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

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
РАБОТА С ОЧЕРЕДЬЮ

Название предмета: Типы и структуры данных

Студент: Варламова Екатерина Алексеевна

Группа: ИУ7-31Б

2020г.

I. Описание условия задачи

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 T1, 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 T2 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 T3 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.

Техническое задание

1. Описание исходных данных

Программа ожидает:

- название файла с заявками (в аргументах командной строки)
- номер действия (подробнее о том, какой номер соответствует определённому действию, будет выведено при запуске программы).
- интервал между приходом заявок первого типа (изначально установлено значение по умолчанию, но можно изменить)
- диапазон времени обслуживания заявок первого типа (изначально установлено значение по умолчанию, но его можно изменить)

- диапазон времени обслуживания заявок второго типа (изначально установлено значение по умолчанию, но его можно изменить)

2. Описание результата программы

Результатом работы программы могут являться (в зависимости от введённого действия):

1. Результат моделирования:
   - 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 общее время моделирования, время простоя аппарата, количество вошедших в систему и вышедших из нее заявок первого типа и количество обращений заявок второго типа.
2. Статистика по времени выполнения и объёму памяти при обработке очередей, реализованных списком и динамическим массивом.

Формат вывода:

1. После обслуживания каждых 100 заявок 1-го типа

first out: <количество вышедших первых заявок>

first in : <количество вошедших первых заявок>

second count: <количество обращений второй заявки>

average time in queue: <среднее время в очереди>

current length of queue: <размер очереди сейчас>

average length of queue: <средний размер очереди>

2. в конце процесса:

time from begining: <общее время моделирования>

time of free machine: <время простоя>

first out: <количество вышедших первых заявок>

first in : <количество вошедших первых заявок>

second count: <количество обращений второй заявки>

average time in queue: <среднее время в очереди>

counted time: <расчётное время>

percent: <процент расхождения расчётного времени и общего времени моделирования >

3. Статистика выводится в виде таблицы.

3. *Описание задачи, реализуемой программой*

В программе моделируется процесс обработки заявок двух типов. Когда обработана 1000 заявок 1-ого типа, моделирование обработки завершается. Моделирование работает с обеими очередями: реализованной на основе списка и на основе массива. Сравнить эффективность очередей можно в разделе статистика.

4. Способ обращения к программе

Способ обращения к программе - консольный. Дальнейшие инструкции будут выведены после запуска.

5. Описание возможных аварийных ситуаций и ошибок пользователя

- ошибка ввода действия;
- ошибка в названии файла или его отсутствие (физическое и/или в аргументах командной строки);
- отказ операционной системы выделить запрашиваемую память;

Во всех указанных случаях программа сообщит об ошибке

II. Описание внутренних структур данных

В программе есть 2 основных структуры данных:

1. Очередь на основе списка

Описание на языке С стека выглядит таким образом:

```
typedef struct node
{
    struct node *next;
    elem_t *value;
} node_t;

typedef struct
{
    node_t *head;
    node_t *tail;
    vector_t adrs;
    size_t count;
} queue_on_list_t;
```

Данные хранятся в форме односвязного линейного списка. Структура очереди состоит из: указателя head на голову списка (место выхода), tail на хвост списка (место входа), текущего количества элементов. Дополнительно ввиду условия задачи хранится вектор adrs адресов освобождённых элементов списка. У данной структуры есть следующие особенности:

- данные в памяти располагаются в произвольном порядке;
- при добавлении элемента каждый раз запрашивается память под узел элемента;
- при удалении элемента каждый раз память под узлом освобождается;

2. Очередь на основе динамического массива

Описание на языке С стека выглядит таким образом:

```
typedef struct {
    elem_t **data;
    size_t head;
    size_t tail;
    size_t count;
    size_t max_count;
} queue_on_array_t;
```

Здесь сами данные хранятся в закольцованном массиве data, head представляет собой индекс головы (место выхода) очереди в массиве, tail – индекс хвоста (место входа) очереди в массиве, count и max_count – текущее количество элементов и максимальное соответственно. У данной структуры есть следующие особенности:

- данные в памяти хранятся последовательно;
- выделение памяти происходит один раз на максимальный размер очереди;
- при добавлении и удалении элемента лишь

Вспомогательной структурой является вектор, который реализует хранение данных. Его описание на языке С выглядит следующим образом:

```
typedef struct
{
    void *front;
    size_t size_of_element;
```

```
    size_t size;
    size_t alloc_size;
} vector_t;
```

III. Описание алгоритма

Пока количество вышедших первых заявок меньше тысячи последовательно выполняются следующие действия (каждая итерация представляет собой промежуток времени между поступлением заявок первого вида):

1. Обрабатывается столько заявок, на сколько хватает времени (если заявка первого вида, то она после обработки уходит из системы, а если второго, то выполняется вставка обратно в очередь не далее 4 позиции)
2. Поступает новая заявка первого типа

Изначально в обработку поступает заявка 2-ого типа.

Из описания алгоритма очевидно, что простоя у аппарата не будет: вторая заявка в любом случае будет в системе, а значит у аппарата всегда есть работа.

Оценка по времени

Для обработки очереди были использованы очереди, реализованные на основе списка и на основе массива. Исходя из особенностей каждой структуры, мы можем сделать вывод о том, что по времени массив окажется быстрее из-за отсутствия выделения памяти и освобождения памяти при добавлении и удалении элемента соответственно.

Оценка по памяти

Оценка по памяти во многом зависит от того, насколько большим выбирается максимум длины очереди на массиве. Если он сравним с максимальной длиной очереди, то массив окажется эффективнее по памяти, чем список, ведь массив состоит из указателей на данные, а список из указателя на данные и на следующий элемент. В обратном случае список окажется эффективнее благодаря своей гибкости.

Проверка выводов экспериментально

queue	time	memory
on linked list	853.30	4225.60
on vector	429.30	2185.60

Такой результат мы получаем при учёте фактически используемой памяти массивом (реальные затраты по памяти зависят от выбора максимума длины очереди).

Итог

Мы экспериментально подтвердили сделанные ранее оценки.

IV. Выводы по проделанной работе

В ходе работы был проведён сравнительный анализ реализаций очередей (на основе односвязного списка и массива), в результате которого установилось, что список является более гибкой по памяти СД по сравнению с массивом: затраты по памяти в списке зависят от фактической длины очереди, а в массиве память выделяется под определённый максимум длины очереди. Таким образом, при “неправильно” выбранном максимуме массив затрачивает много неиспользуемой памяти, что очень неэффективно. По фактически используемой памяти же массив оказался эффективнее в 2 раза. По времени массив оказывается более эффективным (в 2 раза), так как не происходит выделение памяти при добавлении, вставке и удалении элемента.

Ответы на вопросы

1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с одной стороны (с «хвоста»), а исключение – с другой стороны (с «головы»).

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

При хранении списком память выделяется по одному элементу (при добавлении, вставке). При хранении массивом память выделяется один раз на максимальное количество элементов в очереди.

3. Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?
4. При хранении списком память освобождается по одному элементу (при удалении). При хранении массивом память освобождается один раз после завершения работы с очередью.
5. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди элементы последовательно удаляются.

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

См. «Выводы по проделанной работе».

6. В каком случае лучше реализовать очередь посредством указателей, а в каком – массивом?

См. «Выводы по проделанной работе».

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

Недостатки:

- при реализации обычным массивом (не кольцевой структуры) при удалении элемента необходимо сдвигать весь массив
- при реализации массивом кольцевой структуры у очереди есть верхняя граница (то есть возможно переполнение)
- при реализации списком затрачивается дополнительная память на указатели и время на выделение памяти под новый элемент, освобождение его.

Достоинства:

- при реализации массивом кольцевой структуры добавление и удаление элементов очень быстрое (просто меняются указатели на начало или конец)
- при реализации списком затрачиваемая память теоретически ограничена лишь объёмом оперативной памяти

8. Что такое фрагментация памяти?

Фрагментация – процесс появления незанятых участков в памяти (как оперативной, так и виртуальной и на магнитных носителях). Вызвана наличием в каждом виде памяти деления на мелкие единицы фиксированного размера, в то время как объём информации не обязательно кратен этому делению.

9. На что необходимо обратить внимание при тестировании программы?

На переполнение очереди при реализации массивом кольцевой структуры и на процент расхождения расчётного времени и общего времени моделирования (он должен быть в пределах 2-3 процентов).

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

Выделение памяти происходит блоками — непрерывными фрагментами оперативной памяти (таким образом, каждый блок — это несколько идущих подряд байт). В какой-то момент в куче может не оказаться блока подходящего размера и, даже если свободная память достаточна для размещения объекта, операция выделения памяти окончится неудачей.