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»

Студентка Гурова Наталия Алексеевна

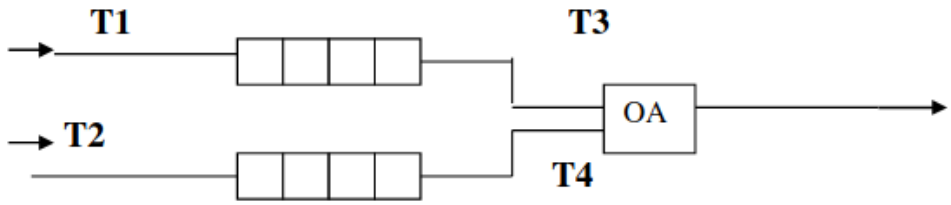
Группа ИУ7 – 34Б

Принял Барышникова Марина Юрьевна

2021 г.

Описание задания

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 Т1и Т2, равномерно распределенными от 1 до 5 и от 0 до 3 единиц времени соответственно. В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за времена Т3 и Т4, распределенные от 0 до 4 и от 0 до 1 соответственно, после чего покидают систему. 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 повторнымобслуживанием).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок 1-го типаинформацию 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:

Целое число от 0 до 4 - номер команды.

Выходные данные:

Результат выполнения определенной команды.

Функции программы:

1. Моделирование очереди в виде массива.
2. Моделирование очереди в виде односвязного линейного списка.
3. Изменение времени обработки заявки.
4. Отчет о затрачиваемой памяти и времени.

Обращение к программе:

Запускается через терминал с помощью команды ./app.exe

Аварийные ситуации:

1. Некорректный ввод номера команды.

   На входе: не число в диапазоне от 0 до 4.

   На выходе: Invalid input command

2. Некорректный ввод номера интервала времени

   На входе: не число в диапазоне от 0 до 4.

   На выходе: Invalid input command

3. Некорректный ввод выбора времени.

   На входе: левая граница не вещественное положительное число

   На выходе: Invalid input left border

4. Некорректный ввод выбора времени.

   На входе: правая граница не вещественное положительное число

   На выходе: Invalid input right border

Описание структуры данных

Структура для описания границ времени обработки

```
typedef struct
{
    double min;
    double max;
} times_t;
```

Поля структуры:
- *min* - нижняя граница времени
- *max* - верхняя граница времени

Структура для описания узла односвязного линейного списка.

```
typedef struct node
{
    struct node *next;
    char data;
} Node_t;
```

Поля структуры:
- *data* - данные в узле
- *next* - указатель на следующий узел

Структура для описания очереди.

```
typedef struct queue
{
    char name[MAX_LEN_NAME];
    void *address_low_border;
    void *address_up_border;
    void *tail;
    void *head;

    int count_elems;
    int size_of_elems;

    int count_requests;
    int middle_len;
    int len_now;
    int sum_time;
    int requests_to_input;
    int requests_to_output;
} queue_t;
```

Поля структуры:
- *name* - название очереди
- *address_low_border* - адрес нижней границы
- *address_up_border* - адрес верхней границы
- *tail* - указатель на "хвост" очереди
- *head* - указатель на "голову" очереди
- *count_elems* - число элементов в очереди
- *size_of_elems* - размер типа данных в очереди
- *count_requests* - число запросов в очереди
- *middle_len* - средняя длина очереди
- *len_now* - текущая длина очереди
- *sum_time* - общее время работы с очередью
- *requests_to_input* - число запросов на вход
- *requests_to_output* - Число запросов на выход

Оценка эффективности

Работа ОА

Массив

	Число заявок 1-го типа	Число заявок 2-го типа	Время моделирования (ус.е.в.)	Время работы (реальное время в мкс.)
1	1000	2024	3013.429243	42
2	1000	2026	3052.810877	41
3	1000	1971	2958.759087	40
4	1000	1999	3012.510636	43
5	1000	1975	2979.292093	41
6	1000	2019	2996.318552	42
7	1000	1993	2967.007416	42
8	1000	1983	2950.963042	42
9	1000	1993	3028.803186	43
10	1001	1998	3015.691885	42
Среднее	1000	1998.1	2997	41.8

Список

	Число заявок 1-го типа	*Число заявок 2-го типа*	*Время моделирования (ус.е.в.)*	*Время работы (реальное время в мкс.)*
1	1001	2031	3029.341624	48
2	1000	1966	3006.491653	42
3	1001	1923	2957.131046	46
4	1000	2012	3021.905576	43
5	1000	2051	3063.977783	42
6	1000	2016	3061.863002	43
7	1000	1934	2952.349742	44
8	1000	2025	2998.866451	45

9	1000	1983	2995.102573	45
10	1000	2048	3041.926969	44
Среднее	1000.2	1998.9	3012.3	44.2

Очередь как массив или как список

RESULT TIME

	ADD	DELETE
ARRAY	210	147
LIST	3612	210

RESULT MEMORY

COUNT ELEMS	ARRAY	LIST
10	10	160
100	100	1600
1000	1000	16000
10000	10000	160000

Расчет времени работы очереди

Теоретический расчет времени *моделирования* = max(среднее время прихода заявки 1 типа, среднее время обработки заявки 1 типа) * (количество).

Стандартные временые границы :

Время поступления заявки первого типа : 1..5

Время поступления заявки второго типа: 0..3

Время обработки заявки первого типа : 0..4

Время обработки заявки второго типа : 0..1

Теоретические результаты :

Время моделирования : 3000 е.в.

Время обработки заявок 1 типа : 2 * 1000 = 2000

Время, когда ОА не работает: (время моделирования – время обработки заявок 1 типа) = 1000

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

Число заявок 2 типа, вошедших : (время моделирования / среднее время прихода 2 заявки) = 2000

Число заявок 2 типа, вышедших : 2000 (время, когда ОА не работает / среднее время прихода 2 заявки) (вышедшие + оставшиеся в очереди, в результатах)

Практические результаты :

RESULT

Total time	Real time	Time without work	Error rate
3006.463332	40	5.312967	0.215444%

	Middle time	Count tasks input	Count tasks output	Count tasks throw	Error rate input
Queue 1	3.000000	1000	1000	-	0.214981%
Queue 2	1.500000	1988	1634	354	0.813691%

Временые границы, при которых время обработки больше, чем время прихода:

Время поступления заявки первого типа : 1..5

Время поступления заявки второго типа : 0..3

Время обработки заявки первого типа : 0..10

Время обработки заявки второго типа : 0..1

Теоретические результаты :

Время моделирование равно 5000 е.в.

Время обработки заявок 1 типа : 5 * 1000 = 5000

Время, когда ОА не работает : (время моделирования – время обработки заявок 1 типа) = 0

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

Число заявок 2 типа, вошедших : 3300 (время моделирования / среднее время прихода 2 заявки)

Число заявок 2 типа, вышедших : 0 (время, когда ОА не работает / среднее время прихода 2 заявки) (вышедшие + оставшиеся в очереди, в результатах)

Практические результаты :

```
RESULT
-------------------------------------------------------------
|  Total time  |  Real time  |  Time without work  |  Error rate  |
-------------------------------------------------------------
|  5092.935392 |         40  |           1.222938  |   1.858708%  |
-------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------
|         |  Middle time  |  Count tasks input  |  Count tasks output  | Count tasks throw  |  Error rate input   |
-------------------------------------------------------------------------------------------------------------
| Queue 1 |     3.000000  |              1708  |               1000  |         -          |          0.609955%  |
-------------------------------------------------------------------------------------------------------------
| Queue 2 |     1.500000  |              3343  |                  1  |             3342  |          1.540082%  |
-------------------------------------------------------------------------------------------------------------
```

Временые границы, при которых время прихода и обработки заявки 2 типа одинаковое

Время поступления заявки первого типа : 1..5

Время поступления заявки второго типа : 0..1

Время обработки заявки первого типа : 0..4

Время обработки заявки второго типа : 0..1

Теоретические результаты :

Время моделирование равно 3000 е.в.

Время обработки заявок 1 типа : (среднее время обработки 1 типа) * (количество) = 2 * 1000 = 2000

Время, когда ОА не работает (в отношении 1 очереди): (время моделирования – время обработки заявок 1 типа) = 1000

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

Число заявок 2 типа, вошедших : 6000 (время моделирования / среднее время прихода 2 заявки)

Число заявок 2 типа, вышедших : 1500 (время, когда ОА не работает (в отношении 1 очереди) / среднее время прихода 2 заявки) (вышедшие + оставшиеся в очереди, в результатах)

Практические результаты :

```
RESULT
------------------------------------------------------------------
|  Total time  |  Real time  |  Time without work  |  Error rate  |
------------------------------------------------------------------
|  2977.195257 |         42  |           1.592395  |   0.760158%  |
------------------------------------------------------------------


-------------------------------------------------------------------------------------------------------------------
|         |  Middle time  |  Count tasks input  |  Count tasks output  | Count tasks throw  |  Error rate input   |
-------------------------------------------------------------------------------------------------------------------
| Queue 1 |     3.000000  |              1000   |               1000   |         -          |          0.765981%  |
-------------------------------------------------------------------------------------------------------------------
| Queue 2 |     0.500000  |              6006   |               1331   |             4675   |          0.866747%  |
-------------------------------------------------------------------------------------------------------------------
```

Ответы на контрольные вопросы

1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с «хвоста», а исключение – из «головы». Принцип работы очереди: первым пришел – первым вышел (FIFO).

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

При реализации массивом (кольцевым), кол-во элементов * размер одного элемента. Если массив статический, то память выделяется в стеке, если массив динамический, то в куче.

При реализации списком, под каждый новый элемент выделяется память размером (размер элемента + размер указателя в куче), для каждого элемента отдельно.

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

При удалении элемента из очереди в виде массива, перемещается указатель, память не освобождается. Память освобождается в конце программы. Если массив статический, то после завершения программы, если динамический — с помощью функции free().

При удалении элемента из очереди в виде списка, освобождается память из данного элемента сразу. (Указатель на «голову» переходит на следующий элемент, считанный элемент удаляется, память освобождается)

4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди, головной элемент («голова») удаляется, и указатель смещается. То есть при просмотре очереди ее элементы удаляются.

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

При реализации очереди в виде массива (кольцевого статического), может возникнуть переполнение памяти, фрагментации не возникает. Быстрее работают операции добавления и удаления элементов. Также необходимо знать тип данных.

При реализации в виде списка — легче удалять и добавлять элементы, переполнение памяти может возникнуть только если закончится оперативная память, однако может возникнуть фрагментация памяти.

Если изначально знать размер очереди и тип данных, то лучше воспользоваться массивом. Не зная размер — списком.

Также способ реализации зависит от того, в чем мы больше ограниченны, в памяти или во времени.

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

Если важна скорость выполнения, то лучше использовать массив, так как все операции с массивом выполняются быстрее, но очередь ограничена по памяти (так как массив статический).

Но если неизвестно сколько будет элементов в очереди — то лучше использовать список, так как он ограничен только оперативной памятью, но может возникнуть фрагментация памяти.

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

При реализации очереди в виде массива не возникает фрагментация памяти, так же может возникнуть переполнение очереди, и тратиться дополнительное время на сдвиги элементов (классический массив). Сдвигов нет, если использовать кольцевой статический массив, но усложняется реализация алгоритмов добавления и удаления элементов.

При реализации очереди в виде списка, проще выполнять операции добавления и удаления элементов, но может возникнуть фрагментация памяти.

8. Что такое фрагментация памяти?

Фрагментация – чередование участков памяти при последовательных запросах на выделение и освобождение памяти. «Занятые» участки чередуются со «свободными» - однако последние могут быть недостаточно большими для того, чтобы сохранить в них нужное данное.

9. На что необходимо обратить внимание при тестировании программы?

При реализации очереди в виде списка необходимо следить за освобождением памяти при удалении элемента из очереди. Если новые элементы приходят быстрее, чем уходят старые, то может возникнуть фрагментация памяти.

При реализации очереди в виде массива (кольцевого) надо обратить внимания на корректную работу с ним, чтобы не произошло записи в невыделенную память.

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

Программа дает запрос ОС на выделение блока памяти необходимого размера. ОС находит подходящий блок, записывает его адрес и размер в таблицу адресов, а затем возвращает данный адрес в программу.

При запросе на освобождение указного блока программы, ОС убирает его из таблицы адресов, однако указатель на этот блок может остаться в программе. Обращение к этому адресу и попытка считать данные из этого блока может привести к неопределенному поведению, так как данные могут быть уже изменены.

Вывод

Преимущества очереди в виде массива:

1. Используется меньше памяти (в 16 раз меньше, чем в случае списка)
2. Операции добавления и удаления выполняются быстрее (в 1,5 и более раза)

Недостатки очереди в виде массива:

1. Ограничение размера очереди (возможно переполнение)
2. Операции добавления и удаления реализовать сложнее, чем в случае списка

Таким образом, обе реализации очереди имеют свои преимущества и недостатки. Так как нельзя однозначно сказать, какой способ лучше, в случае реальной задачи нужно выбирать тот метод, который наиболее отвечает требованиям. Если важна скорость обработки, а примерное количество элементов заранее известно, то лучше выбрать реализацию с помощью массива. В случае же, когда не известна верхняя граница количества элементов, лучше воспользоваться реализацией со списком.