|  |  |
| --- | --- |
| **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 Лазутин Александр Владимирович

Группа ИУ7 – 33Б

Проверяющий Рыбкин Ю. А.

*2022 г.*

Оглавление

[Описание условия задачи 4](#_Toc120885697)

[Описание технического задания 5](#_Toc120885698)

[Аварийные ситуации: 7](#_Toc120885699)

[Описание структуры данных 8](#_Toc120885700)

[Описание алгоритма 9](#_Toc120885701)

[Набор тестов 10](#_Toc120885702)

[Оценка эффективности 13](#_Toc120885703)

[Тестирование задания 14](#_Toc120885704)

[Контрольные вопросы 14](#_Toc120885705)

[Вывод 17](#_Toc120885730)

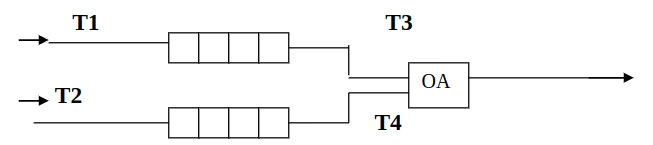
# Описание условия задачи

Разработать программу работы со стеком, реализующую операции 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 время моделирования, время простоя ОА, количество вошедших в систему и вышедших из нее заявок первого и второго типов.

# 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" очереди по одной и обслуживаются также равновероятно за времена Т3 и Т4, распределенные от 0 до 4 е.в. и от 0 до 1 е.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 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, представляющее собой пункт меню:

целое число в диапазоне от 0 до 10

2. Дополнительный ввод: поле типа int в зависимости от команды

Выходные данные:

Результат выполнения команды или сообщение об ошибке

Функции программы:

0. Выйти из программы

Очередь в виде массива:

1. Добавить элемент

2. Удалить элемент

3. Распечатать очередь

Очередь в виде списка:

4. Добавить элемент

5. Удалить элемент

6. Распечатать очередь

7. Распечатать массив освободившихся адрессов

Задание:

8. Для массива

9. Для списка

10. Замеры времени и памяти

# Аварийные ситуации:

1. Неверно введен пункт меню

(не число или число меньшее 0 или больше 10)

2. Достигнут максимально возможный размер очереди (как для массива, так и для списка)

(добавлено более 5 тысяч элементов)

3. Неверно введено число элементов для теста по времени

(не число или число меньшее 1 или большее 1000)

# Описание структуры данных

Структура для хранения элемента очереди (список)

typedef struct node\_s

{

int elem;

struct node\_s \*next;

} node\_t;

Поля структуры:

2. int elem — значение текущего элемента в очереди

3. struct node\_s \*next — указатель на следующий элемент очереди

*Структура для хранения указателя на голову и хвост и размера очереди (список)*

typedef struct list\_s

{

node\_t \*head;

node\_t \*tail;

int size;

} node\_t;

Структура для хранения очереди (массив)

typedef struct arr\_s

{

int arr[5000];

int begin;

int end;

int size;

} arr\_t;

Поля структуры:

1. int arr[] — массив элементов очереди

2. int begin — голова очереди

3. int end — хвост очереди

4. int size — размер очереди

Структура для хранения массива освободившихся адресов, которая хранит в себе массив указателей на элементы списка, которые освободились и длину этого массива.

typedef struct list\_s

{

node\_t \*arr\_clear[1000];

int len;

} arr\_clear\_t;

# Описание алгоритма

1. Выводится меню программы.

2. Пользователь вводит номер любой команды, которой соответствует свое назначение.

3. Работа программы осуществляется, пока не будет совершена ошибка при вводе или пока не будет введен 0.

# Набор тестов

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод пункта меню | aaa | Ошибка: пункты меню это числа от 0 до 10 |
| 2 | Некорректный ввод количества элементов для замеров памяти и времени  (не число) | aaa | Ошибка: неверно введено количество элементов |
| 3 | Некорректный ввод количества элементов для замеров памяти и времени  (число меньшее 1 или большее 1000) | 0 | Ошибка: неверно введено количество элементов |
| 4 | Добавить элемент в очередь (массив и список) | Команда 1 или 4 | Элемент добавлен в очередь |
| 5 | Удалить элемент из очередь (массив и список) | Команда 2 или 5 | Элемент удален из очереди |
| 6 | Распечатать очередь  (массив и список) | Команда 3 или 6 | Распечатанная очередь |
| 7 | Переполнение очереди (массив и список) | Попытка добавить элемент в очередь, если уже добавлено 5000 элементов | Ошибка: очередь переполнена |
| 8 | Очередь пуста  (массив и список) | Попытка распечатать или удалить элемент из очереди, если в ней ничего нет | Очередь пустая |
| 9 | Распечатать массив освободившихся адресов  (список) | Команда 7 | Печатается массив освободившихся адресов |
| 10 | Распечатать пустой массив адресов (список) | Попытка распечатать массив освободившихся адресов, если он пуст | Массив освободившихся адресов пуст |
| 11 | Выполнить задачу (массив) | Команда 8 | Вывод результатов выполнения задания со всем временами |
| 12 | Выполнить задачу  (список) | Команда 9 | Вывод результатов выполнения задания со всем временами |
| 13 | Вывод замеров времени и памяти | Команда 10 | Результат замеров времени добавления и удаления элементов и замеров памяти |
| 14 | Выход из программы | Команда 0 | Выход из программы, очистка консоли |

# Оценка эффективности

Добавление элементов (в тиках)

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 691 | 1023 |
| 100 | 4724 | 11045 |
| 500 | 16204 | 38938 |
| 1000 | 38912 | 96032 |

Удаление элементов (в тиках)

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 302 | 754 |
| 100 | 2035 | 4803 |
| 500 | 6892 | 17573 |
| 1000 | 14032 | 46741 |

Замеры памяти

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 48 | 160 |
| 100 | 408 | 1600 |
| 500 | 2008 | 8000 |
| 1000 | 4008 | 16000 |

# Тестирование задания

Время моделирования заявок  
Т1: от 1 до 5 Т2: от 0 до 3  
Т3: от 0 до 4 Т4: от 0 до 1

***Теоретические вычисления:***

Заявки первого типа обрабатываются в среднем 2 е.в, а заявки второго типа – 0.5 е.в. Число заявок 1 типа, вошедших = 1000, вышедших = 1000, число заявок 2 типа, вошедших: 2000, вышедших = 2000. Следовательно время моделирования должно составлять: 1000 \* 2 + 2000 \* 0.5 = 3000 е.в.

Погрешность времени моделирования:

(3022.343 – 3000) / 3000 ~ 0.007 = 0.7%

Погрешность появляется из-за того, что в теоретических расчетах не учитывается время простоя автомата.

# **Контрольные вопросы**

# **1. Что такое FIFO и LIFO?**

# **FIFO** и **LIFO** — методы оценки товарно-материальных ценностей. При учёте запасов однородного товара, купленного в разное время по разной цене, приходится определять: что выдавать первым; и физически, и на бумаге. По методу *FIFO* первым выдаётся самый старый из пришедших товаров, по методу *LIFO* — самый новый.

# **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 на стеке 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 из нее удаляются.

# **5. От чего зависит эффективность физической реализации очереди?**

# Очередь на список работает медленнее и под каждый элемент требуется больше памяти, чем при хранении очереди на статическом массиве. Однако при хранении списком количество элементов в очереди ограничено только размером оперативной памяти, и список при количестве элементов до 1500 выигрывает по 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 всегда ограничен размером массива.

# При реализации же очереди на списке при добавлении и удалении элемента происходит выделение 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».**

# Алгоритм близнецов значительно снижает фрагментацию памяти и резко ускоряет поиск блоков. Наиболее важным преимуществом этого подхода является то, что даже в наихудшем случае время поиска не превышает. Это делает алгоритм близнецов труднозаменимым для ситуаций, когда необходимо гарантированное время реакции - например, для задач реального времени.

# **9. На что необходимо обратить внимание при тестировании программы?**

# Проверить правильность работы программы при различном заполнении очередей, т.е., когда время моделирования определяется временем обработки заявок и когда определяется временем прихода заявок;

# O отследить переполнение очереди, если очередь в программе ограничена.

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

# В си работать с динамической памятью можно при помощи соответствующих функций распределения памяти (calloc, malloc, free), для чего необходимо подключить библиотеку malloc.h

# 

# 

# Вывод

При реализации очереди стоит учитывать два осноных 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, но нужно предусмотреть дополнительную проверку, чтобы не допустить фрагментации памяти.

Если реализовать очередь на списке, то операциии добавления и удаления элементов будут на нем происходить дольше из-за постоянной необходимости выделения памяти под следующий элемент списка

Массив быстрее списка :

* при удалении примерно в 3 раза
* при добавлении примерно в 2 раза