|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 ???

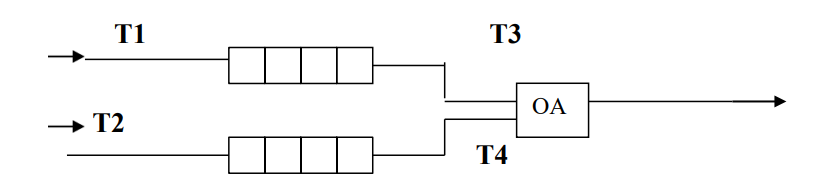
Группа ИУ7 – 3?Б

Преподаватель Барышникова М.Ю.

*2021*

**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 Т1и Т2, равномерно распределенными от 1 до 5 и от 0 до 3 единиц времени (е.в.) соответственно. В ОА они поступают из "головы" очередипо одной и обслуживаются также равновероятно за времена Т3и Т4, распределенные от 0 до 4е.в. и от 0 до 1е.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 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: номер команды от 0 до 4.
2. Вещественное число: интервал работы (время)

Выходные данные:

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

Функции программы:

1. Моделирование очереди в виде массива.
2. Моделирование очереди в виде списка.
3. Изменение времени работы.
4. Сравнение времени работы массива и списка

Обращение к программе:

Запускается через терминал

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: число, большее чем 4 или меньшее, чем 0.

На выходе: сообщение «Invalid input»

2. Некорректный ввод выбора времени.

На входе: отрицательное число.

На выходе: сообщение «Invalid input»

**Описание структуры данных**

*Структура для описании времени*

**typedef struct times**

**{**

**double Tmin;**

**double Tmax;**

**} time\_t;**

Поля структуры:

*Tmin*  - нижняя граница времени

*Tmax*  - верхняя граница времени

*Структура для описания узла списка.*

**typedef struct node**

**{**

**char data;**

**struct node \*next;**

**} node\_t;**

Поля структуры:

*data*  - данные в узле

*\*next*  - указатель на следующий узел

*Структура для описания всей информации про очередь.*

**typedef struct queue**

**{**

**void\* start;**

**void\* end;**

**void\* p\_in;**

**void\* p\_out;**

**int count\_req;**

**int out\_req;**

**int in\_req;**

**} queue\_t;**

Поля структуры:

*void\* start* - Адрес нижней границы очереди

*void\* end* - Адрес верхней границы очереди

*void\* p\_in* - Указатель на "хвост" очереди

*void\* p\_out*  - Указатель на "голову" очереди

*int count\_req* - Число запросов в очереди

*int out\_req* - Число вышедших вопросов

*int in\_req*  - Число пришедших запросов

**Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользователь вводит** | **Вывод** |
| 1 | Некорректный ввод команды | 32 (разрешено от 0 до 4) | Invalid input |
| 2 | Пустой ввод | Пустой ввод. | Invalid input |
| 3 | Команда 1 | 1 | Характеристика очереди в виде массива. |
| 4 | Команда 2 | 2 | Характеристика очереди в виде списка. |
| 5 | Команда 3 | 3  2 1 4 | Время изменено. |
| 6 | Команда 4 | 4 | Сравнение массива и списка. |

**Оценка эффективности**

***Добавление элемента (в тактах)***

|  |  |
| --- | --- |
| ***Массив*** | ***Список*** |
| 80 | 3840 |

***Удаление элемента (в тактах)***

|  |  |
| --- | --- |
| ***Массив*** | ***Список*** |
| 100 | 160 |

***Память (в байтах)***

|  |  |  |
| --- | --- | --- |
| ***Количество элементов*** | ***Массив*** | ***Список*** |
| 10 | 10 | 90 |
| 100 | 100 | 900 |
| 1000 | 1000 | 9000 |
| 10000 | 10000 | 90000 |

# **Расчет времени работы очереди**

*Теоретический расчет времени моделирования*: max(среднее время прихода заявки 1 типа или среднее время обработки заявки 1 типа) \* (количество).   
  
1 тип – так как у него абсолюный преоритет, количество равно 1000.

При данных временых границах :  
Т1 : 1..5  
Т2 : 0..3  
Т3 : 0..4  
Т4 : 0..1

***Теоретические результаты :***

Время моделирование равно 3000 е.в.  
Число заявок 1 типа, вошедших : 1000, вышедших : 1000  
Число заявок 2 типа, вошедших : 2000   
Число заявок 2 типа, вышедших : 2000

***Практические результаты:***

Время моделирование равно 3045 е.в.  
Число заявок 1 типа, вошедших : 1000, вышедших : 1000  
Число заявок 2 типа, вошедших : 2023  
Число заявок 2 типа, вышедших : 1695

Погрешность: 1.5%

**Ответы на контрольные вопросы**

*1. Что такое FIFO и LIFO?*

FIFO - «первым пришел — первым вышел» («First In – First Out») . Это метод для обработки структур данных, где первый элемент обрабатывается первым, а новейший элемент обрабатывается последним. LIFO — «последним пришел – первым ушел» («Last In - First Out»). Это метод для обработки структур данных, где последний элемент обрабатывается первым, а первый элемент обрабатывается последним.

*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 статический, 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. От чего зависит эффективность физической реализации очереди?*

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. 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.

*7. 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.

*8. Для чего нужен алгоритм «близнецов»?*

Алгоритм близнецов нужен для выделения памяти. Идея этого алгоритма состоит в том, что организуются списки свободных блоков отдельно для каждого размера 2k, 0 <= k <= m. Вся область памяти кучи состоит из 2m слов, которые, можно считать, имеют адреса с 0 по 2m – 1. Первоначально свободным является весь блок из 2m слов. Далее, когда требуется блок из 2k слов, а свободных блоков такого размера нет, расщепляется на две равные части блок большего размера; в результате появится блок размера 2k (т.е. все блоки имеют длину, кратную 2). Когда один блок расщепляется на два (каждый из которых равен половине первоначального), эти два блока называются *близнецами*. Позднее, когда оба близнеца освобождаются, они опять объединяются в один блок.

*9.* *Какие дисциплины выделения памяти вы знаете?*

Дисциплины выделения памяти решают вопрос: какой из свободных участков должен быть выделен по запросу. Выбор дисциплины распределения не зависит от способа учета свободной памяти. Две основные дисциплины сводятся к принципам "самый подходящий" и "первый подходящий". По дисциплине "самый подходящий" выделяется тот свободный участок, размер которого равен запрошенному или превышает его на минимальную величину. По дисциплине "первый подходящий" выделяется первый же найденный свободный участок, размер которого не меньше запрошенного.

*10. 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.

*11. 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**

Недостатком очереди в виде массива является то, что такая очередь будет ограничена по памяти. Однако, явное преимущество — это то, что операции удаления и добавления элемента выполняются очень быстро.

К недостаткам очереди в виде списка можно отнести тот факт, что используется большее количество памяти, так как помимо самих элементов необходимо хранить указатели. К преимуществам можно отнести то, что очередь-список позволяет воспользоваться памятью, ограниченной лишь объёмом оперативной памяти компьютера. Также при использовании списка происходит фрагментация памяти.