Отчёт

Дисциплина: Типы и структуры данных
Лабораторная работа №5
по теме: «Работа с очередью»

Работу выполнил:
студент группы ИУ7-32Б
Иванов Всеволод

Описание условия задачи

Реализовать операции работы с очередью, которая представлена в виде массива (статического или динамического) и в виде односвязного списка.

Оценить преимущества и недостатки каждой реализации, сравнив затрачиваемое на добавление и удаление элементов время и память.

Описание технического задания

Программа для моделирования процесса работы ОА, обрабатывающего заявки, поступающие в очереди.

Область применения – любые допустимые параметры времени

Сроки выполнения – 2 недели

Основой для разработки программы послужила потребность в реализации очереди различными способами. Выполняемой задачей была поставлена симуляция обслуживания заявок обрабатывающим автоматом. Требуется смоделировать работу автомата с двумя очередями заявок. Приоритет обработки – чередующийся. Заявки поступают и обрабатываются в указанном диапазонам времени (значение случайно варьируется в пределах диапазонов), после чего покидают систему.

Для реализации задачи использовались очереди заявок, хранящие время их поступления, в виде связанных списков и в виде массива. Для связанных списков использовалась область свободной памяти в виде стека адресов.

Программа должна:

- Содержать указание формата и диапазона данных при вводе
- Содержать указание операций, производимых программой
- Контролировать правильность ввода данных
- Указание операции, производимой программой: добавление элемента в очередь, удаление элемента из очереди
- Иметь пояснения при выводе результата
- Рассчитывать процент погрешности моделирования работы ОА
- Иметь возможность изменять времена прихода и обработки заявок
- Отображать результаты сравнения времени и памяти работы программы при использовании очереди в виде связанного списка и массива

- Выводить промежуточные состояния ОА (через каждые 100 отработанных заявок 1-го типа)
- Выводить итог работы ОА после 1000 отработанных заявок 1-го типа (количество поступивших и вышедших заявок, смоделированное время)

Этап постановки

Исходными данными являются диапазоны времени обработки и вхождения заявок. Результатом являются данные моделирования (количество поступивших и вышедших заявок, смоделированное время), а также сравнение времени добавления и удаления элемента из различных реализаций очереди.

Обращение к программе происходит выбором функций программы путём введения их кодов в консоли. Вывод результатов работы программы также производится в консоль.

Возможными ошибками являются:

- Ввод некорректных диапазонов времени
    - Одно из значений менее 0
    - Верхняя граница меньше нижней
    - Ввод символов вместо чисел
- Ввод символов в поле команды
- Ввод несуществующей команды

Особых требований для технических средств, аппаратной совместимости, программных средств не имеется.

Описание СД

Для хранения элементов связанного списка была создана структура node_t, содержащая:

- Вещественное число time типа float, хранящая время прихода соотв. заявки.
- Указатель next на следующий элемент связанного списка

```
// List element type
typedef struct cell_tag
{
    float time;
    struct cell_tag *next;
} cell_t;

typedef cell_t* list_t;
```

Сам связанный список представлен указателем на элемент списка

На данной структуре также простроен стек области свободной памяти free_mem_t

```
// Free memory cells type
typedef list_t free_mem_t;
```

Очередь на связанном списке с привязанной областью свободной памяти реализована структурой lqueue_t, содержащей:

- Указатель free на стек свободной памяти free_mem_t
- Указатели pin и pout на начало и конец списка типа list_t

```
// List queue type ( with free memory var)
typedef struct
{
    free_mem_t* free;
    list_t pin;
    list_t pout;
} lqueue_t;
```

Для хранения очереди на массиве использована структура aqueue _t, хранящая:

- Целое число size – размер выделенной памяти под очередь
- Целые числа pin и pout – индекс начала и конца очереди в массиве
- Целое число count – счётчик количества элементов
- Массив time_arr типа array_t, хранящий элементы очереди

```
typedef float* array_t;

// Array stack type
typedef struct
{
    int size;
    int count;
    int pin;
    int pout;
    array_t time_arr;
} aqueue_t;
```

Для хранения параметров работы ОА была использована структура options_t:

- Двухэлементные массивы t1, t2, t3, t4 типа float, хранящие диапазоны времён T1, T2, T3, T4
- Целое число step типа int – шаг печати ОА
- Целое число max_1 типа int – количество обрабатываемых заявок 1-го типа.

```
typedef struct
{
    float t1[2];
    float t2[2];
    float t3[2];
    float t4[2];
    int step;
    int max_1;
} options_t;
```

Описание алгоритма

Работа ОА осуществляется по следующему алгоритму. Сначала определяется время прихода первых заявок в обеих очередях и текущее время устанавливается на 0. После чего начинается повторение следующих действий:

- Если текущее время = времени прихода очередной заявки, то она добавляется в очередь и генерируется время прихода следующей заявки
- В случае, если ОА завершил обработку заявки, то оценивается текущее состояние очередей, и на основе него определяется статус ОА (в работе с 1 или 2 очередью или простаивает). После чего, если возможно, в него поступает новая заявка, удаляемая из очереди.
- Происходит подсчёт статистических данных и переход по времени до ближайшего следующего события (поступление в 1 или 2 очередь, или завершение отработки заявки)

Оценка времени работы очередей производится путём многократного замера времени работы ОА. После этого общее время делится на количество повторов операции.

Набор тестов

Негативные тесты

Ввод	Проверяемый случай	Вывод
50	Использование несуществующей команды	Ошибка: команда не существует
afa	Ввод символов	Ошибка: неправильный формат ввода
-1 2	Некорректный диапазон времени	Ошибка: некорректный диапазон времени
3 1	Некорректный диапазон времени	Ошибка: некорректный диапазон времени
-1.51	Ввод некорректного времени в очередь	Ошибка: недопустимое значение

Тепличные тесты

Ввод	Проверяемый случай	Вывод
начало программы Команда 0	Работа без изменения параметров	Переход к основному меню
начало программы Команда 1	Работа с изменением параметров	Переход к режиму изменения параметров
тестовый режим Команда: 0	Выход из тестового режима	Выход *завершение работы программы*
тестовый режим Команда: 10 1.2	Добавление элемента в очередь на связанном списке	Вывод очереди Вывод области свободной памяти
тестовый режим Команда: 11	Удаление элемента из очереди на связанном списке	Вывод удалённого элемента Вывод очереди Вывод области свободной памяти
тестовый режим Команда: 20 1.2	Добавление элемента в очередь на массиве	Вывод очередь
тестовый режим Команда: 21	Удаление элемента из очереди на массиве	Вывод удалённого элемента Вывод очереди
тестовый режим Команда: 30	Сравнение времени и памяти при моделировании с помощью очереди на списке и массиве	Время и память, затраченные на моделирование работы ОА Сравнение в процентах эффективности очередей
основное меню Команда: 0	Переход в тестовый режим	*запуск тестового режима*
основное меню Команда: 1	Моделирование ОА с использованием очереди на массиве	*вывод информации о результатах моделирования* *завершение программы*

основное меню Команда: 2	Моделирование ОА с использованием очереди на списке	*вывод информации о результатах моделирования* *завершение программы*

Граничные тесты

Ввод	Проверяемый случай	Вывод
[1, 1] [0.5, 0.5] [1.5, 1.5] [0.5, 0.5]	Работа при фиксированном времени прихода и обработки	*вывод информации о результатах моделирования*

Сравнение эффективностей

Время моделирования обработки 1 000 заявок:

```
Test of time and memory at SM work on 1000 requests from queue 1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Stack    List            Array           Advantage
Time     1.300 ms.       0.340 ms.       3.8 times
Memory     1677 b          659 b         2.5 times
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Время моделирования обработки 20 000 заявок:

```
Test of time and memory at SM work on 20000 requests from queue 1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Stack    List            Array           Advantage
Time     77.840 ms.      6.820 ms.       11.4 times
Memory     8857 b          3123 b        2.8 times
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Время моделирования обработки 100 000 заявок:

```
Test of time and memory at SM work on 100000 requests from queue 1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Stack    List            Array           Advantage
Time     776.400 ms.     33.760 ms.      23.0 times
Memory    17934 b          7207 b        2.5 times
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Отношение времени и памяти реализаций очередей на связанном списке и на массиве при разных размерностях в среднем:

Количество заявок	*Время*	*Память*
1 000	***4 раза***	***2.5 раз***
20 000	***11 раз***	***2.5 раз***
100 000	***23 раз***	***2.5 раз***

При работе с очередью также наблюдается фрагментация элементов, что главным образом обусловлено тем, что область свободной памяти построена на стеке, что нарушает порядок следования элементов при их удалении и добавлении

Работа программы

```
Input time
Queue 1) T1:    [ 1.000000, 5.000000 ]
Queue 2) T2:    [ 0.000000, 3.000000 ]

Processing time
Queue 1) T3:    [ 0.000000, 4.000000 ]
Queue 2) T4:    [ 0.000000, 1.000000 ]

Print step: 100
Max processing requests: 1000
```

Теоретическое время рассчитывается по формуле:

t(теор) = out1 * t3 + out2 * t4 + время простоя

При параметрах, заданных по условию задачи (см. рис. выше) моделирование выдаёт следующие результаты:

```
Modeling completed!

Time:            3001.623
Downtime:       4.841
Queue 1: In - 1012       Out - 1000
Queue 2: In - 1990       Out - 1981

Comparing time and predicted time (Queue 1 and 2 processing time + downtime)
Time: 3001.623   Predicted time: 2995.341       Error: 0.210 %
```

Теоретическое время t(теор) = 1000 * 2 + 1981 * 0.5 + 4.841 = 2995.341

Погрешность моделирования = (3001.623 - 2995.341) / 3001.623 = 0.21 %

Для заявок:

В данной задаче теоретически темп прихода новых заявок и темп их обработки совпадают, поэтому за 3000 е.в. должно прийти 3000/3 и 3000/1.5 заявок, а обработка в таком случае также определяется соотношением времён обработки двух очередей. Таким образом, к 1 обработанной заявке первой очереди должно приходиться 2 заявки из второй (отношение 1 к 2).

Параметр	Ожидаемое	Фактическое	Погрешность
Вход 1й очереди	1000	1012	1.2%
Вход 2й очереди	2000	1990	0.5%
Выход 1й очереди	1000	1000	0%
Выход 2й очереди	1981	1981	0.95%

Другие запуски моделирования:

```
--------------------------------------------------------------------------
Modeling completed!

Time:            3011.154
Downtime:       27.163
Queue 1: In - 1004       Out - 1000
Queue 2: In - 2028       Out - 2002

Comparing time and predicted time (Queue 1 and 2 processing time + downtime)
Time: 3011.154   Predicted time: 3028.163        Error: 0.562 %
--------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
Modeling completed!

Time:            3033.312
Downtime:       8.823
Queue 1: In - 1012       Out - 1000
Queue 2: In - 2077       Out - 2054

Comparing time and predicted time (Queue 1 and 2 processing time + downtime)
Time: 3033.312   Predicted time: 3035.823        Error: 0.083 %
--------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
Modeling completed!

Time:            3006.716
Downtime:       72.607
Queue 1: In - 1003       Out - 1000
Queue 2: In - 1972       Out - 1967

Comparing time and predicted time (Queue 1 and 2 processing time + downtime)
Time: 3006.716   Predicted time: 3056.107        Error: 1.616 %
--------------------------------------------------------------------------
```

Ответы на вопросы

1. Что такое очередь?

Очередь - структура данных, являющаяся последовательным списком, в котором включение элементов идёт с одной стороны, а исключение с другой (с «хвоста» и «головы»). При этом одновременно доступ возможен только к «голове» очереди. Очередь функционирует по правилу FIFO.

2. Каким образом и сколько памяти выделяется под хранение очереди при различной его реализации?

При реализации очереди в виде массива, для его хранения отводится непрерывная область памяти. Память в размере **sizeof(data) * len** байт. Длинна массива обычно отводится с запасом и расширяется при его достижении.

При реализации очереди в виде связанного списка, для хранения каждого его элемента отводится отдельная область памяти. Элементы в ней связаны при помощи указателей на следующий элемент. Таким образом, под очередь отводится **(sizeof(data) + sizeof(pointer)) * len** байт. Выделенная память при этом может быть фрагментированной.

3. Каким образом освобождается память при удалении элемента очереди при различной реализации очереди?

При удалении элемента из очереди на связанном списке память данного элемента может освобождаться сразу, или помещаться в область свободной памяти до востребованности.

При удалении элемента из очереди на массиве в нём изменяется индекс «головы» очереди. Занимаемая память может уменьшаться в случае, когда этот индекс достигает установленной границы относительно размера занимаемой памяти (например при < 1/3)

4. Что происходит с элементами очереди при его просмотре?

Просмотр элемента очереди возможен только в случае, если он находится на «голове». В таком случае его просмотр возможен путём удаления из очереди

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

В случае, когда требуется эффективность по памяти лучше реализация очереди массивом, так как время добавления и удаления будут меньше, чем в

списке. Также такая реализация может дать выигрыш в занимаемой памяти в некоторых случаях.

В случае, когда требуется использовать больше памяти лучше использовать реализацию на связанном списке, так как ему не требуется непрерывность памяти, поэтому его размер ограничен только размером свободной оперативной памяти.

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

В случае, когда размер очереди постоянно сильно варьируется, лучше использовать реализацию на связанном списке, так как для массива потребуется делать реструктуризацию при каждом превыделении памяти под него. В других случаях лучше использовать массив, так как он эффективнее по времени и занимаемой памяти

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

Достоинства очереди на массиве заключаются в быстроте добавления и удаления элементов, так как при этом не происходит выделения памяти. Недостатком является необходимость в перевыделении памяти и последующей реструктуризации очереди.

Недостатком очереди на связанном списке является большее время добавления и удаления элемента. В случае, если существует область свободной памяти, время обработки можно уменьшить, но оно всё равно будет больше времени при реализации на массиве, так как производится больше операций. Достоинством является лёгкая расширяемость и возможность занимать память без запаса.

8. Что такое фрагментация памяти?

Фрагментация памяти — это выделение непоследовательных адресов памяти. Таким образом при фрагментации элементы очереди могут оказываться в памяти не рядом, а на определённом удалении друг от друга, что может повлиять на время их последовательной обработки

9. На что необходимо обратить внимание при тестировании программы?

Следует обратить внимание на то, что при активной работе с очередью на списке, будет возникать фрагментация памяти, так как область свободной памяти реализована на стеке, а следовательно, имеет другой порядок входа-выхода элементов.

При тестировании ОА следует использовать различные пограничные ситуации, чтобы проверить ОА на устойчивость работоспособности и оценки времени обработки заявок.

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

При динамических запросах на выделение памяти, файловый менеджер находит свободную область памяти требуемого размера и помечает её как занятую. При этом возвращается адрес выделенной памяти. При освобождении менеджер помечает данную память как свободную

Выводы по проделанной работе

Из проведённого анализа следует вывод, что даже при использовании уже зарезервированной памяти из области свободной памяти очередь на связанном списке проигрывает очереди на массиве по времени. Выигрыш по занимаемой памяти у очереди на массиве примерно равен 2.5 разам. Однако, очередь на массиве имеет недостаток в том, что выделяемая под него память должна быть непрерывна, а также может требовать частичной реструктуризации при изменении размера. Связанный список в свою очередь может быть фрагментированным. Таким образом он может хранить большее количество элементов и не требовать реструктуризации в принципе.