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»

Вариант 6

Студент: Писаренко Дмитрий Павлович

Группа: ИУ7-34Б

Студент _____ Писаренко Д.П.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Рыбкин Ю.А.

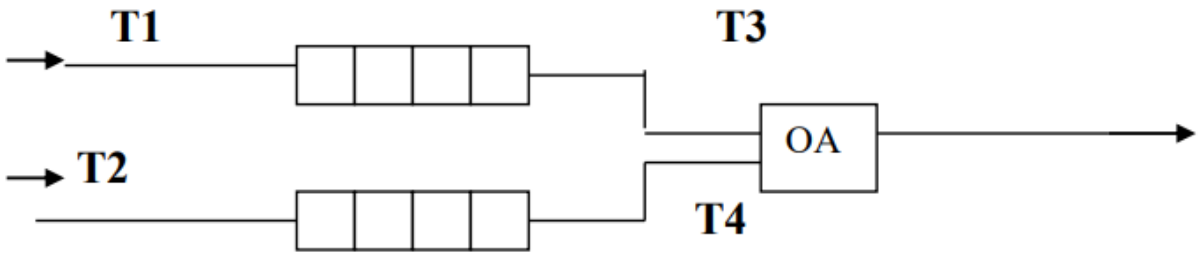
подпись, дата *фамилия, и.о.*

Цель работы

Цель работы: приобрести навыки работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка, провести сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании указанных структур данных, оценить эффективности программы по времени и по используемому объему памяти.

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 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. В конце процесса выдать общее время моделирования и количестве 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 ОА добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Техническое задание

Исходные данные

Выбор действия: целое число от 0 до 10.

0. Выход из программы

Не требует ввода от пользователя.

1. *Добавить элемент (массив)*

Включается случайный элемент от 1 до 100 в хвост очереди, реализованной в виде одномерного массива.

2. Удалить элемент (массив)

Исключается элемент из головы очереди, реализованной в виде одномерного массива.

3. Распечатать очередь (массив)

На экран выводится текущее состояние очереди, реализованной в виде одномерного массива.

4. *Добавить элемент (список)*

Включается случайный элемент от 1 до 100 в хвост очереди, реализованной в виде списка.

5. Удалить элемент (список)

Исключается элемент из головы очереди, реализованной в виде списка, в массив освободившихся адресов добавляется адрес удаленного элемента.

6. Распечатать очередь (список)

На экран выводится текущее состояние очереди, реализованной в виде списка.

7. Распечатать массив освободившихся адресов

На экран выводятся адреса удаленных ранее элементов.

8. *Выполнить задание для массива*

Моделируется процесс, указанный в задании для очереди, реализованной в виде массива.

9. *Выполнить задание для списка*

Моделируется процесс, указанный в задании для очереди, реализованной в виде списка.

10. Замеры времени и памяти

От пользователя требуется ввести кол-во элементов, на котором он желает произвести замеры: от 1 до 1000. Замеры производятся для добавления и удаления и для массива, и для списка.

```c
typedef struct node_s
{
    int elem;
    struct node_s *next;
} node_t;
```

```
int elem – значение текущего элемента очереди
struct node_s *next – указательный на следующий элемент очереди
```

Описание полей структуры node_t

```c
typedef struct list_s
{
    node_t *head;
    node_t *tail;
    int size;
} list_t;
```

```
node_t *head – указатель на голову очереди
node_t *tail – указатель на хвост очереди
int size – размер очереди
```

Описание полей структуры list_t

```c
typedef struct arr_s
{
    int arr[MAX_ELEMS];
    int begin;
    int end;
    int size;
} arr_t;
```

```
int arr[MAX_ELEMS] – массив элементов очереди
int begin – голова очереди
int end – хвост очереди
int size – размер очереди
```

Описание полей структуры arr_t

```c
typedef struct
{
    node_t *arr_clear[MAX_ELEMS];
    int len;
} arr_clear_t;
```

```
node_t *arr_clear[MAX_ELEMS] – массив адресов удаленных элементов
int len – длина массива адресов удаленных элементов
```

Описание полей структуры arr_clear_t

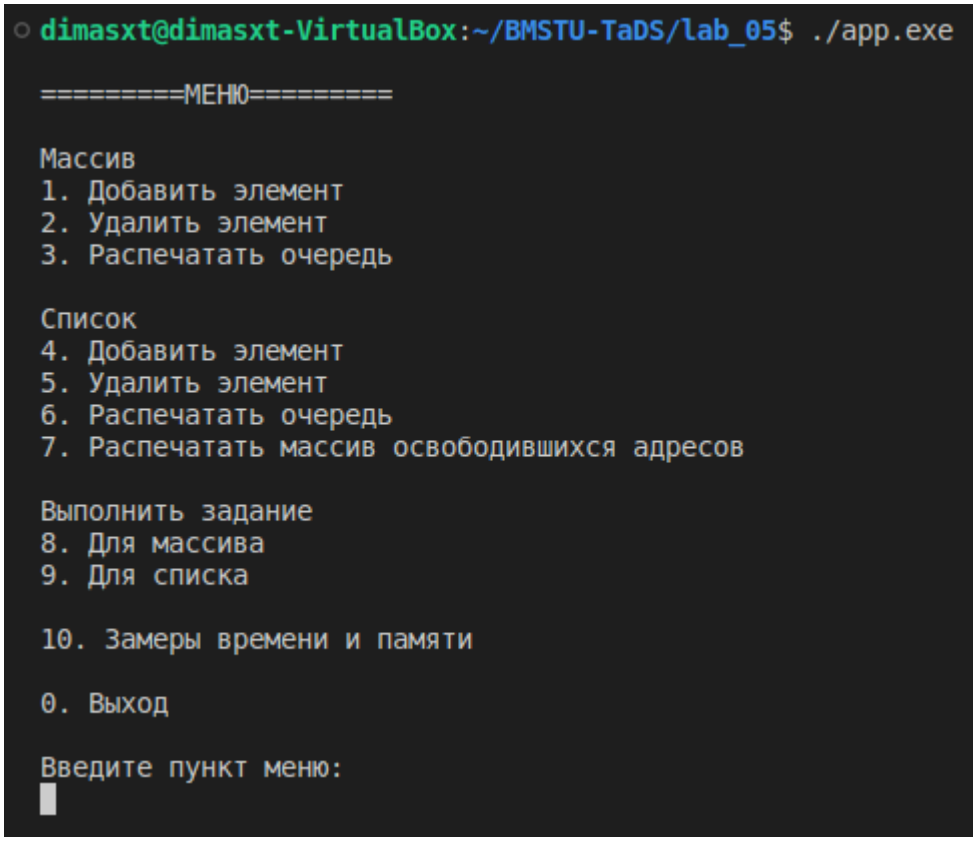
Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Сборка осуществляется с помощью команды **make release**

Запуск выполняется с помощью команды **./app.exe**

Дальнейшая работа производится с помощью меню:



Тестирование

Позитивные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 1	Отсутствуют	Ожидание следующего ключа
2	Ключ = 2	Отсутствуют	Ожидание следующего ключа
3	Ключ = 1 Ключ = 1 Ключ = 1 Ключ = 3	3 случайных элемента от 1 до 100, например: 89 45 5	Ожидание следующего ключа
4	Ключ = 4	Отсутствуют	Ожидание следующего ключа
5	Ключ = 5	Отсутствуют	Ожидание следующего ключа
6	Ключ = 4 Ключ = 4 Ключ = 4 Ключ = 6	3 случайных элемента от 1 до 100 и их адреса, например: 43 | 0x5574de29bb00 28 | 0x5574de29b2c0 58 | 0x5574de29bb20	Ожидание следующего ключа
7	Ключ = 4 Ключ = 5 Ключ = 7	Массив освободившихся адресов: 0x55ae9d25d2c0	Ожидание следующего ключа
8	Ключ = 8 или Ключ = 9	Каждые 100 заявок выводится информация о текущей и средней длине каждой очереди В конце процесса выводится общее время моделирования и количество вошедших в	Ожидание следующего ключа

		систему и вышедших из нее заявок обоих типов, среднее время пребывания заявок в очереди, количество «выброшенных» заявок второго типа	
9	Ключ = 0	Отсутствуют	Завершение программы

Негативные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = fdwsfd	Номер меню - число от 0 до 10	Код ошибки 1
2	Ключ = 10 Кол-во элементов: 10000	Ошибка: неверно введено количество элементов	Ожидание следующего ключа
3	*В очереди 1000 элементов* Ключ = 1	Ошибка: очередь переполнена	Код ошибки 2

Таблицы с результатами измерения времени и памяти

Время измерялось в тактах процессора, в таблицах указано в мкс (10^{-6} с).

Память в таблицах указана в байтах.

Добавление

Размер	Массив		Список	
	Время	Память	Время	Память
10	0.046	4012	0.089	160
100	0.596	4012	1.426	1600
500	3.906	4012	16.967	8000
1.000	9.026	4012	24.846	16000

Удаление

Размер	Массив		Список	
	Время	Память	Время	Память
10	0.040	4012	0.132	160
100	0.467	4012	1.147	1600
500	2.243	4012	5.288	8000
1.000	4.425	4012	12.828	16000

По приведенным выше таблицам можно сделать следующие выводы:

- По времени выполнения: во всех тестах очередь-массив выигрывает в скорости у очередь-списка в 2-3 раза.

- По памяти: хранение в виде очереди-списка выгоднее при количестве заполненных элементов до 25% от общего.

Вычисления

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

Найдем время моделирования:

MAX(ср. время прихода заявки 1ого типа; ср. время обработки заявки 1ого типа) * 1000 = MAX(3;2) * 1000 = 3000 ед. вр.

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

Посчитаем число вошедших заявок второго типа:

Ср. время прихода заявки 1ого типа = 3 ед. вр., а ср. время прихода заявки 2ого типа = 1.5 ед. вр., отсюда 1000/0.5 = 2000 заявок

Практические вычисления

```
Общее время моделировния = 3038.310547
Погрешность моделирования = 1.277%

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 = 2031
Заявок 2ой очереди вышло = 1661
Среднее время обработки заявки в 2ой очереди (ожидаемое) = 1.500000
Заявок 2го типа вернулось обратно в конец очереди = 1217

Погрешность ввода заявок в 1ую очередь 1.261%
Погрешность ввода заявок во 2ую очередь 0.270%

Время простоя = 8.775879
```

Контрольные вопросы

1. Что такое FIFO и LIFO?

Это принципы работы, FIFO – первый зашел первый вышел, LIFO – последний зашел, первый вышел. FIFO используется для реализации очереди, а LIFO для реализации стека.

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

При реализации очереди на статическом массиве память выделяется один раз 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.

3. 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.

4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди элементы удаляются из нее. В качестве наглядности в данной лабораторной работе это не было реализовано.

5. От чего зависит эффективность физической реализации очереди?

От объема оперативной памяти.

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

При реализации очереди на статическом массиве её размер 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

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

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

8. Для чего нужен алгоритм «близнецов».

Главная идея, лежащая в основе методов близнецов, заключается в том, что все блоки имеют лишь строго определённые размеры. Поэтому его используют тогда, когда мы знаем какой размер элементов нам нужен и что он не будет изменятся. Что сокращает использование памяти.

9. Какие дисциплины выделения памяти вы знаете?

Статическая – выделяется на уровне трансляции, динамическая – выделяется во время выполнения программы.

10. На что необходимо обратить внимание при тестировании программы?

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

11. 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

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. Точные цифры приведены в пункте “Таблицы с результатами измерения времени и памяти”.