прихода заявок больше времени их обработки, поэтому время моделирования определяется временем прихода заявок. Так как заявки в очереди приходят параллельно, а время прихода заявок первого типа больше времени прихода второго, то время моделирования:

3 * 1000 = 3000 е. в.

Заявки в первую очередь приходят в среднем за 3 е. в., во вторую за 1.5 е. в.. то есть при приходе 1000 заявок 1-ого типа, во вторую очередь придет в 2 раза больше, то есть 2000. Однако заявки второго типа в среднем обрабатываются 0.5 е. в., против 2 е. в. обработки заявок первого типа. Таким образом, время моделирования должно составит: 1000 * 2 + 2000 * 0.5 = 3000 е. в.

Погрешность времени моделирования:

(3067.105-3000)/3000=0.02 (или 2%), что соответствует погрешности в 2-3%.

Также можно заметить, что в теоретических расчетах не учитывается время простоя автомата, что и дает погрешность в 2%.

Контрольные вопросы

Что такое очередь?

Очередь – структура данных с доступом к элементам «первым зашел – первым вышел».

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

При реализации очереди на статическом массиве память выделяется один раз на стеке под максимальное количество элементов в очереди.

При реализации очереди списком память выделяется под каждый элемент отдельно при его поступлении в очередь. Для каждого элемента также выделяется память под указатель на предыдущий элемент.

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

При хранении очереди на статическом массиве память не освобождается, циклически перемещается указатель на выход из очереди. При хранении очереди списком, при удалении элемента указатель на выход из

очереди перемещается на предыдущий элемент, а память из-под удаляемого элемента освобождается.

Что происходит с элементами очереди при ее просмотре?

При просмотре очереди элементы удаляются из нее.

Каким образом эффективнее реализовывать очередь. От чего это зависит?

Очередь на список работает медленнее и под каждый элемент требуется больше памяти, чем при хранении очереди на статическом массиве. Однако при хранении списком количество элементов в очереди ограничено только размером оперативной памяти, и список при количестве элементов до 1500 выигрывает по памяти.

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

Реализовывать очередь посредством указателей лучше, когда количество элементов в ней неизвестно и может быть настолько большим, что сохранить очередь на стеке не получится. Иначе лучше реализовывать очередь на массиве, так как он работает быстрее.

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

При реализации очереди на статическом массиве её размер ограничен всегда ограничен размером массива. При реализации же очереди на списке при добавлении и удалении элемента происходит выделении или освобождение памяти, что сказывается на времени работы операций.

Что такое фрагментация памяти?

При фрагментации памяти между занятыми областями памяти находится свободные области, то есть память как бы разбита на занятые и свободные фрагменты.

На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на верную работу с выделением и освобождением памяти.

Каким образом физически выделяется и освобождается память при динамических запросах?

При выделении памяти находится область необходимого размера и помечается, как занятая. При освобождении памяти область помечается, как свободная.

Вывод

Таким образом, операции над очередью на массиве выполняются быстрее в 2-5 раз в зависимости от количества элементов в ней. При количестве элементов до 1500 список выигрывает по памяти у статического массива в 1.5-150 раз в зависимости от количества обрабатываемых элементов, но при больших количествах выигрывает статический массив, что можно наблюдать при моделировании.

По выводу адресов элементов очереди на списке можно сделать вывод, что, если один элемент был удален, то следующих добавленный встает на его место и фрагментации памяти не происходит.

При моделировании получено ожидаемое время с погрешностью 2%, что говорит об успешности проделанной работы.