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»

Студент Шелия София Малхазовна

Группа ИУ7 – 35Б

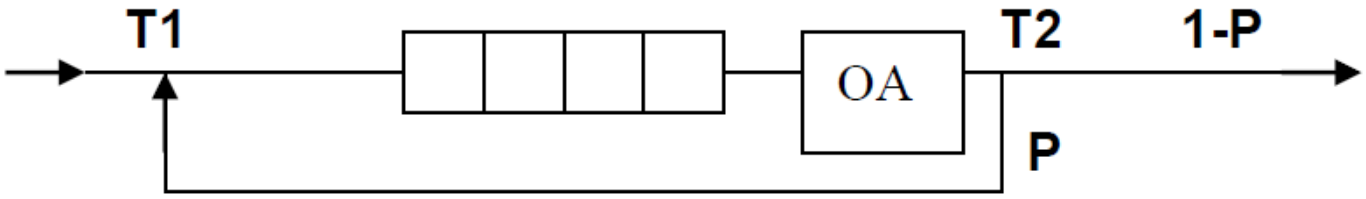
2020 г.

Цель работы: отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценка эффективности программы (при различной реализации) по времени и по используемому объему памяти.

1. Описание условия задачи.

Вариант №5

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



Заявки поступают в "хвост" очереди по случайному закону с интервалом времени **T1,** равномерно распределенным от **0 до 6** единиц времени (е.в.). ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время **T2** от **0 до 1** е.в., Каждая заявка после ОА с вероятностью **P=0.8** вновь поступает в "хвост" очереди, совершая новый цикл обслуживания, а с вероятностью **1-P** покидает систему. (Все времена – **вещественного** типа). В начале процесса в системе заявок нет.
Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок. Выдавать после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди. В конце процесса выдать общее время моделирования и количество вошедших в систему и вышедших из нее заявок, среднее время пребывания заявки в очереди, время простоя аппарата, количество срабатываний ОА. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

2. Техническое задание

Исходные данные и результат

Ввод:

1. Пользователь вводит цифру (от 0 до 6), в зависимости от выбранного пункта меня.
   - 1 - Добавление элемента в очередь.

   Пользователь вводит целое число – элемент, который он хочет добавить. Максимальное количество элементов в очереди задается заранее.
   - 2 – Удаление элемента из очереди
   - 3 – Печать текущего состояния очереди
   - 4 – Выполнение задания для связного списка
   - 5 – Выполнение задания для кольцевого массива

- 6 – Сравнение времени обработки для разной реализации очереди
- 0 – Завершение программы

Вывод: (для пунктов 1 - 6)

1. Программа добавляет элемент в очередь, представленную в виде массива и связного списка, а потом выводит их, а также выводит адреса элементов связного списка и свободной области.
2. Программа удаляет элемент из очереди, представленной в виде массива и связного списка, а потом выводит их, а также выводит адреса элементов связного списка и свободной области.
3. Программа выводит текущее состояние связного списка и статического массива, а также адреса элементов связного списка и свободной области.
4. Вывод по заданию
5. Вывод по заданию
6. Программа выводит сравнительную таблицу времени обработки очереди при различном количестве элементов

Описание задачи, реализуемой программой

1. Программа создает две очереди, реализованные в виде статического массива и связного списка.
2. Программа производит операции добавления/удаления элемента одновременно для обеих реализаций очереди.
3. Программа выводит текущее состояние очереди в обеих реализациях, а также адреса элементов связного списка и свободной области.
4. Программа выполняет задание.
5. Программа выводит сравнительную таблицу времени обработки очереди при различном количестве элементов

Способ обращения к программе

Запуск приложения возможен через терминал MSYS2, а именно.

1. gcc -std=c99 -Wall -Werror -c *.c
2. gcc -o main.exe *.o
3. ./main.exe

Описание возможных аварийных ситуаций и ошибок пользователя

Ошибки пользователя при выборе пункта меню

- Пустой ввод
- Вводятся посторонние символы (например, буквы)
- Вводится отрицательное число
- Вводится не целое число
- Вводится отсутствующий пункт меню

Ошибки пользователя при вводе элемента для добавления

- Пустой ввод
- Вводятся посторонние символы (например, буквы)
- Вводится не целое число

Аварийные ситуации.

- Попытка добавления элемента в переполненную очередь
- Попытка удаление элемента из пустой очереди

3. Описание внутренних структур данных

Для хранения очереди в виде статического массива я использовала следующую структуру:

```
//Структура очереди для статического массива
#define MAX_ELEMENTS_NUMBER 1001
typedef struct queue_sa{
    int queue[MAX_ELEMENTS_NUMBER]; - Элементы очереди
    int begin; - Начало очереди
    int end; - Конец очереди
}queue_sa;
```

Для хранения очереди в виде связного списка я использовала следующую структуру:

```
//Структура очереди
typedef struct queue_ll{
    int data;
    struct queue_ll *next;
}queue_ll;

typedef struct List
{
    int size;
    queue_ll *head; - начало очереди
    queue_ll *tail; - конец очереди
}List;
```

4. Алгоритм

Сначала программа генерирует два времени time_in и time_ser (время входа новой заявки и обработки старой), затем выполняет действие, которое меньше по времени (от действия с большим временем далее отнимается значения действия с меньшим временем, а также к общему счетчику времени прибавляется время

более быстрого действия). Далее время для более быстрого события обновляется, и происходит новое сравнение.
Если какая-то заявка выходит из очереди, то генерируется вероятность того, что она вернется обратно.
Если в момент, когда должна происходить обработка - очередь пустая, то увеличивается время простоя.
Моделирование заканчивается в момент, когда ОА НАВСЕГДА покинут 1000 заявок

5. Тесты

Негативные тесты

Входные данные	**Результат**	**Условие проверки**
Для выбора пункта меню		
Выберите действие: 1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор:	Некорректный ввод.	Пустой ввод
Выберите действие: 1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор: цыкпцы	Некорректный ввод.	Вводятся посторонние символы (например, буквы)
Выберите действие: 1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для	Некорректный ввод.	Вводится отрицательное число

связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор: -5		
Выберите действие: 1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор:14	Некорректный ввод.	Вводится отсутствующий пункт меню
Выберите действие: 1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор:45.545	Некорректный ввод.	Вводится не целое число
Для ввода значения элемента		
Введите целое число - значение элемента для добавления:	Некорректный ввод. Введите целое число:	Пустой ввод
Введите целое число - значение элемента для добавления: 1ава	Некорректный ввод. Введите целое число:	Вводятся посторонние символы (например, буквы)
Введите целое число - значение элемента для добавления: 34.12	Некорректный ввод. Введите целое число:	Вводится не целое число

	Очередь пустая, удаление невозможно	Попытка удаления элемента из пустой очереди
	Очередь переполнена, добавление невозможно (максимальное	Попытка добавления элемента в переполненную очередь

	количество элементов - %d)	

Позитивные тесты

Входные данные	Результат	Условие проверки
1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор: 1 Введите целое число - значение элемента для добавления: 1	**До:** Элементы очереди (массив): 1 -> NULL Элементы очереди (связной список): 1 -> NULL Адреса элементов связного списка: 0000000000095EB0 0000000000095ED0 Адреса элементов свободной области: **После:** Элементы очереди (массив): 1 -> 2 -> NULL Элементы очереди (связной список): 1 -> 2 -> NULL Адреса элементов связного списка: 0000000000095EB0 0000000000095ED0 Адреса элементов свободной области:	Добавление элемента в очередь
1 - Добавить элемент в очередь (максимальное количество элементов 1001) 2 - Удалить элемент из очереди 3 - Распечатать текущее состояние очереди 4 - Выполнить задание для связного списка 5 - Выполнить задание для кольцевого массива 6 - Сравнить время обработки очереди, реализованной в виде массива и связного списка 0 - Завершить программу Выбор: 2	**До:** Элементы очереди (массив): 1 -> 2 -> NULL Элементы очереди (связной список): 1 -> 2 -> NULL Адреса элементов связного списка: 0000000000095EB0 0000000000095ED0 Адреса элементов свободной области: **После:** Элементы очереди (массив): 2 -> NULL Элементы очереди (связной список): 2 -> NULL Адреса элементов связного списка: 0000000000095ED0 0000000000095EB0 Адреса элементов свободной области: 0000000000095EB0	Удаление элемента из очереди

6. Сравнение двух реализаций стека

Количество итераций 1000000
Время

Добавление элемента

Размер очереди	Связный список	Статический массив
10	0.0000000320 сек.	0.0000000090 сек.
20	0.0000000370 сек.	0.0000000110 сек.
50	0.0000000410 сек.	0.0000000120 сек.
100	0.0000000390 сек.	0.0000000130 сек.
500	0.0000000390 сек.	0.0000000120 сек.
1000	0.0000000530 сек.	0.0000000090 сек.

Удаление элемента

Размер очереди	Связный список	Статический массив
10	0.0000000230 сек.	0.0000000080 сек.
20	0.0000000220 сек.	0.0000000030 сек.
50	0.0000000260 сек.	0.0000000080 сек.
100	0.0000000280 сек.	0.0000000040 сек.
500	0.0000000240 сек.	0.0000000080 сек.
1000	0.0000000170 сек.	0.0000000040 сек.

Занимаемая память

Размер очереди	Статический массив	Связный список
10	4016	184
20	4016	344
50	4016	824
100	4016	1624
500	4016	8024
1000	4016	16024

Выводы по проделанной работе

Очередь, реализованная связным списком, проигрывает по времени обработки статическому массиву, но в большинстве случаев выигрывает по памяти. Однако при приближении количества элементов к максимальному количеству элементов в очереди, связной список начинает проигрывать и по памяти.

Ответы на контрольные вопросы

1. Что такое очередь?

Очередь – структура данных типа «список», позволяющая добавлять элементы лишь в конец списка (хвост), и извлекать их из его начала (головы). Принцип работы очереди: первым пришел – первым вышел, т.е. First In – First Out (FIFO).

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

Для кольцевого статического массива: память для всей очереди выделяется на стеке, в зависимости от указанного количества элементов. (Если массив динамический память выделяется на куче постепенно). Объём выделяемой памяти равен n * sizeof (элемента очереди), где n – количество элементов в очереди.

Для связного списка: Память выделяется постепенно на куче при добавлении нового элемента. Объём выделяемой памяти равен n * (sizeof(элемент очереди) + sizeof(указатель на следующий элемент)), где n – количество элементов в очереди

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

Для кольцевого статического массива: память не освобождается, просто изменяется указатель на начало очереди.

Для связного списка: для удаляемого элемента происходит освобождение памяти, и смещается указатель на начало очереди.

4. Что происходит с элементами очереди при ее просмотре?

Элементы очереди уничтожаются, так как происходит постоянное ее “смещение”.

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

По времени обработки самой выигрышной является реализация очереди в виде кольцевого массива. И если известно максимальное количество элементов в очереди, то несомненно стоит использовать именно эту реализацию, в ином случае лучше использовать реализацию в виде связного списка, так как тогда переполнение очереди возможно только при условии исчерпания оперативной памяти.

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

Ответ в пункте 5.

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

При использовании связного списка для реализации очереди может возникнуть фрагментация памяти, а также такой способ реализации является проигрышным по времени, но зато размер очереди ограничен лишь объёмом оперативной памяти. При реализации очереди статическим кольцевым массивом наша очередь всегда будет ограниченна определенным размером, но операции при такой реализации выполняются гораздо быстрее.

8. Что такое фрагментация памяти?

Фрагментация памяти - процесс появления незанятых участков в памяти, которые чередуются с занятыми, но непригодны для какого-либо использования (фрагментация может приводить к ошибкам выделения памяти)

9. На что необходимо обратить внимание при тестировании программы?

- Корректное освобождение памяти
- Работа с переполненной очередью
- Работа с пустой очередью

10. Каким образом физически выделяется и освобождается память при динамических запросах?

ОС выбирает какой-то блок памяти, куда будет происходить запись данных (причем иногда в выбранном блоке может быть недостаточно памяти для наших нужд, что приводит к возникновению ошибок) При освобождении памяти ОС удаляет данные из данного блока памяти, тем самым делая его снова свободным.