****

|  |  |
| --- | --- |
|  | **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»

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 ПО ТиСД**

## “Работа с очередью”

Группа ИУ7-34Б

Название предприятия **НУК ИУ МГТУ имени Н. Э. Баумана**

|  |  |  |
| --- | --- | --- |
| Студент |  | Гареев Георгий Антонович |
| Преподаватель |  |  |

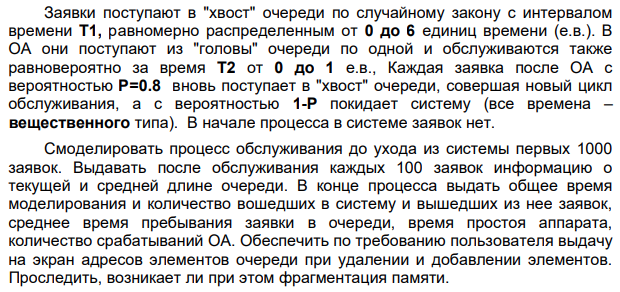
## Описание условия задачи

Цель работы

Отработка навыков работы с типом данных «очередь», представленным в виде одномерного массива и односвязного линейного списка. Сравнительный анализ реализации алгоритмов включения и исключения элементов из очереди при использовании двух указанных структур данных. Оценка эффективности программы (при различной реализации) по времени и по используемому объему памяти.

Техническое задание





Исходные данные

Интервалы времени T1 и T2 (по умолчанию взяты из условия).

Входные данные

Номер команды меню, далее значения в зависимости от функции.

Выходные данные

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, реализуемой программой

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.

Способ обращения в программе

Пользователю выдается меню, в котором он должен выбрать номер функции 0-3:

* 0 – выход из программы
* 1 – вывод работы очереди для метода статического массива.
* 2 – вывод работы очереди для метода связных списков.
* 3 – оценка качества выполнения моделирования.

Описание возможных аварийных ситуаций и ошибок пользователя

1. Память динамически не выделилась.
2. Переполнение очереди.
3. Неправильный ввод диапазона значений, однако в таком случае берутся значения по умолчанию.

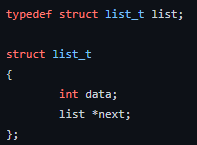
Описание внутренних СД

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

Для статического массива:



Для односвязного списка:



Для одного элемента



## Описание алгоритма

* 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 элемента из очереди-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.

## Тесты

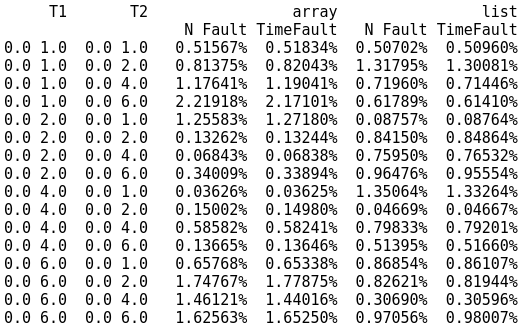
Позитивные

1. Периоды 0-1 и 0-1 (~2500 е.в.)
2. Периоды 0-0 0-0 (0 тиков)
3. Периоды 2-2 1-1 (~10000 е.в.)

Негативные

1. Таких нет

## Оценка качества моделирования

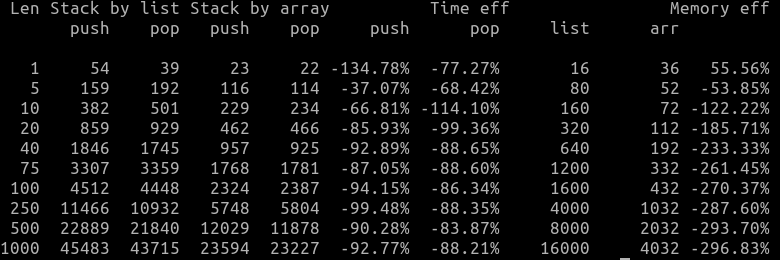


Где N Fault – погрешность заявок, Time Fault – погрешность времени работы.

Как можно заметить, погрешности не превышают предполагаемого ее значения в 2-3%.

## Оценка эффективности

Далее показания времени работы исчисляются в тиках процессора, а память в байтах.



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»).

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.

## Вывод

Как видно из результатов метод хранения в виде статического массива быстрее в среднем в два раза для всех значений длины очереди. Результаты таковы, потому что для односвязного списка в процессе происходит освобождение и выделение памяти под элементы очереди, что значительно прибавляет времени выполнения, в массиве же указатель просто меняет свое значение, что почти не занимает времени.

Поскольку для хранения в виде массива был выбран статический метод, то 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 контрольные 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 ограничивается при разработке программы: время или память.

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

Реализовать посредством указателей эффективнее, если полное освобождение памяти преобладает над 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.