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 Козлова Ирина Васильевна

Группа ИУ7 – 32Б

2020 г.

Оглавление

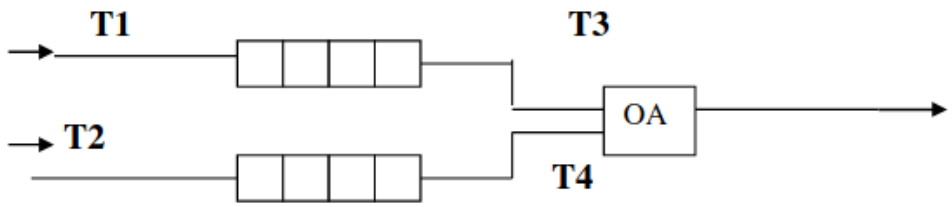
Описание условия задачи

Система массового обслуживания состоит из обслуживающих аппаратов (ОА) и очередей заявок двух типов, различающихся временем прихода и обработки. Заявки поступают в очереди по случайному закону с различными интервалами времени (в зависимости от варианта задания), равномерно распределенными от начального значения (иногда от нуля) до максимального количества единиц времени.В ОА заявки поступают из «головы» очереди по одной и обслуживаются за указанные в задании времена, распределенные равновероятно от минимального до максимального значений (все времена – вещественного типа).

Требуется смоделировать процесс обслуживания первых 1000 заявок первого типа, выдавая после обслуживания каждых 100 заявок первого типа информацию о текущей и средней длине каждой очереди и о среднем времени пребывания заявок каждого типа в очереди. В конце процесса необходимо выдать на экран общее времямоделирования, время простоя ОА, количество вошедших в систему и вышедших из нее заявок первого и второго типов.
Очередь необходимо представить в виде вектора и списка. Все операции должны быть оформлены подпрограммами. Алгоритм для реализации задачи один, независимо от формы представления очереди. Необходимо сравнить эффективность различного представления очереди по времени выполнения программы и по требуемой памяти. При реализации очереди списком нужно проследить, каким образом происходит выделение и освобождение участков памяти, для чего по запросу пользователя необходимо выдать на экран адреса памяти, содержащие элементы очереди при добавлении или удалении очередного элемента.

Описание технического 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 T1и T2, равномерно распределенными от 1 до 5и от 0 до 3единиц времени (е.в.) соответственно. В ОА они поступают из "головы" очередипо одной и обслуживаются также равновероятно за времена T3и T4, распределенные от 0 до 4е.в. и от 0 до 1е.в. соответственно, после чего покидают систему. (Все времена – вещественноготипа) 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 абсолютнымприоритетом 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 количествевошедших 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:

1. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 4.
2. **Командно-зависимые данные:**
   целочисленные значения (количество элемента стека)

Выходные данные:

1. Результат выполнения определенной команды.
2. Моделирование и характеристика для очереди в виде списка/массива.

Функции программы:

1. Моделирование и характеристика для очереди в виде массива.
2. Моделирование и характеристика для очереди в виде массива.
3. Изменение времени обработки заявки.

0. Выход из программы.

Обращение к программе:

Запускается через терминал. Так же можно собрать программу используя makefile, и запустить ее с помощью команды run.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 13 или меньшее, чем 0.
   На выходе: сообщение «ERROR!!! Invalid command entered, please re-enter!!!»
2. Некорректный ввод номера команды.
   На входе: пустой ввод.
   На выходе: сообщение «Invalid command entered, please re-enter!!!»
3. Некорректный ввод выбора времени.
   На входе: число, большее чем 4 или меньшее, чем 1.
   На выходе: сообщение «ERROR!!! Invalid command entered, please re-enter!!!»
4. Некорректный ввод времени обработки.
   На входе: буква.
   На выходе: сообщение «ERROR!!! Invalid number input! Please choose some command!»

Описание структуры данных

Структура для описании границ времени обработки

```
typedef struct times
{
     double min;
     double max;
} times_r;
```

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

Структура для описания узла списка.

```
typedef struct node
{
     char inf;
     struct node *next;
} node_r;
```

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

Структура для описания всей информации про очередь.

```
typedef struct queue
{
     char name[30];
     void* low;
     void* up;
     void* p_in;
     void* p_out;
     int count_len;
     size_t size;
     int count_req;
     int sum_len;
     int tmp_len;
     int sum_time;
     int out_req;
     int in_req;
} queue_r;
```

Поля структуры:
char name[30] - Имя очереди
void low* - Адрес нижней границы очереди
void up* - Адрес верхней границы очереди
void p_in* - Указатель на "хвост" очереди
void p_out* - Указатель на "голову" очереди
int count_len - Число элементов в очереди
size_t size - Размер типа данных в очереди
int count_req - Число запросов в очереди
int sum_len - Средняя длина очереди
int tmp_len - Текущая длина очереди
int sum_time - Общее время работы с очередью
int out_req - Число запросов на выход в очереди
int in_req - Число запросов на вход в очереди

Описание алгоритма

1. Выводится меню данной программы.
2. Пользователь вводит номер команды из предложенного меню.

3. Пока пользователь не введет 0 (выход из программы), ему будет предложено вводить номера команд и выполнять действия по выбору.

НАБОР ТЕСТОВ

	Название теста	**Пользователь вводит**	**Вывод**
1	Некорректный ввод команды	45 (разрешено от 0 до 4)	Invalid command entered, please re-enter!!!
2	Пустой ввод	Пустой ввод.	Invalid command entered, please re-enter!!!
3	Команда 1	1	Вывод временной характеристики про очередь в виде массива.
4	Команда 2	2	Вывод временной характеристики про очередь в виде односвязного списка.
5	Команда 3 (неверный выбор промежутка времени)	8 (разрешено от 1 до 4) 0 s >	ERROR!!! Invalid number input! Please choose some command!

6	Команда 3 (неверный ввод границ, например буква или какой-либо символ)	D 4 3 f @ 7 3 «	ERROR!!! Invalid number input! Please choose some command!
7	Команда 3	Все введено верно	Выведены новые значения для временных промежутков
8	Команда 4	4	Вывод времени выполнения операций над очередью
9	Команда 0	0	Выход из программы

Оценка эффективности

Работа ОА

На массиве

	Число заявок 1-го типа	***Число заявок 2-го типа***	***Время моделирования (ус.е.в.)***	***Время работы (реальное время в мкс.)***
1	1002	1950	3015,3096	3350
2	1000	2003	2976,8032	3087
3	1000	2002	3020,4946	2913

4	1001	2006	3001,5489	3778
5	1001	1978	2972,8102	3625
6	1000	2103	3069,5333	3980
7	1000	2029	3012,9536	2973
8	1003	2031	3051,2047	3161
9	1001	1973	3001,2885	3960
10	1001	2034	3050,1438	2955
Среднее	1000,9	2010,9	3017,2090	3378,2

На списке

	Число заявок 1-го типа	***Число заявок 2-го типа***	***Время моделирования (ус.е.в.)***	***Время работы (реальное время в мкс.)***
1	1001	2010	2957,4566	15337
2	1000	2019	3002,0350	7506
3	1001	2030	3010,2803	12335
4	1001	2083	3126,2976	9821
5	1001	1979	2985,8095	11371
6	1000	1985	3020,7205	5907
7	1000	2019	2992,7828	6428
8	1000	2006	2990,4523	9666
9	1000	2017	3023,7761	9185
10	1000	2033	3026,5663	5768
Среднее	1000,4	2018,1	3013,6177	9332,4

Операции над очередью

Добавление элемента (в тактах процессора)

Массив	***Список***
466	5237

Удаление элемента (в тактах процессора)

Массив	***Список***
581	929

Память (в байтах)

Количество элементов	***Массив***	***Список***
10	10	160
100	100	1600
1000	1000	16000
10000	10000	160000

РАСЧЕТ ВРМЕНИ РАБОТЫ ОЧЕРЕДИ

Теоретический расчет времени моделирования : (среднее время прихода заявки 1 типа или среднее время обработки заявки 1 типа) * (количество). Выбирается большее время.

1 тип – так как у него абсолюный преоритет, количество равно 1000.

При данных временых границах :

T1 : 1..5

T2 : 0..3

T3 : 0..4

T4 : 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 : 2000 (время моделирования / среднее время прихода 2 заявки) = 2000

Число заявок 2 типа, вышедших : 2000 (время, когда ОА не работает (в отношении 1 очереди) / среднее время прихода 2 заявки) (вышедшие + оставшиеся в очереди, в результатах)

Практические результаты :

```
Общее время моделирования (в усл. ед. в.):       2990.257771
Погрешность работы ОА:   0.324741%

Среднее время обработки заявки 1 очереди:        3.000000
Среднее время обработки заявки 2 очереди:        1.500000
Число вошедших в 1 очередь:      1000
Число вышедших из 1 очереди:     1000
Число вошедших во 2 очередь:     1993
Число вышедших из 2 очереди:     1630
Число выброшенных заявок из 2 очереди: 1301
Время работы (реальное)(в мкс): 3412

Погрешность ввода 1 очередь:     0.325799%
Погрешность ввода 2 очередь:     0.025341%
Время не работы ОА (в усл. ед. в.):      13.097420
```

Из результатов видно, что погрешность в пределах допустимой по заданию.

<u>При данных временых границах :</u> (время обратотки больше, чем время прихода)

T1 : 1..5

T2 : 0..3

T3 : 0..10

T4 : 0..1

Теоретические результаты :

Время моделирование равно 5000 е.в.

Время обработки заявок 1 типа : (среднее время обработки 1 типа) * (количество) = 5 * 1000 = 5000

Время, когда ОА не работает (в отношении 1 очереди): (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 (в отношении 1 очереди) / среднее время прихода 2 заявки) (вышедшие + оставшиеся в очереди, в результатах)

Практические результаты :

```
Общее время моделирования (в усл. ед. в.):       5038.039835
Погрешность работы ОА:   0.760797%

Среднее время обработки заявки 1 очереди:        3.000000
Среднее время обработки заявки 2 очереди:        1.500000
Число вошедших в 1 очередь:      1702
Число вышедших из 1 очереди:     1000
Число вошедших во 2 очередь:     3368
Число вышедших из 2 очереди:     5
Число выброшенных заявок из 2 очереди: 1
Время работы (реальное)(в мкс): 2652

Погрешность ввода 1 очередь:     1.348941%
Погрешность ввода 2 очередь:     0.277095%
Время не работы ОА (в усл. ед. в.):      2.160881
```

Из результатов видно, что погрешность в пределах допустимой по заданию.

<u>При данных временых границах :</u> (время прихода и обратотки заявки 2 типа одинаковое)

T1 : 1..5

T2 : 0..1

T3 : 0..4

T4 : 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 :

```
Общее время моделирования (в усл. ед. в.):       2959.113330
Погрешность работы ОА:   1.362889%

Среднее время обработки заявки 1 очереди:        3.000000
Среднее время обработки заявки 2 очереди:        0.500000
Число вошедших в 1 очередь:      1000
Число вышедших из 1 очереди:     1000
Число вошедших во 2 очередь:     5954
Число вышедших из 2 очереди:     1307
Число выброшенных заявок из 2 очереди: 2517
Время работы (реальное)(в мкс): 4569

Погрешность ввода 1 очередь:     1.381720%
Погрешность ввода 2 очередь:     0.604460%
Время не работы ОА (в усл. ед. в.):      0.113874
```

Из результатов видно, что погрешность в пределах допустимой по заданию.

Ответы на контрольные вопросы

1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с «хвоста», а исключение – с «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?

При реализации списком, под каждый новый элемент выделяется память размером sizeof(element) + 8 байт (для указателя) в куче, для каждого элемента отдельно.

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

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 статически, то после завершении 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

К недостаткам очереди в виде списка можно отнести тот факт, что используется большее количество памяти, так как помимо самих элементов необходимо хранить указатели. Также при работе в очередями-списками может возникнуть фрагментация памяти. К преимуществам можно отнести тот факт, что очередь-список позволяет воспользоваться памятью, ограниченной лишь объёмом оперативной памяти компьютера, а также операции удаления и добавления элемента в очередь легче реализовать, чем с очередью-массивом, но при выполнении этих операций выполняется выделение или освобождение памяти, что может перевести к ошибке.

К недостаткам очереди в виде массива можно отнести то, что такая очередь будет ограничена по памяти и может возникнуть переполнение. Преимущество очереди-массива над очередью-списком — операции удаления и добавления элемента выполняются намного быстрее.

Приложение 1

Вывод программы на вызов различных команд

команда 1

```
Обработано заявок 1го типа:      600
Текущая длина First Queue:       0
Средняя длина First Queue:       0.570000
Текущая длина Second Queue:      258
Средняя длина Second Queue:      104.784562

Обработано заявок 1го типа:      700
Текущая длина First Queue:       1
Средняя длина First Queue:       0.566024
Текущая длина Second Queue:      284
Средняя длина Second Queue:      131.035892

Обработано заявок 1го типа:      800
Текущая длина First Queue:       0
Средняя длина First Queue:       0.565000
Текущая длина Second Queue:      271
Средняя длина Second Queue:      150.413651

Обработано заявок 1го типа:      900
Текущая длина First Queue:       1
Средняя длина First Queue:       0.565797
Текущая длина Second Queue:      307
Средняя длина Second Queue:      163.089864

Общее время моделирования (в усл. ед. в.):       3043.443165
Погрешность работы ОА:   1.448106%

Среднее время обработки заявки 1 очереди:        3.000000
Среднее время обработки заявки 2 очереди:        1.500000
Число вошедших в 1 очередь:      1000
Число вышедших из 1 очереди:     1000
Число вошедших во 2 очередь:     2027
Число вышедших из 2 очереди:     1739
Число выброшенных заявок из 2 очереди: 1315
Время работы (реальное)(в мкс): 3351

Погрешность ввода 1 очередь:     1.427435%
Погрешность ввода 2 очередь:     0.096705%
Время не работы ОА (в усл. ед. в.):      8.878957
```

команда 2

```
Обработано заявок 1го типа:      300
Текущая длина First Queue:       1
Средняя длина First Queue:       0.540765
Текущая длина Second Queue:      70
Средняя длина Second Queue:      50.385604

Обработано заявок 1го типа:      400
Текущая длина First Queue:       1
Средняя длина First Queue:       0.535581
Текущая длина Second Queue:      53
Средняя длина Second Queue:      52.906117

Обработано заявок 1го типа:      500
Текущая длина First Queue:       1
Средняя длина First Queue:       0.538462
Текущая длина Second Queue:      78
Средняя длина Second Queue:      56.220430

Обработано заявок 1го типа:      600
Текущая длина First Queue:       0
Средняя длина First Queue:       0.543333
Текущая длина Second Queue:      118
Средняя длина Second Queue:      62.081747

Обработано заявок 1го типа:      700
Текущая длина First Queue:       0
Средняя длина First Queue:       0.538571
Текущая длина Second Queue:      116
Средняя длина Second Queue:      69.260376
```

```
Общее время моделирования (в усл. ед. в.):       2997.336912
Погрешность времени моделирования:       0.088770%

Число вошедших в 1 очередь:      1000
Число вышедших из 1 очереди:     1000
Число вошедших во 2 очередь:     1996
Число вышедших из 2 очереди:     1788
Число выброшенных заявок из 2 очереди: 592
Время работы (реальное)(в мкс): 11195
Среднее время обработки заявки 1 очереди:        3.000000
Среднее время обработки заявки 2 очереди:        1.500000
Погрешность ввода 1 очередь^     0.088848%
Погрешность ввода 2 очередь^     0.111329%
Время не работы ОА (в усл. ед. в.):      7.802959

Количество повторно используемых адресов: 3361
Количество неиспользуемых адресов: 19
Неиспользуемые адреса:
0x40a5f0
0x40a8f0
0x40aa30
0x40aa50
0x40aa70
0x40aab0
0x40ab30
0x40abf0
0x40ac50
0x40a630
0x40a670
0x40a210
0x40acb0
0x40acd0
0x40ad70
0x409c50
0x409b90
0x40adb0
0x40a310
```

команда 3

```
Для помощи нажмите 5
ВАШ ВЫБОР: 3
Change the processing time.
1: min = 1.000000; max = 5.000000
2: min = 0.000000; max = 3.000000
3: min = 0.000000; max = 4.000000
4: min = 0.000000; max = 1.000000
What interval to change?
4
Input left and right borders: 4
5
AFTER
1: min = 1.000000; max = 5.000000
2: min = 0.000000; max = 3.000000
3: min = 0.000000; max = 4.000000
4: min = 4.000000; max = 5.000000
```

команда 4

```
Для помощи нажмите 5
ВАШ ВЫБОР: 4
Вывод сравнений по времени
ДОБАВЛЕНИЕ
Очередь-массив   929
Очередь-список   3065

УДАЛЕНИЕ
Очередь-массив   681
Очередь-список   1751

ПАМЯТЬ
Колчество элементов = 10
Очередь-массив   10
Очередь-список   90

Колчество элементов = 100
Очередь-массив   100
Очередь-список   900

Колчество элементов = 1000
Очередь-массив   1000
Очередь-список   9000

Колчество элементов = 10000
Очередь-массив   10000
Очередь-список   90000
```