Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
«РАБОТА С ОЧЕРЕДЬЮ»

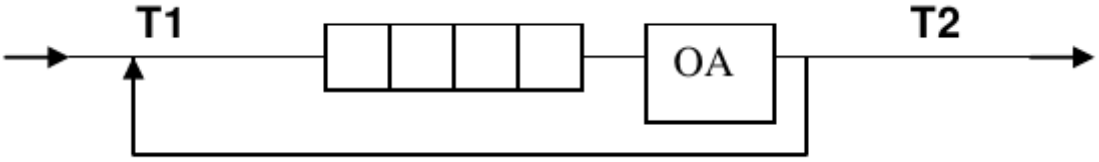
Студент, группа **Кононенко С., ИУ7-33Б**

2019 г.

Описание условия задачи

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок, выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди, а в конце процесса - общее время моделирования и количестве вошедших в систему и вышедших из нее заявок, количестве срабатываний ОА, время простоя аппарата. По требованию пользователя выдать на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

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



Заявки поступают в "хвост" очереди по случайному закону с интервалом времени Т1, равномерно распределенным от 0 до 6 единиц времени (е.в.). В ОА они поступают из "головы" очереди по одной и обслуживаются также равновероятно за время Т2 от 0 до 1 е.в., Каждая заявка после ОА вновь поступает в "хвост" очереди, совершая всего 5 циклов обслуживания, после чего покидает систему. (Все времена – вещественного типа) В начале процесса в системе заявок нет.

Техническое задание

Входные данные:

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

Выходные данные:

1. Результат моделирования работы ОА – данные о рабочем времени автомата, ожидаемое рабочее время автомата, погрешнсоть, число вошедших заявок, число вышедших заявок, число необрботанных заявок, число срабатываний автомата, время простоя автомата, количество переиспользованных адресов и адресов, взятых из новой памяти (в случае реализации на списке)).
2. Количественная характеристика сравнения вариантов моделирования ОА очереди.

Функция программы: программа выполняет ряд функций, указанных при её первом запуске. Она позволяет:

1. Ввести данные обслуживающего аппарата и вывести статистику работы.
2. Вывести количественную характеристику выполнения операций над очередью.

п. 1-2 выполняют действия над очередью, реализованной на основе массива.

3. Ввести данные обслуживающего аппарата и вывести статистику работы.
4. Вывести количественную характеристику выполнения операций над очередью.

п. 3-4 выполняют действия над очередью, реализованной на основе связного списка.

Обращение к программе: запускается из терминала.

Аварийные ситуации:

1. Некорректный ввод номера команды.
   На входе: число, большее чем 4 или меньшее, чем 0.
   На выходе: сообщение «Введено недопустимое значение! Повторите поппытку»
2. Некорректный ввод количества элементов очереди.
   На входе: отрицательное целое число, число, превышающее максимально допустимое число для количества элементов стека или буква.
   На выходе: сообщение «Введено недопустимое значение! Повторите поппытку.»
3. Некорректный ввод характеристик обслуживающего аппарата очереди.
   На входе: буква или, любой другой нечисловой символ или отрицательное число.
   На выходе: сообщение «Введено недопустимое значение! Повторите поппытку.»
4. Некорректный ввод характеристик обслуживающего аппарата очереди.
   На входе: правая граница интервала больше левой.
   На выходе: сообщение «ВПравая граница должна быть больше левой! Повторите попытку.»
5. Попытка создать новую очередь, при имеющейся в программе.
   На входе: попытка создания новой очереди.
   На выходе: сообщение «Очередь уже существует. Выход из программы...»

Структуры данных

Хранение заявки:

```
typedef struct
{
   double time_out;
   int num;
} task_t;
```

Поля структуры:

- ***double* time_out –** время, затрачиваемое на обработку заявки;
- ***int* num –** количество раз, сколько была обработана заявка;

Реализация очереди на основе массива:

```
typedef struct
{
   unsigned capacity, size, rear, front;
   task_t *arr;
} queuearr_t;
```

Поля структуры:

- ***unsigned* capacity, size, rear, front –** максимальный допустимый размер, текущий размер, хвост и голова очереди;
- ***task_t* *arr –** указатель на массив заявок;

Реализация очереди на основе линейного односвязного списка:

```
typedef struct
{
   unsigned capacity, size;
   queuenode_t *front, *rear;
} queuelist_t;
```

Поля структуры:

- ***unsigned* capacity, size –** максимальный допустимый размер и текущий размер очереди;
- ***queuenode_t* *front, *rear –** указатели на голову и хвост очереди;

Реализация элемента очереди:

```
typedef struct queuenode
{
   task_t task;
   struct queuenode *next;
} queuenode_t;
```

Поля структуры:

- ***task_t* task –** заявка ОА;
- ***struct queuenode* *next –** указатель на следующий элемент очереди;

Реализация массива свободных областей:

```
typedef struct
{
   size_t *arr;
   int capacity;
   int ind;
} arr_t;
```

Поля структуры:

- ***size_t* *arr –** указатель на массив адресов;
- ***int* capacity –** максимальная ёмкость массива;
- ***int* ind –** текущий незанятый элемент.

Алгоритм

1. Пользователь вводит номер команды из меню.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с двумя реализациями очереди – на основе массива или на основе линейного односвязного списка.
3. При выборе команды вывода количественной характеристики выполнения операций над очередью, выводится среднее значение добавления/удаления элементов из очереди на основе 1000 добавлений/удалений.
4. При выборе команды вывода статистики ОА, выводится статистика ОА после обработки каждой 100 заявки, а также общие данные, описанные в секции “Выходные данные”.

Тесты

	Тест	Пользовательский ввод	Результат
1	Некорректный ввод комманды	5	Введено недопустимое значение! Повторите попытку.
2	Некорректный ввод количества элементов очереди	-1	Введено недопустимое значение! Повторите поппытку
3	Некорректный ввод количества элементов очереди	A	Введено недопустимое значение! Повторите поппытку
4	Некорректный ввод характеристики ОА	Ввод интервала прихода: A 8	Введено недопустимое значение! Повторите поппытку
5	Некорректный ввод характеристики ОА	Ввод интервала прихода: 0 B	Введено недопустимое значение! Повторите поппытку
6	Некорректный ввод характеристики ОА	Ввод интервала прихода: 8 7	Правая граница должна быть больше левой! Повторите попытку.
7	Некорректный ввод характеристик ОА	Ввод количества обслуживаний одной заявки: -1	Введено недопустимое значение! Повторите поппытку
8	Попытка создания очереди при уже существующей	Попытка создания очереди	Очередь уже существует! Выход и программы...
9	Корректный ввод характеристик ОА	10000; 0 6; 0 1; 5	Вывод информации ОА и количественная характеристика моделирования
10	Вывод количественной характеристики выполнения операций над очередью	Выбор команды	Добавление элементов в очередь на основе **X**: **Время** Удаление элементов из очереди на основе **X**: **Время**

Оценка эффективности

Измерения эффективности реализаций очереди будут производиться в единицах измерения – тактах процессорв. Для измерения была специально написана ассемблерная функция, поэтому погрешность измерений минимальна. При записи результатов использовалось среднее количество тактов, полученное по результатам 1000 измерений.

Время добавления элемента (в тактах процессора):

Массив	Список
98	336

Время удаления элемента (в тактах процессора):

Массив	Список
55	182

Объём занимаемой памяти (в байтах):

Количество элементов	Массив	Список
10	144	224
100	1224	2024
1000	12024	20024

Контрольные вопросы

1. 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.

2. 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.

3. 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.

4. Что происходит с элементами очереди при ее просмотре?

Эти элементы удаляются из очереди.

5. 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.

6. 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.

7. 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.

8. На что необходимо обратить внимание при тестировании программы?

Корректное освобождение памяти.

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

При запросе памяти, ОС находит подходящий блок памяти и записывает его в «таблицу» занятой памяти. При освобождении, ОС удаляет этот блок памяти из «таблицы» занятой пользователем памяти.

Вывод

Использование связанных списков невыгодно при реализации очереди. Списки проигрывают как по памяти, так и по времени обработки. Но, когда заранее неизвестен максимальный размер очереди, то можно использовать связанный списки, так как в отличии от статического массива, списки ограничены в размерах только размером оперативной памяти. Стоит отметить, что при проведении тестов ни разу не наблюдалась фрагментация памяти (на моём ПК), но даже при этом, список проигрывает во времени обработки кольцевому массиву.