Отчет по лабораторной работе № 5 по дисциплине

“Типы и структуры данных”

Сушина А.Д. ИУ7-31б

Работа № 5

Обработка очередей

Цель работы: приобрести навыки работы с типом данных «очередь»,

представленным в виде одномерного массива и односвязного линейного списка,

провести сравнительный анализ реализации алгоритмов включения и

исключения элементов из очереди при использовании указанных структур

данных, оценить эффективности программы по времени и по используемому

объему памяти.

# Описание условия задачи

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.

# Описание ТЗ

### Общая концепция системы

Программа выполняет операции над очередью, реализованной двумя способами. Очередь генерируется компьютером. Моделируется ситуация, описанная в условии задачи. Пользователь может выбирать интервалы времени и способ реализации.

### Требования к функциональным характеристикам

Программа должна выполнять следующие функции:

* возможность изменения времен прихода и обработки заявок с экрана;
* наличие пояснений при выводе результата;
* вывод процента погрешности работы программы.
* Добавление элемента в очередь для двух реализаций
* удаление элемента из очереди для двух реализаций
* моделирование ситуации
* вывод времени работы программ
* вывод адресов памяти, задействованных при выполнении
* сравнение двух реализаций

На **вход** программа получает действие (номер пункта меню). При необходимости элементы типа int, для ограничения временных промежутков.

**Выход** должен быть представлен в виде набора данных. Также программа должна вывести данные о времени выполнения алгоритма.

Программа должна выдавать корректный ответ при вводе любых данных. Если произошла ошибка ввода, программа должна сообщить об этом.

## Аварийные ситуации

1. Ввод несуществующего пункта меню

Программа выведет сообщение “Input error”.

1. Ввод значений другого типа

Программа выведет сообщение “Input error”.

1. Переполнение очереди

программа выведет FULL ORDER

1. Невозможность завершить моделирование

программа выведет Can’t end modulation

## Способ обращения к программе

Программа представляет собой файл app.exe. Запускается в консоли. Для запуска достаточно команды ./app.exe. Если файл отсутствует можно собрать его с помощью утилиты make.

# Описание структур данных

struct j - структура для хранения элемента очереди

{

double i;

struct j \*next;

};

double mas[N] = {-1}; - массив с очередью 1

double mas1[N] = {-1}; - массив с очередью 2

double \*pin = mas, \*pout = mas; - начало и конец очереди 1

double \*pin1 = mas1, \*pout1 = mas1; - начало и конец очереди 2

struct j \*empty[2\*N] = {NULL}; - массив пустых адресов

struct j \*full[2\*N] = {NULL}; - массив полных адресов

struct j \*pin = NULL, \*pout = NULL; - начало и конец 1 очереди

struct j \*empty1[2\*N] = {NULL}; - массив пустых

struct j \*full1[2\*N] = {NULL}; - массив полных

struct j \*pin1 = NULL, \*pout1 = NULL; - начало и конец 2 очереди

long long int begin\_time, end\_time; - время в тиках

double t1 = 0, t2 = 0, toa = 0, tmin = 0, tstop = 0, t\_all = 0;

* время добавления эл1, эл2, время обработки, минимальное время,время простоя, общее время выполнения

int type = 1; - тип элемента в ОА

int count\_in1 = 0, count\_out1 = 0, count\_in2 = 0, count\_out2 = 0;

* кол-во эл1 на вход, на выход, кол-во эл2 на вход, на выход

int count1 = 0, count2 = 0, count\_sr1 = 0, count\_sr2 = 0, iter;

* колво эл-тов в 1 очереди, кол во элементов во 2 очереди, среднее первой очереди, во второй

double time\_in1 = 0, time\_in2 = 0, t1\_sr = 0, t2\_sr = 0, t3\_sr = 0, t4\_sr = 0; - средние генерирующиеся времена

# Описание алгоритма

Моделирование:

while( count вышедших 1 < 1000)

{

….Если tприхода1 = 0

Найти новое t1

Если t2 = 0

….Найти новое t2

Если tобработки = 0

...Если предыдущая заявка была первого типа или нет больше заявок 2ого типа

…...удалить заявку из первой очереди  
…...Найти время обработки

...Иначе если есть заявка во второй очереди  
…...удалить заявку из второй очереди   
…...найти время обработки

Найти минимальное t

Если tmin = t1

… добавить элемент 1 типа

Если tmin = t2

… добавить элемент 2 типа

Вычесть тмин из всех времен

}

* Пока не выход

1. Run with massiv

Выполнить моделирование с помощью массива

2. Run with spisok

Выполнить моделирование с помощью списка

3. Print parameters

печать параметров

4. Change parameters

ввод новых параметров

5. Return original parameters

Вернуть исходные параметры

6. Run with adresses

Выполнить моделирование  
Напечатать свободные адреса памяти.

7. time difference

Выполнить моделирование для списка

Выполнить моделирование для массива

Вывести данные о затраченном времени и памяти

Сравнить затраченное время и память

8. exit

# Тесты

1.Несуществующий пункт меню

Вход: 10

Вывод: Input error

2. Некорректный ввод t (t < 0)

Вход:

n: -1

Вывод: Input error

3. Переполнение массива

Вход: t1 0 1

t2 0 3

t3 0 10

t4 0 10

Вывод: FULL MASSIV

4. Невозможно завершить выполнение

Вход: t1 0 1

t2 0 1

t3 0 5

t4 0 5

Вывод: can’t stop modulation

Теоретически рассчитываем время моделирования. Время обработки заявки 1 типа

лежит в интервале от 0 до 4 е.в., значит среднее значение времени обработки одной

заявки 2 е.в. Значит, общее время обработки **1000** заявок будет: 1000\*2 = **2000 е.в**.

Для прихода 1000 заявок, если каждая из них приходит в среднем за 3 е.в., потребуется **3000 е.в.**

Следовательно, время моделирования будет определяться временем прихода заявок, т.е. оно должно быть равно **3000 е.в.**

В то же время, за 3000 е.в. успеет прийти **2000** заявок второго типа, если их среднее время прихода равно 1.5 е.в.

В свободное от обработки заявок 1ого типа время, ОА будет обрабатывать заявки 2ого типа. На обработку 2000 заявок 2 типа, если среднее время обработки равно 0.5 е.в. потребуется **1000 е.в.**

Время простоя будет равно разнице между временем обработки заявок и временем их обслуживания: 5000 – 4000 = 1000 е.в. Но, с учетом того, что это время будет использовано для обработки заявок 2 типа, время простоя = 1000 - 1000 = **0 е.в.**

На практике получаем следующие результаты:

All time of modulation = 3072.661122

Wait time = -15.877438

in 1 type = 1027

in 2 type = 2067

out 1 type = 1000

out 2 type = 2039

time in ticks(spisok) = 50210826

memory is 49504

procent is 3.350000

# Оценка эффективности

|  |  |  |
| --- | --- | --- |
| Выполнение модуляции  Массив  t, такты | Выполнение модуляции  Список  t, такты | % |
| 984892 | 48229875 | 97 |
| Память  Массив  t, такты | Память  Список  t, такты | % |
| 8000 | 48000 | 60% |
| Память на 1 эл очереди  Массив | Память на 1 эл очереди  Список | % |
| 8 | 16 | 50% |

# Выводы по проделанной работе

Была реализована программа, моделирующая ситуацию с 2мя очередями с чередующимся приоритетом.

Были реализованы функции добавления и удаления элементов для двух способов реализации очередей: списком и массивом.

Реализация Списком проигрывает по времени реализации массивом на 97%. По памяти на ~60%. Таким образом, реализация списком проигрывает реализации массивом. Но с массивом сложнее отслеживать конец очереди и необходимо задуматься о возможности переполнения очереди.

# 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»). Принцип работы очереди: первым пришел – первым вышел, т.

е. First In – First Out (FIFO).

2. Каким образом, и какой объем памяти выделяется под хранение очереди при

различной ее реализации?

При реализации очереди массивом, память выделяется только под элементы массива.

Она выделяется единовременно под весь массив фиксированной длины.

При реализации списком память выделяется под элемент и его адрес. Она выделяется динамически, каждый раз при добавлении элемента.

3. Каким образом освобождается память при удалении элемента из очереди при

ее различной реализации?

При реализации массива память не освобождается до конца работы с очередью.

При реализации списком память освобождается при каждом удалении элемента из очереди.

4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди ее элементы удаляются.

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

Эффективнее реализовывать очередь массивом. Но, если количество заявок, которые необходимо обработать много меньше длины массива, то эффективнее реализовывать очередь списком.

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

каком – массивом?

Очередь списком лучше реализовывать в случае, если не важно время работы или невозможно сказать какое количество памяти понадобится для хранения очереди. Массивом лучше реализовывать в том случае, если важно время обработки.

7. Каковы достоинства и недостатки различных реализаций очереди в

зависимости от выполняемых над ней операций?

Массив:

* Плюсы:
  + высокая скорость работы (при реализации циклического массива)
  + хранится только сам элемент
* Минусы:
  + необходимость сдвига данных при каждом удалении элемента(если массив не циклический)
  + необходимость отслеживать переполнение массива

Список:

* плюсы:
  + память ограничена только памятью компьютера
* минусы:
  + скорость работы существенно ниже из-за динамического выделения памяти
  + хранится не только сам элемент, но и указатель на следующий элемент списка
  + необходимость отслеживания правильности работы с памятью

8. Что такое фрагментация памяти?

Фрагментация памяти - это ситуация, когда каждый блок памяти выделяется в новом месте, а не единым большим блоком.

9. На что необходимо обратить внимание при тестировании программы?

При тестировании программы необходимо:  
o проверить правильность работы программы при различном заполнении  
очередей, т.е., когда время моделирования определяется временем обработки  
заявок и когда определяется временем прихода заявок;  
o отследить переполнение очереди, если очередь в программе ограничена.  
  
При реализации очереди списком необходимо тщательно следить за  
освобождением памяти при удалении элемента из очереди. При этом по запросу  
пользователя должны выдаваться на экран адреса памяти, содержащие элементы  
очереди при добавлении или удалении элемента очереди. Следует проверять,  
происходит ли при этом фрагментация памяти (выделение непоследовательных  
адресов память).

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

Память выделяется с помощью функции Malloc. Эта функция выбирает доступный участок памяти нужного размера и помещает туда элемент.

Память освобождается функцией free.