|  |  |
| --- | --- |
| **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 Ковалец Кирилл

Группа ИУ7 – 33Б

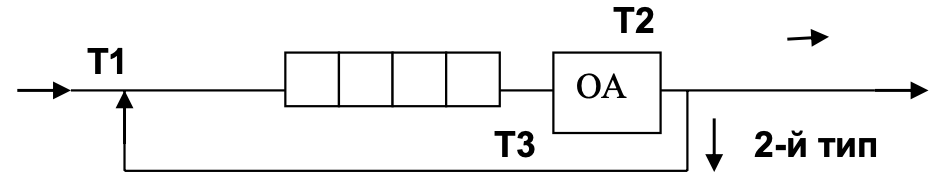
*2020 г.*

**Описание условия задачи**

Провести сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании указанных структур данных, оценить эффективности программы по времени и по используемому объему памяти.

**Описание технического задания**

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



Заявки 1-го типа поступают в "хвост" очереди по случайному закону с интервалом времени **Т1,** равномерно распределенным от **0 до 5** единиц времени (е.в.). В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время **Т2** от **0 до 4** е.в., после чего покидают систему.

Единственная заявка 2-го типа постоянно обращается в системе, обслуживаясь в ОА равновероятно за время **Т3** от **0 до 4** е.в. и возвращаясь в очередь не далее 4-й позиции от "головы". В начале процесса заявка 2-го типа входит в ОА, оставляя пустую очередь. (Все времена – **вещественного** типа)

Смоделировать процесс обслуживания первых 1000 заявок **1-го типа.** Выдавать после обслуживания каждых 100 заявок **1-го типа** информацию о текущей и средней длине очереди, количестве вошедших и вышедших заявок и о среднем времени пребывания заявок в очереди. В конце процесса выдать общее время моделирования, время простоя аппарата, количество вошедших в систему и вышедших из нее заявок первого типа и количество обращений заявок второго типа. По требованию пользователя выдать на экран адреса элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

**Входные данные:**  
Номер команды, отвечающий за определённое действие.

**Команды:**

1. Добавить элементы в очередь;

2. Добавить случайные элементы в очередь;

3. Удалить элементы из очереди;

4. Вывести текущее состояние очереди (массив);

5. Вывести текущее состояние очереди (список);

6. Вывести массив освобождённых адресов;

0. Выйти из программы.

**Выходные данные:**

1. Результат выполнения определённой команды;
2. Результат сравнения обработки 2-х типов стеков по времени и памяти.

**Обращение к программе:**  
Запускается через терминал командой make run

**Сообщения при аварийных ситуациях:**

1. Не удалось прочитать номер команды;
2. Номер команды должен быть >= 0 и <= 6;
3. Не удалось прочитать кол-во добавляемых элементов;
4. Кол-во добавляемых элементов не может быть отрицательным;
5. Произошло переполнение очереди;
6. Не удалось прочитать элемент очереди;
7. Не удалось прочитать кол-во удаляемых элементов;
8. Кол-во удаляемых элементов не может быть отрицательным;
9. В очереди нет столько элементов;
10. Не удалось выделить память.

**Описание структуры данных**

**turn\_array\_t** - структура, содержащая информацию об очереди, реализованного с помощью массива.

typedef struct turn\_array

{

int \*head;

int \*tail;

int size;

} turn\_array\_t;

**Поля структуры:**

1. int \* head – указатель на “голову” очереди;
2. int \* tail – указатель на “хвост” очереди;
3. int size – кол-во элементов очереди.

**elem\_turn\_list\_t** - структура, содержащая информацию об элементе очереди, реализованного с помощью односвязного списка.

typedef struct elem\_turn\_list

{

int64\_t start;

int elem;

struct elem\_turn\_list \*next;

} elem\_turn\_list\_t;

**Поля структуры:**

1. int64\_t start – время добавления элемента в очередь (в тиках);
2. int elem – элемент очереди;
3. struct elem\_turn\_list \*next – указатель на следующий элемент очереди.

**turn\_list\_t** - структура, содержащая информацию о всей очереди, реализованного с помощью односвязного списка.

typedef struct turn\_list

{

elem\_turn\_list\_t \*head;

int size;

} turn\_list\_t;

**Поля структуры:**

1. elem\_turn\_list\_t \* head – указатель на “голову” очереди;
2. int size – кол-во элементов очереди.

**array\_of\_freed\_areas\_t** - структура, содержащая информацию о массиве освобождённых адресов.

typedef struct array\_of\_freed\_areas

{

elem\_turn\_list\_t \*\*array;

int size;

} array\_of\_freed\_areas\_t;

**Поля структуры:**

1. elem\_turn\_list\_t \*\*array – массив освобождённых адресов;
2. int size – кол-во элементов в массиве.

**Описание алгоритма**

1. Выводится меню программы (каждой команде присвоен номер);
2. Пользователь вводит номер команды, который отвечает за определённое действие;
3. Ввод осуществляется до того момента, пока не будет введён 0, являющийся признаком выхода из программы.

**Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название теста** | **Входные данные** | **Результат** |
| 2 | Номер команды - число | k | Не удалось прочитать номер команды |
| 3 | Номер команды >= 0 и <=6 | 8 | Номер команды должен быть >= 0 и <= 6 |
| 4 | Кол-во добавляемых элементов в очередь - число | k | Не удалось прочитать кол-во добавляемых элементов |
| 5 | Кол-во добавляемых элементов в очередь > 0 | -1 | Кол-во добавляемых элементов не может быть отрицательным |
| 6 | Максимальный размер очереди - 1000 | В очереди содержалось 5 элементов, попытались добавить ещё 996 | Произошло переполнение очереди |
| 7 | Элемент очереди – целое число | k | Не удалось прочитать элемент очереди |
| 8 | Кол-во удаляемых элементов - число | k | Не удалось прочитать кол-во удаляемых элементов |
| 9 | Кол-во удаляемых элементов в очереди > 0 | -1 | Кол-во удаляемых элементов не может быть отрицательным |
| 10 | Попытка удалить элементов больше, чем есть в очереди | В очереди содержалось 5 элементов, попытались удалить 6 | В очереди нет столько элементов |
| 11 | Добавление элементов в очередь | Добавление 5 элементов в незаполненную очередь | Элементы успешно добавлены в очередь |
| 12 | Добавление элементов в заполненную очередь | Попытка добавить 1 элемент | Очередь заполнена |
| 13 | Удаление элементов из стека | Удаление 5 элементов из заполненного стека | Элементы успешно удалены из стека |
| 14 | Удаление элементов из пустой очереди | Попытка удалить 1 элемент | Очередь пуста |
| 15 | Запрос на печать массива, содержащего адреса освобождённых элементов очереди (список) | Команда 6 | Печать массива |
| 16 | Вывести текущее состояние очереди | Команда 4 | Печать элементов очереди в столбец  (для списка выводится адрес каждого элемента очереди) |
| 17 | Сравнение работы двух очередей (списка и массива) по памяти и времени | Запрос на сравнение очередей  (Номер команды – 2) | Печать требуемых по заданию значений |
| 18 | Выход из программы | Команда 0 | Выход из программы |

**Оценка эффективности**

Сортировка каждой таблицы будет измеряться в тактах процессора (процес- сор со средней частотой 2.3gHz).

**Сравнение эффективности**

**Добавление элементов (в тиках)**

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 1233 | 3802 |
| 100 | 2328 | 17254 |
| 500 | 7387 | 77008 |
| 1000 | 13765 | 155498 |

**Удаление элементов (в тиках)**

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 555 | 2676 |
| 100 | 39254 | 16239 |
| 500 | 605261 | 75833 |
| 1000 | 2382834 | 155399 |

**Сравнение памяти**

|  |  |  |
| --- | --- | --- |
| Размер | Массив | Список |
| 10 | 4024 | 248 |
| 100 | 4024 | 2408 |
| 500 | 4024 | 12008 |
| 1000 | 4024 | 24008 |

# **Тесты**

Интервал времени прихода в очередь заявки 1-го типа (T1): от 0 до 5 е.в.;

Интервал времени обработки заявки 1-го типа в очереди: от 0 до 4 е.в.;

Интервал времени обработки заявки 2-го типа в очереди: от 0 до 4 е.в.

Кол-во вошедших заявок в очередь - 1001.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип сравнения | Ожидаемое время | Реальное время | Погрешность |
| Время моделирования | 2503 | 2533 | 1.19% |
| Время работы аппарата | 2000 | 2037 | 1.85% |
| Время простоя | 500 | 486 | 2.88% |
| Время работы аппарата + время простоя | 2500 | 2523 | 0.92% |

Интервал времени прихода в очередь заявки 1-го типа (T1): от 0 до 4 е.в.;

Интервал времени обработки заявки 1-го типа в очереди: от 0 до 5 е.в.;

Интервал времени обработки заявки 2-го типа в очереди: от 0 до 5 е.в.;

Кол-во вошедших заявок в очередь - 1099.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип сравнения | Ожидаемое время | Реальное время | Погрешность |
| Время моделирования | 2198 | 2250 | 2.36% |
| Время работы аппарата | 2500 | 2449 | 1.96% |
| Время простоя | ~0 | 2 | - |
| Время работы аппарата + время простоя | 2500 | 2451 | 1.99% |

# **Ответы на контрольные вопросы**

**Что такое очередь?**

Очередь — структура данных, для которой выполняется правило FIFO, то есть первым зашёл — первым вышел. Вход находится с одной стороны очереди, выход — с другой.

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

При хранении кольцевым массивом: кол-во элементов \* размер одного элемента очереди. Память выделяется на стеке при компиляции, если массив статический. Либо память выделяется в куче, если массив динамический.

При хранении списком: кол-во элементов \* (размер одного элемента очереди + указатель на следующий элемент). Память выделяется в куче для каждого элемента отдельно.

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

При хранении кольцевым массивом память не освобождается, а просто меняется указатель на конец очереди. При хранении списком, память под удаляемый кусок освобождается.

**Что происходит с элементами очереди при ее просмотре?**

Эти элементы удаляются из очереди.

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

Зная максимальный размер очереди, лучше всего использовать кольцевой статический массив. Не зная максимальный размер, стоит использовать связанный список, так как такую очередь можно будет переполнить только если закончится оперативная память.

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

При использовании кольцевого массива тратится больше времени на обработку операций с очередь, а так же может возникнуть фрагментация памяти. При реализации статическим кольцевым массивом, очередь всегда ограничена по размеру.

**Что такое фрагментация памяти?**

Фрагментация памяти — разбиение памяти на куски, которые лежат не рядом друг с другом. Можно сказать, что это чередование свободных и занятых кусков памяти.

**На что необходимо обратить внимание при тестировании программы?**

Корректное освобождение памяти.

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

При запросе памяти, ОС находит подходящий блок памяти и записывает его в «настоящее» начало массиватаблицу» занятой памяти. При освобождении, ОС удаляет этот блок памяти из «настоящее» начало массиватаблицы» занятой пользователем памяти.

# **Вывод**

Если очередь реализована статическим массивом, то добавление в неё новых элементов будет происходить быстрее (в 3 раза для 10 элементов и в 11 раз для 1000 элементов), чем в очередь, реализованную списком. Это связанно с тем, что для хранения очереди в виде списка требуется выделить память для указателей на следующие элементы списка. Удаление элементов из массива происходит медленнее (в 2 раза для 100 элементов и в 15 раз для 1000 элементов), чем из списка, так как при удалении элемента из массива, требуется сдвинуть оставшиеся на один вправо. Чтобы и удаление элементов из массива было эффективнее по времени, надо использовать кольцевой массив.

Вариант хранения очереди в виде списка может выигрывать в том случае, если стек реализован статическим массивом (массив на 1000 элементов должен быть заполнен менее, чем на 20%). Так же, если не известен размер стека, то в таком случае стоит использовать списки (или динамические массивы).