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» (ИУ)

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«ОБРАБОТКА ОЧЕРЕДЕЙ»

по курсу:

«ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Вариант: 1

Студент:

Авдейкина Валерия Павловна, группа ИУ7-33Б _____

(подпись, дата)

Руководители:

Преподаватель ИУ7 _____

Силантьева Александра Васильевна (подпись, дата)

Преподаватель ИУ7

Барышникова Марина Юрьевна _____

(подпись, дата)

Проверяющий:

_____

(подпись, дата)

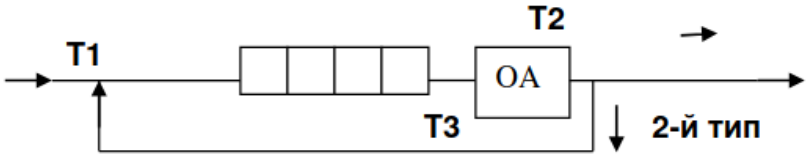
Оценка: _____

2022 г.

Оглавление

Описание условий задачи

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).

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.

Техническое задание

1 Входные данные

- Входные данные при вводе номера выбранной опции меню:
  - Целые беззнаковые числа [0; 9]
- Входные данные для различных опций меню:

Номер опции	Название опции	Входные данные
0	Выход	-
1	Добавить элемент (МАССИВ)	Значение элемента
2	Удалить элемент (МАССИВ)	-
3	Показать содержимое очереди (МАССИВ)	-
4	Добавить элемент (СПИСОК)	Значение элемента
5	Удалить элемент (СПИСОК)	-
6	Показать содержимое очереди (СПИСОК)	-
7	Оценка фрагментации	-
8	Моделирование работы аппарата (СПИСОК)	-
9	Сравнить эффективность работы очередей	Размер

Таблица 1: Входные данные опций меню

- Размер: натуральное число
- Значение элемента: целое число в указанных пределах

2 Выходные данные

- Выходные данные при вводе номера выбранной опции меню:
  - Выходные данные команды в случае корректного ввода
- Выходные данные для различных опций меню:

Номер опции	Название опции	Выходные данные
0	Выход	Сообщение
1	Добавить элемент (МАССИВ)	Сообщение об успехе / Сообщение об ошибке
2	Удалить элемент (МАССИВ)	Сообщение об успехе / Сообщение об ошибке
3	Показать содержимое очереди (МАССИВ)	Содержимое
4	Добавить элемент (СПИСОК)	Сообщение об успехе / Сообщение об ошибке
5	Удалить элемент (СПИСОК)	Сообщение об успехе / Сообщение об ошибке
6	Показать содержимое очереди (СПИСОК)	Содержимое
7	Оценка фрагментации	Данные
8	Моделирование работы аппарата (СПИСОК)	Данные
9	Сравнить эффективность работы очередей	Данные

Таблица 2: Выходные данные опций меню

3 Задача, реализуемая программой

Программа реализует обработку очереди и моделирование обработки 1000 заявок.

4 Способ обращения к программе

Программа вызывается в командной строке без каких-либо аргументов. Ввод данных производится с клавиатуры после соответствующего приглашения к вводу («`Команда...`» / «`Введите..`»).

5 Возможные аварийные ситуации и ошибки со стороны пользователя

1. Некорректный ввод номера опции:
    1.1. Недопустимые символы (*поведение программы не определено*)
    1.2. Недопустимый номер, пустой ввод
2. Некорректный ввод элемента
3. Некорректный ввод размера

Описание внутренних структур данных

В ходе работы были составлены следующие структуры:

1. Очередь в виде массива (вектора):

```
struct array_queue
{
   int *data;
   size_t size;
   size_t begin;
   size_t end;
   size_t max_size;
};
```

Итоговый размер структуры: 40 Б

2. Очередь в виде линейного односвязного списка (представлена структура элемента очереди):

```
typedef struct node
{
    int value;
    struct node *next;
} node_t;

struct list_queue
{
    node_t *head;
    node_t *tail;
};
```

Итоговый размер структуры: 16 Б

Результаты моделирования

- Ожидаемое время работы = (0 + 4) / 2 * 1000 + (0 + 4) / 2 * 1000 / 3 ≈ 2666.7 е.в.
- Так как очередь заполняется постоянно, ожидаемое время заполнения очереди = ожидаемое время работы

```
ИТОГ
Время прихода (е.в.): 2729.467 (Ожидаемое: 2666.667, погрешность: 2.355%)
Время работы (е.в.): 2731.073 (Ожидаемое: 2666.667, погрешность: 2.415%)
Время простоя (е.в.): 1.607
ВОШЛО 1 типа: 1112
ВЫШЛО 1 типа: 1000
Количество обращений 2 типа: 337
```

Время простоя должно стремиться к нулю при более корректной обработке последней заявки, так как заявки 2-го типа подаются в очередь постоянно (по условию).

Алгоритм работы с очередью

Очередь — структура с типом доступа FIFO (первый зашел — первый вышел), следовательно:

- для удаления элемента из начала необходимо удалить голову очереди — O(1)
- для удаления элемента из конца очереди необходимо пройти всю очередь — O(n)
- для добавления элемента в конец очереди необходимо добавить к ее концу новый элемент
- для добавления элемента в начало или середину необходимо пройти часть очереди/всю очередь

Функция
`list_queue_t *new_list_queue(void)`
`void clear_list_queue(list_queue_t *list_queue)`
`void free_list_queue(list_queue_t *list_queue)`
`int list_queue_insert(list_queue_t *list_queue, int value, ssize_t index)`
`int list_queue_push(list_queue_t *list_queue, int value)`
`int list_queue_pop(list_queue_t *list_queue, int *value)`
`array_queue_t *new_array_queue(size_t init_size)`
`void free_array_queue(array_queue_t *array_queue)`
`int array_queue_expand(array_queue_t *array_queue)`
`int array_queue_insert(array_queue_t *array_queue, int value, int index)`
`int array_queue_push(array_queue_t *array_queue, int value)`
`int array_queue_pop(array_queue_t *array_queue, int *value)`

Таблица 3. Описание основных функций

Оценка эффективности

Эффективность оценивалась в трех плоскостях: заполнение и удаление очередей разного размера с использованием разных структур. Количество итераций: 100.

Тип структуры	Кол-во эл-тов	Размер (Б)	Время добавления элемента, тики	Время удаления элемента, тики
Вектор	100	440	36.63	1.17
	200	840	75.83	1.95
	500	2040	190.730	4.55
	1000	4040	381.49	8.94
Список	100	1616	44.23	3.18
	200	3216	83.45	5.34
	500	8016	212.44	13.71
	1000	16016	422.47	26.6

Данных в таблице можно сделать следующие выводы:

- Время, занимаемое операцией удаления или добавления элемента, зависит от размера структуры
- В среднем время удаления и добавления элемента из очереди на основе списка больше, так как при этом освобождается и выделяется память, а в очереди на основе массива (вектора) всего лишь меняются значения указателей

Контрольные вопросы

1 Что такое FIFO и LIFO?

LIFO и FIFO — принципы работы со структурами. При LIFO (last in — first out) элементы, добавленные последними, удаляются первыми. При FIFO — добавленные последними, удаляются также последними.

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

При реализации очереди в виде массива память выделяется под массив элементов, а при реализации в виде списка — под каждый элемент списка (значение элемента, указатель на следующий).

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

При реализации очереди в виде массива память при удалении элемента либо не освобождается, либо освобождается блоками/целиком, так как это массив. При реализации в виде списка при удалении память сразу же освобождается из-под самого элемента.

4 Что происходит с элементами очереди при ее просмотре?

При просмотре очереди ее элементы удаляются.

5 От чего зависит эффективность физической реализации очереди?

Эффективность физической реализации зависит от времени выполнения программы с этой реализацией и памяти, затраченной на реализацию.

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

Использование массива позволяет выиграть по скорости доступа к элементам, алгоритмам включения элементов. Также позволяет выиграть по памяти в рамках одного элемента очереди. В кольцевой структуре сложнее реализовывать алгоритмы включения/удаления элементов, легче при работе со статической памятью. Тем не менее, статический массив даёт ограничение по количеству элементов, вероятнее получить переполнение стека.

Достоинствами же использования списка являются возможность фрагментации памяти, скорость удаления элемента, ограниченность виртуальной памятью размерами очереди.

7 Что такое фрагментация памяти, и в какой части ОП она возникает?

Фрагментация памяти — процесс записи информации в использованных и освобожденных блоках памяти так, что она находится в разных местах доступной памяти. Возникает в динамической памяти.

8 Для чего нужен алгоритм «близнецов»?

Алгоритм «близнецов» используется для ускорения работы программы, уменьшения вероятности возникновения фрагментации памяти и улучшения поиск блоков памяти для выделения. В основе метода лежит создание буферов малого размера путем деления пополам больших буферов и слияния смежных буферов по мере возможности.

9 Какие дисциплины выделения памяти вы знаете?

«Самый подходящий» – выделяется участок памяти, равный требуемому или превышающий его на минимальную величину.

«Первый подходящий» – выделяется участок памяти, найденный первым, больший или равный требуемому.

10 На что необходимо обратить внимание при тестировании программы?

Если реализована очередь-список, то необходимо следить за высвобождением памяти из-под элементов списка.

Если реализована очередь-массив, необходимо проверять корректность операций на кольце, чтобы не произошло записи в невыделенную память.

11 Каким образом физически выделяется и освобождается память при динамических запросах?

При динамических запросах выделения памяти выделенная память располагается в куче: указывается размер запрашиваемой под объект памяти, и, в случае успеха, выделенная область памяти становится недоступной при последующих операциях выделения памяти.

При динамических запросах освобождения памяти память возвращается в кучу и становится доступной при дальнейших операциях выделения памяти.

Выводы

В ходе работы были изучены детали обработки очередей с различными реализациями и основы моделирования различных кассовых аппаратов.