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 ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 «ОБРАБОТКА ОЧЕРЕДЕЙ»

Студент Тузов Даниил Александрович

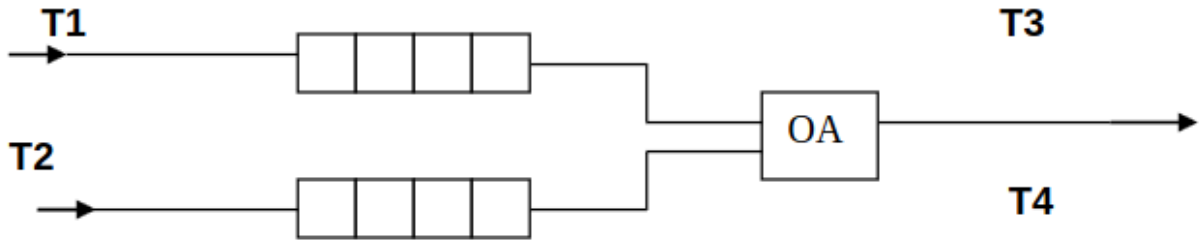
Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна
Силантьева Александра Васильевна

2023 г.

Оглавление

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 T1 и T2, 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 T3 и T4, 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 ТЕХНИЧЕСКОГО ЗАДАНИЯ

Входные данные:
Пользователю предлагается ввести число, отвечающее за действие. При выборе редактирования параметров моделирования, предлагается ввести эти параметры

Выходные данные:

На выходе получаем результат моделирования: общее время, информацию по количеству обработанных и поступивших заявок для каждой очереди и сравнение теоретического времени моделирования с общим. Так же получаем время добавления и удаления элемента для очереди на списке и на массиве.

Описание программы:
Программа имитирует работу очереди двумя способами: с помощью массива, с помощью списка

Способ обращения к программе:
Взаимодействие через консоль.

Аварийные ситуации:

1. Ошибка памяти
   Код ошибки — 2
2. Ошибка: удаление из пустой очереди
   Код ошибки — 1
3. Ошибка: добавление в заполненную очередь
   Код ошибки — 3
4. Ошибка ввода
   Код ошибки — 4

ОПИСАНИЕ СТРУКТУР ДАННЫХ

Описание узла списка
item – значение, которое лежит в узле
next – указатель на следующий узел

```
typedef struct node
{
double item;
struct node *next;
} node_t;
```

Описание очереди на списке
head – указатель на начало очереди
tail – указатель на конец очереди

```
typedef struct list
{
```

```
node_t *head;
node_t *tail;
} list_t;
```

Описание очереди на массиве (очередь циклическая)
arr – массив элементов очереди
head – индекс начала очереди
tail – индекс конца очереди

```
typedef struct array
{
double arr[N];
int head;
int tail;
} array_t;
```

Описание структуры параметров моделирования
mint1 – минимальное время для T1
maxt1 – максимальное время для T1
mint2 – минимальное время для T2
maxt2 – максимальное время для T2
mint3 – минимальное время для T3
maxt3 – максимальное время для T3
mint4 – минимальное время для T4
maxt4 – максимальное время для T4

```
typedef struct times
{
double mint1;
double maxt1;
double mint2;
double maxt2;
double mint3;
double maxt3;
double mint4;
double maxt4;
} times_t;
```

ОПИСАНИЕ АЛГОРИТМА

1. Пользователь выбирает действие
2. Согласно выбранному действию вызывается соответствующая функция
    1. Функция моделирования работает по принципу выбора наиболее раннего действия. Сначала выполняется такое действие, потом меняется актуальное время моделирования
    2. Функция редактирования параметров изменяет параметры моделирования в соответствии с введенными пользователем
    3. Функция получения времени удаления и добавления в очередь для массива и списка проводит 1000 замеров каждой операции и усредненный результат печатает на экран
3. Выполняется логика этой функции
4. Если во время выполнения функции возникла ошибка, то выводится сообщение, поясняющее эту ошибку
5. Если ошибки не возникло, возвращаемся к 1 пункту

ОПИСАНИЕ ФУНКЦИЙ

`void description();` – функция выводит описание работы программы

`void menu();` – вывод меню для пользователя

`void process_list(times_t *pt);` – моделирование для очередей-списков с заданными параметрами моделирования pt

`void process_array(times_t *pt, array_t *q_1, array_t *q_2);` – моделирование для очередей-списков q_1 и q_2 с заданными параметрами моделирования pt

`void research_list();` – исследование базовых функций для очереди-списка (pop and push)

`void research_array();` – исследование базовых функций для очереди-массива (pop and push)

`bool aqueue_is_empty(const array_t *q);` – проверка на пустоту очереди-массива q. Возвращает true если очередь пустая и false если нет

`bool aqueue_is_full(const array_t *q);` – проверка на заполненность очереди-массива q. Возвращает true если очередь заполнена и false если нет

`int aqueue_push(array_t *q, double item);` – функция добавления в очередь-массив q элемента item. Возвращает код ошибки

`int aqueue_pop(array_t *q, double *item);` – функция удаления из очереди-массива q элемента. Результат удаления записан в item. Функция возвращает код ошибки

`node_t *node_create(double item);` – функция создания узла списка со значением item. Возвращает указатель на узел

`void node_free(node_t *node);` – функция очищает память из-под узла node

`list_t *lqueue_create(void);` – функция создания очереди-списка. Возвращает указатель на созданную очередь

`void lqueue_destroy(list_t *q);` – функция уничтожения очереди-списка q (последовательно очищает память из-под каждого узла)

`void lqueue_make_empty(list_t *q);` – функция делает очередь-список q пустой

`bool lqueue_is_empty(const list_t *q);` – проверка на пустоту очереди-списка q. Возвращает true если очередь пустая и false если нет

`bool lqueue_is_full(const list_t *q);` – проверка на заполненность очереди-списка q. Возвращает true если очередь заполнена и false если нет

`int lqueue_push(list_t *q, double item);` – функция добавления в очередь-список q элемента item. Возвращает код ошибки

`int lqueue_pop(list_t *q, double *item);` – функция удаления из очереди-списка q элемента. Результат удаления записан в item. Функция возвращает код ошибки

`double get_times1(times_t *t);` – функция генерирует случайное время T1, удовлетворяющее начальным параметрам моделирования t. Возвращает это время

`double get_times2(times_t *t);` – функция генерирует случайное время T2, удовлетворяющее начальным параметрам моделирования t. Возвращает это время

`double get_times3(times_t *t);` – функция генерирует случайное время T3, удовлетворяющее начальным параметрам моделирования t. Возвращает это время

`double get_times4(times_t *t);` – функция генерирует случайное время T4, удовлетворяющее начальным параметрам моделирования t. Возвращает это время

`void edit_times(times_t *t);` – функция изменения параметров моделирования t

`void print_times(times_t *t);` – функция печати текущих параметров моделирования t

НАБОР ТЕСТОВ

№	Описание теста	Ввод пользователя	Ответ программы
1	Выбор начального действия	abc	Предложение ввести действие заново
2	Выбор начального действия	Число не из интервала [0;4]	Предложение ввести действие заново
3	Ввод параметров моделирования	Максимум меньше минимума	Возврат параметров к начальному значению и печать ошибки
4	Ввод параметров моделирования	abc	Предложение ввести параметр заново
5	Выбор действия над параметрами моделирования	Не целое число из интервала [0;4]	Предложение ввести номер заново
6	Выбор начального действия	0	Завершение работы
7	Выбор начального	1	Моделирование для очереди-

	действия		списка
8	Выбор начального действия	2	Моделирование для очереди-массива
9	Выбор начального действия	3	Изменение параметров моделирования
10	Выбор начального действия	4	Исследование времени работы push and pop
11	Ввод параметров моделирования	1 5	Параметры установлены: 1 5
12	Выбор действия над параметрами моделирования	0	Завершить редактирование
13	Выбор действия над параметрами моделирования	1	Изменить T1
14	Выбор действия над параметрами моделирования	2	Изменить T2
15	Выбор действия над параметрами моделирования	3	Изменить T3
16	Выбор действия над параметрами моделирования	4	Изменить T4

ЗАМЕРЫ ВРЕМЕНИ БАЗОВЫХ ОПЕРАЦИЙ

В ходе эксперимента для получения результатов проводилось 1000 замеров по времени. Усредненный результат представлен в таблице

	Удаление		Добавление	
	M	C	M	C
Время выполнения операции	0.03	0.083	0.044	0.185

*Значения времени в таблице представлены в микросекундах

** M – массив, C – список

ЗАМЕРЫ ВРЕМЕНИ МОДЕЛИРОВАНИЯ

В ходе эксперимента для получения результатов проводилось 1000 замеров по времени. Усредненный результат представлен в таблице

Реализация	Количество заявок 1 типа обработано	Количество заявок 2 типа обработано	Время моделирования в е.в	Время моделирования в мс
M	1000	1987.6	2983.7	2.88
C	1000	1997.1	2991.2	3.01

* M – массив, C – список

Заметим, что с исходными параметрами моделирования количество обработанных заявок второго типа примерно в 2 раза больше. Время моделирования для исходных параметров моделирования 2983.7 и 2991.2 Теоретическое время работы 3000 е.в. Соответственно результаты примерно совпадают.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое FIFO и LIFO?

FIFO — методология применимая в очереди. Означает, что первый вошедший элемент, первым обрабатывается

LIFO — методология применимая в стеке. Означает, что поселедний вошедший элемент, первым обрабатывается

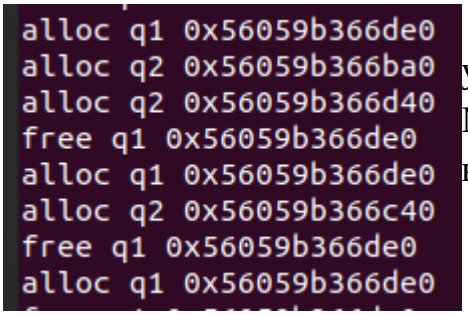
2. Каким образом, и какой объем памяти выделяется под хранение очереди при различной ее реализации?
При реализации массивом выделяется n * размер одного элемента или просто n. При реализации списком выделяется дополнительно n * размер указателя байтов памяти или 2*n. То есть реализации на массиве в 2 раза экономичнее.
3. Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?
При реализации списком сначала удаляется элемент, потом освобождается память. При реализации массивом просто изменяется указатель головы очереди.
4. Что происходит с элементами очереди при ее просмотре?
Значение записывается в локальную переменную и изменяется элемент, находящийся в голове очереди
5. От чего зависит эффективность физической реализации очереди?
От объема хранимой памяти
6. Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций?
Реализация на массиве быстрее по времени и эффективнее по памяти. Реализация на списке интуитивно понятнее.
7. Что такое фрагментация памяти, и в какой части ОП она возникает?
Фрагментация возникает в куче. Фрагментация — это такой эффект, при котором свободные участки памяти в куче чередуются с занятыми
8. Для чего нужен алгоритм «близнецов».
Алгоритм позволяет распределять память без фрагментации
9. Какие дисциплины выделения памяти вы знаете?
Статическое выделение памяти, динамическое выделение памяти: malloc, calloc, realloc
10. На что необходимо обратить внимание при тестировании программы?
На то, что является входными данными
11. Каким образом физически выделяется и освобождается память при динамических запросах?
Память выделяется полноценным блоком в куче. При освобождении памяти освобождается память из-под одного блока, который был выделен динамически

ВЫВОД

В рамках этой лабораторной работы я научился работать с очередью. Я реализовал работу с очередью двумя способами: с помощью статического массива и с помощью связного списка. Сравнив время добавления в очередь и удаления из нее я пришел к выводу:

1. По памяти эффективнее использовать очередь-массив. Выигрыш в 2 раза (для очереди-массива количество используемой памяти = n, для очереди-списка 2n)
2. По времени эффективнее использовать очередь-массив. (Для очереди-массива производится меньше операций)
    - Для добавления выигрыш в 4.2 раза
    - Для удаления порядка 28 раз

Помимо этого сравнив время моделирования очереди-массива и очереди-списка я выяснил, что реализация на очереди-массиве выигрывает всего на 5% Связано это с тем, что в главном цикле программы происходит очень много операций (изменение отчетных параметров и т.д). В таком случае разница во времени работы не столь наглядна. Если убрать все вычисления, то мы получим выигрыш порядка 20%



Так же я проанализировал, как выделяется и удаляется память при реализации на списке. Можно заметить, что фрагментация в памяти возникает.

При T3 > T1 первая очередь постепенно заполняется. Заявки второй очереди не обрабатываются. Соответственно время моделирования задается средним временем обработки первой заявки * на количество заявок (1000)

При исходных параметрах время моделирования задается средним временем поступления заявки в первую очередь * количество заявок (1000)

При T2 ≥ T4 ОА заполняется заявками второго типа. Соответственно может случиться переполнение первой очереди, т.к будут обрабатываться только заявки второго типа.