Лабораторная работа №5 по дисциплине

“Типы и структуры данных”

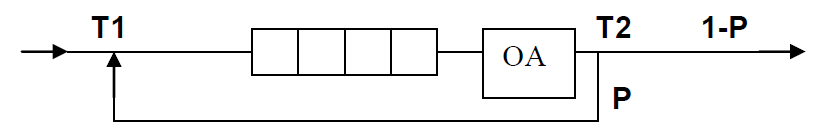
Сорокин Антон ИУ7-32Б

Номер по списку – 24

Вариант 5

***Условие задачи***

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 с вероятностью Р=0.8 вновь поступает в "хвост" очереди, совершая новый цикл обслуживания, а с вероятностью 1-Р покидает систему. (Все времена – вещественного типа). В начале процесса в системе заявок нет.

Смоделировать процесс обслуживания до ухода из системы первых 1000 заявок. Выдавать после обслуживания каждых 100 заявок информацию о текущей и средней длине очереди. В конце процесса выдать общее время моделирования и количество вошедших в систему и вышедших из нее заявок, среднее время пребывания заявки в очереди, время простоя аппарата, количество срабатываний ОА. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

***Техническое задание***

*Исходные данные*

На вход программе могут подаваться интервалы времени T1 и T2.

*Результат*

Информация о текущем состоянии очереди после каждых 100 обработанных заявок, информация о времени моделирования, количестве срабатываний ОА, количестве вошедших и вышедших заявок, времени простоя аппарата, времени пребывания заявки в очереди.

*Описание задачи*

Программа моделирует процесс обслуживания заявок в системе очередей, описанной в условии. Необходимо вычислить временную информацию о работе системы. Очереди реализуются в виде статического массива и в виде списка.

*Аварийное завершение работы программы*

Программа завершается аварийно только при переполнении очереди или ошибке выделения памяти.

***Обращение к программе***

Исполняемый файл app.exe создается путем автоматической сборки проекта с помощью файла makefile. Для выполнения работы следует запустить данный исполняемый файл без указания аргументов.

***Алгоритм***

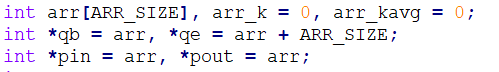
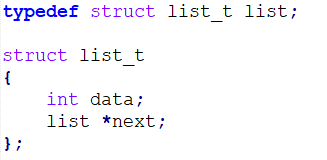
* Добавление элемента в очередь-массив:
  + Если указатели на «вход» и на «выход» равны и количество элементов не равно 0, то вернуть ошибку о переполнении.
  + По указателю на «вход» записать новый элемент.
  + Сдвинуть указатель вправо. Если указатель ссылается на верхнюю границу выделенной памяти, присвоить указатель нижней границе.
  + Увеличить счётчик количества элементов на 1.
* Добавление элемента в очередь-список:
  + Выделить память под новый элемент, в случае неудачи вернуть ошибку.
  + Если очередь пустая, то «голову» очереди присвоить новому элементу.
  + Иначе, по указателю на следующий элемент «хвоста» очереди записать адрес нового элемента
* Удаление элемента из очереди-массива:
  + Если количество элементов равно 0, вернуть ошибку о пустоте.
  + Запомнить элемент, расположенный по указателю на «выход».
  + Сдвинуть указатель вправо. Если указатель ссылается на верхнюю границу выделенной памяти, присвоить указатель нижней границе.
  + Уменьшить счётчик количества элементов на 1.
  + Вернуть сохранённое значение.
* Удаление символа из стека-списка:
  + Если указатель «головы» нулевой, то вернуть ошибку о пустоте.
  + Если указатели на «голову» и на «хвост» совпадают, то сделать указатель на «хвост» пустым.
  + Запомнить элемент, расположенный по указателю на «голову».
  + Запомнить ключ элемента.
  + Указатель на «голову» переставить на следующий элемент.
  + Освободить память из-под сохранённого элемента.
  + Вернуть сохранённый ключ.

***Тестовые данные***

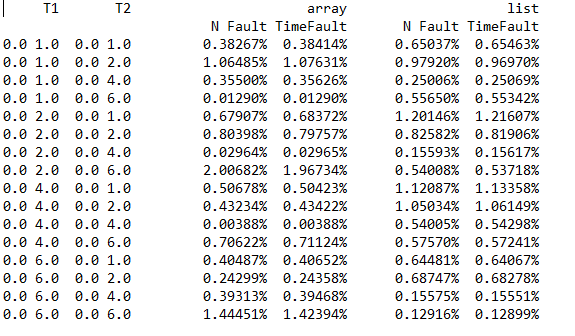
|  |  |  |
| --- | --- | --- |
| T1 | T2 | Время моделирования |
| 0.0 1.0 | 0.0 1.0 | 2500 |
| 0.0 2.0 | 0.0 1.0 | 5000 |
| 0.0 1.0 | 0.0 2.0 | 5000 |
| 0.0 2.0 | 0.0 2.0 | 5000 |
| 0.0 2.0 | 0.0 6.0 | 15000 |
| 0.0 6.0 | 0.0 2.0 | 15000 |
| 0.0 0.0 | 0.0 0.0 | 0 |
| 1.0 1.0 | 1.0 1.0 | 5000 |
| 1.0 1.0 | 2.0 2.0 | 10000 |
| 2.0 2.0 | 1.0 1.0 | 10000 |

***Внутренние структуры данных***

Для хранения очереди используется либо односвязный список, либо статический массив. Длина очереди хранится в отдельной целочисленной переменной. Также вводятся отедльные указатели на «вход» и на «выход».

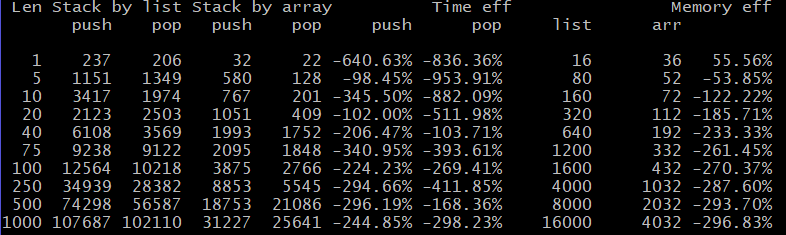
***Оценка качества моделирования***



Исходя из полученных данных тестирования, можно утверждать, что погрешность измерения времени моделирования и количества вхождений в систему не превышает 2-3% при различных интервалах системы.

Это обуславливается случайный характером определения времени работы ОА и переменных интервалов входа заявок в систему в процессе моделирования.

***Оценка эффективности реализаций***



Исходя из полученных данных, можно сделать вывод о том, что для реализации очереди эффективнее использовать статический массив. По времени список проигрывает, поскольку тратится дополнительное время на создание или удаления элемента. В среднем работа с массивом как со стеком быстрее 5-6 раз. По памяти массив использовать эффективнее, поскольку при реализации списка на каждый элемент дополнительно уходит память на указатель на следующий элемент, в то время как при реализации массива дополнительно память уходит только на 4 указателя (границы памяти и указатели на «вход» и на «выход»).

***Вывод***

При реализации очереди эффективнее использовать статический массив и по времени, и по памяти. Однако, если неизвестно, каким может быть максимально допустимое количество элементов, или максимальной границы нет как таковой для решения задачи, то лучше использовать список, т. к. при выделении памяти даже под динамический массив необходимо перезаписать полностью весь массив.

***Ответы на вопросы***

**1. Что такое очередь?**

Очередь – это структура данных, представляющая из себя последовательный список переменной длины. Включение и исключение элементов производится только с одного конца: включение в «хвост», исключение из «головы».

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

При реализации очереди массивом выделяется блок памяти размером, соответствующим максимально допустимому количеству элементов.

При реализации очереди списком под каждый новый элемент выделяется память из кучи. Память выделяется на два поля: на значение (ключ) элемента и на указатель на следующий элемент.

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

При реализации очереди массивом элемент «головы» считывается и указатель на «выход» переходит на следующий элемент в блоке памяти. Впоследствии считанный элемент может быть затёрт новыми элементами.

При реализации очереди списком элемент «головы» списка считывается, указатель на «голову» очереди переходит на следующий элемент по соответствующему указателю, а считанный элемент удаляется.

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

Элементы последовательно извлекаются из «головы» очереди и после просмотра возвращаются последовательно в «хвост» очереди.

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

При реализации очереди эффективнее использовать статический массив и по времени, и по памяти. Однако, если неизвестно, каким может быть максимально допустимое количество элементов, или максимальной границы нет как таковой для решения задачи, то лучше использовать список, т. к. при выделении памяти даже под динамический массив необходимо перезаписать полностью весь массив. При этом может возникнуть фрагментация памяти, поэтому способ реализации в основном зависит от того, что больше ограничивает при разработке программы: время или память.

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

Реализация с помощью указателей применима, если возможное возникновение фрагментации памяти строго контролируется. Реализовать посредством указателей эффективнее, если полное освобождение памяти преобладает на выделением памяти под новый элемент.

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

При реализации очереди в виде массива возникает проблема переполнения массива, что ограничивает проведение операций на больших очередях. При кольцевой структуре возникает проблема определения переполнения или пустоты очереди. Однако, при реализации массивом проще и быстрее обращаться к элементам очереди.

При реализации очереди списком нужно контролировать выделение и освобождение памяти. При этом достаточно просто определить, когда очередь пуста.

**8. Что такое фрагментация памяти?**

Фрагментация – это чередование участков памяти при последовательных запросах на выделение и освобождение памяти. Занятые участки чередуются со свободными, однако свободные могут быть недостаточно большими по размеру для того, чтобы сохранить в них необходимые данные.

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

Необходимо следить за правильностью перехода указателей на «вход» и на «выход» при реализации массивом, чтобы не перезаписать элемент или не выйти за пределы выделенной области памяти.

При реализации очереди списком необходимо следить за освобождением памяти при удалении элемента из очереди. При частом добавлении элементов в очередь может возникнуть фрагментация памяти, что влечёт её неэффективное использование.

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

При запросе на выделение блока памяти определённого 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 или записать данные по этому указателю приводит к неопределённому поведению, что влечёт ошибки.