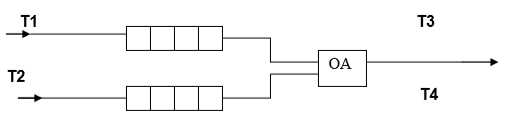
Отчет по 5 лабораторной работе по предмету «Типы и Структуры Данных»

Выполнил студент группы ИУ7-32Б

Чалый Андрей

Вариант 25

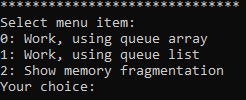
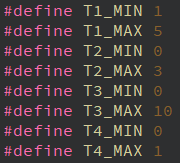
Система массового обслуживания состоит из обслуживающего аппарата (ОА) и двух очередей заявок двух типов.



Заявки 1-го и 2-го типов поступают в "хвосты" своих очередей по случайному закону с интервалами времени Т1 и 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 – вещественного типа) В начале процесса в системе заявок нет. Заявка любого типа может войти в ОА, если:  а) она вошла в пустую систему;  б) перед ней обслуживалась заявка ее же типа;  в) перед ней из ОА вышла заявка другого типа, оставив за собой пустую очередь (система с чередующимся приоритетом). Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок информацию о текущей и средней длине каждой очереди, а в конце процесса - общее время моделирования и количество вошедших в систему и вышедших из нее заявок обоих типов. По требованию пользователя выдать на экран адреса элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

**Входные данные**

Время и вероятность, выбор реализации списком или массивом.



**Выходные данные**

После обслуживания каждых 100 заявок 1-ого типа выводится информация о текущей и средней длине каждой очереди, а в конце процесса - общее время моделирования и количество вошедших в систему и вышедших из нее заявок обоих типов, а также время моделирования.

**Аварийные ситуации**

Переполнение при реализации массивом

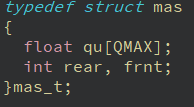
Результат работы программы:

“QUEU IS FULL”

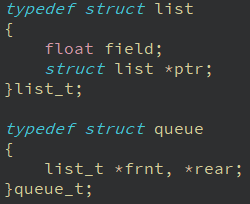
(Программа останавливает обработку заявок и не дает итогового результата при переполнении массива).

**Используемые структуры**

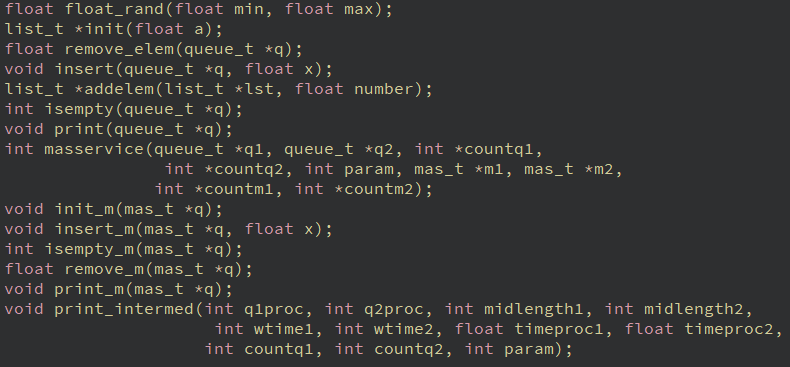
Очередь, реализованная на статическом массиве. Массив не является кольцевым.



Очередь, реализованная на списке.

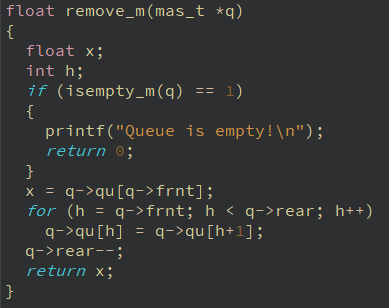
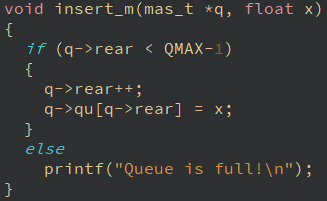


**Используемые функции**

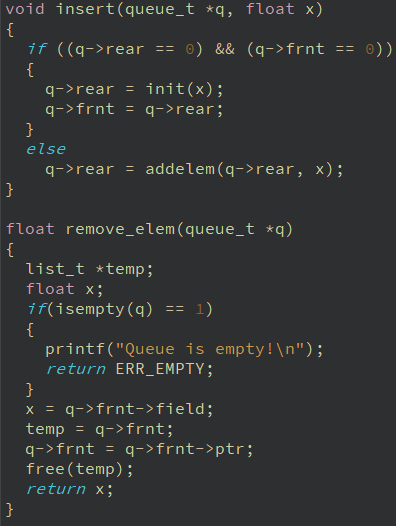


**Алгоритмы**

Для очереди, реализованной на массиве:



Для очереди, реализованной на списке:



**Теоретическое время моделирования**

Среднее время прихода заявок 1-го типа:

(1 + 5) / 2 = 3 е.в;

Среднее время обработки этих заявок

(0 + 4) / 2 = 2 е.в.

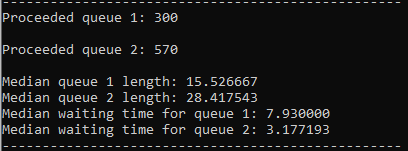
1000 заявок приходят за 3 \* 1000 = 3000 е.в .

1000 заявок обрабатываются за 2 \*1000=2000 е.в.

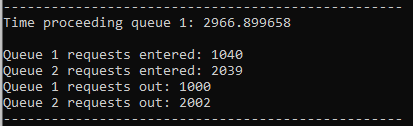
При данных параметрах время прихода заявок больше, чем время их обработки, следовательно, как и указано в методических указаниях, на время моделирования влияет только время прихода заявок. Таким образом, получаем, что среднее время моделирования 3000 е.в.

За это время должно придти не менее (3000/1.5) 2000 заявок 2-го типа. На их обработку требуется (2000\*0.5) 1000 единиц времени. Именно столько простиавает обслуживающий аппарат при обработке только заявок 1-го типа.

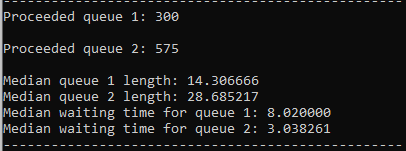
По текущему времени на массиве (пример при 300):



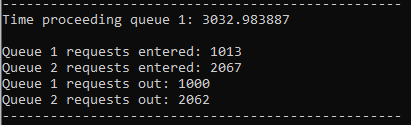
Итоговый результат на массиве:



По текущему времени на списке (пример при 300):

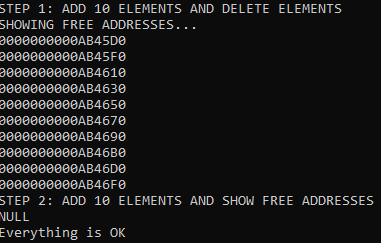


Итоговый результат на списке:



**Сравнение эффективности**

Погрешность в целом примерно равна в обеих реализациях, колеблется от 0.1% до 3%. По времени список тоже менее эффективен, так как он должен освобождать и выделять память каждый раз, когда добавляется, или удаляется элемент, а это затрачивает время. Массив работает не сильно быстро так как он не кольцевой. При работе программы не возникает фрагментация.



При представлении очереди в виде списка используется большее количество памяти для хранения указателей (в представленном примере массив на 50% более эффективен по памяти).

**При данных стандартных данных T1(1..5) T3(0..4)**

Среднее время прихода заявок:

(1 + 5) / 2 = 3 е.в;

Среднее время обработки заявок

(0 + 4) / 2 = 2 е.в.

1000 заявок приходят за 3 \* 1000 = 3000 е.в .

1000 заявок обрабатываются за 2 \* 1000=2000 е.в.

|  |  |  |  |
| --- | --- | --- | --- |
| № (массив) | Число вошедших заявок | Число вышедших заявок | Время моделирования (в единицах времени) |
| 1 | 1005 | 1000 | 3029.137451 |
| 2 | 1001 | 1000 | 2971.223145 |
| 3 | 1003 | 1000 | 3014.571289 |
| 4 | 1009 | 1000 | 2973.293457 |
| 5 | 1006 | 1000 | 3084.639648 |
| Среднее | 1004 | 1000 | 3014.2 |

|  |  |  |  |
| --- | --- | --- | --- |
| № (список) | Число вошедших заявок | Число вышедших заявок | Время моделирования (в единицах времени) |
| 1 | 1002 | 1000 | 3008.989014 |
| 2 | 1003 | 1000 | 3029.974854 |
| 3 | 1017 | 1000 | 3030.339844 |
| 4 | 1001 | 1000 | 2998.523926 |
| 5 | 1000 | 1000 | 3037.893066 |
| Среднее | 1005 | 1000 | 3020.4 |

**При изменении T1(1..5) T2(0..10)**

Среднее время прихода заявок:

(1 + 5) / 2 = 3 е.в;

Среднее время обработки заявок

(0 + 10) / 2 = 5 е.в.

1000 заявок приходят за 3 \* 1000 = 3000 е.в .

1000 заявок обрабатываются за 5 \* 1000=5000 е.в.

|  |  |  |  |
| --- | --- | --- | --- |
| № (массив) | Число вошедших заявок | Число вышедших заявок | Время моделирования (в единицах времени) |
| 1 | 1696 | 1000 | 4985.513672 |
| 2 | 1634 | 1000 | 4896.073730 |
| 3 | 1654 | 1000 | 4903.774902 |
| 4 | 1671 | 1000 | 5070.074219 |
| 5 | 1622 | 1000 | 4914.458496 |
| Среднее | 1655 | 1000 | 4953.6 |

|  |  |  |  |
| --- | --- | --- | --- |
| № (список) | Число вошедших заявок | Число вышедших заявок | Время моделирования (в единицах времени) |
| 1 | 1650 | 1000 | 4955.969727 |
| 2 | 1685 | 1000 | 5003.825195 |
| 3 | 1686 | 1000 | 5005.237305 |
| 4 | 1668 | 1000 | 4952.419434 |
| 5 | 1696 | 1000 | 5071.703125 |
| Среднее | 1005 | 1000 | 4997.2 |

**Ответы на вопросы**

1. Что такое очередь?

Очередь – это последовательный список переменной длины, включение элементов в который идет с одной стороны, а исключение – с другой стороны.

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

В списке память под элемент очереди выделяется непосредственно в процессе его добавления. Объем памяти, который занимает очередь, изменяется в процессе выполнения программы и напрямую зависит от количества элементов в очереди в каждый момент времени. При реализации очереди массивом выделяется последовательная область памяти константного размера. Выделение памяти происходит в начале работы программы. При необходимости память перевыделяется.

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

В массиве освобождение памяти происходит в конце работы программы (или при удалении очереди). При удалении элемента из очереди происходит только смещение указателя. В списке при удалении элемента из очереди происходит освобождение памяти, которая была выделена под этот элемент.

4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди элемент головной из нее удаляется.

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

Если необходимо избежать фрагментации памяти, то лучше использовать очередь на массиве. Однако такой способ лучше использовать в том случае, если заранее известно количество элементов в очереди. Иначе лучше реализовывать список.

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

Очередь лучше реализовывать с помощью указателей, если новые элементы в среднем появляются реже, чем происходит полное очищение очереди – в общем случае фрагментация не возникает.

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

При реализации очереди массивом не возникает фрагментации памяти, однако может произойти переполнение очереди, а также затрачивается дополнительное время на сдвиг элементов. При реализации очереди списком затрачивается большее количество времени при добавлении нового элемента, для хранения указателей требуется дополнительная память.

8. Что такое фрагментация памяти?

При последовательных запросах на выделение и освобождении памяти под элемент не всегда выделяется память, которая была только что освобождена.

9. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо обратить внимание на переполнение очереди, фрагментацию памяти при реализации очереди списком.

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

Программа дает запрос ОС на выделение блока памяти необходимого размера. ОС находит подходящий блок, записывает его адрес и размер в таблицу адресов, а затем возвращает данный адрес в программу. При запросе на освобождение указанного блока программы, ОС убирает его из таблицы адресов, однако указатель на этот блок может остаться в программе. Попытка считать данные из этого блока может привести к ошибке программы или неверному результату, поскольку они могут быть уже изменены.