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 Равашдех Фадей Хешамович

Группа ИУ7 – 35Б

Преподаватель Никульшина Т. А.

2023

Оглавление

ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ.....3

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....3

ОПИСАНИЕ СТРУКТУР ДАННЫХ.....4

ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ.....5

ОПИСАНИЕ АЛГОРИТМА.....6

НАБОР ТЕСТОВ.....6

ОЦЕНКА ЭФФЕКТИВНОСТИ.....7

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....7

ВЫВОД.....9

Описание условия задачи

Система массового обслуживания состоит из двух обслуживающих аппаратов (ОА1 и ОА2) и двух очередей заявок. Всего в системе обращается 100 заявок.

Заявки поступают в "хвост" каждой очереди; в ОА они поступают из "головы" очереди по одной и обслуживаются по случайному закону за интервалы времени T1 и T2, равномерно распределенные от 0 до 6 и от 1 до 8 единиц времени соответственно. (Все времена – вещественного типа).
Каждая заявка после ОА1 с вероятностью P=0.7 вновь поступает в "хвост" первой очереди, совершая новый цикл обслуживания, а с вероятностью 1-P входит во вторую очередь. В начале процесса все заявки находятся в первой очереди.
Смоделировать процесс обслуживания до выхода из ОА2 первых 1000 заявок. Выдавать на экран после обслуживания в ОА2 каждых 100 заявок информацию о текущей и средней длине каждой очереди, а в конце процесса - общее время моделирования, время простоя ОА2, количество срабатываний ОА1, среднее времени пребывания заявок в очереди. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Описание технического задания

Входные данные: Номер пункта меню.
Выходные данные: Моделирование и сравнения эффективности для двух реализаций очередей.
Обращение к программе:
Запускается через терминал командой: ./app.exe.
Аварийные ситуации:
1. Некорректный ввод пункта;
2. Прекращение ввода (EOF);
3. Ввод некорректного пункта меню.
Пункты меню:
0. Информация о программе
1. Запустить моделирование для очереди на статическом массиве
2. Запустить моделирование для очереди на списке
3. Сравнение
4. Выйти

Описание структур данных

Структура заявки:

```
struct application
{
    double time;    // Время попадания заявки в очередь
};
```

Структура очереди на статическом массиве

```
struct array_queue_struct
{
    applic_t array[MAX_ARR_LEN];// Статисческий массив заявок
    size_t len;                 // Кол-во элементов

    applic_t* q_1;              // Адрес нижней границы
    applic_t* q_m;              // Адрес верхней границы
    applic_t* p_in;            // Указатель за последний элемент
    applic_t* p_out;            // Указатель на первый элемент
};
```

Структура очереди на списке

```
struct list
{
    list_t *next;   // Указатель на следующий элемент списка
    applic_t a;     // Заявка
};

struct list_queue_struct
{
    list_t **p_in;
    list_t *p_out;
};
```

Вспомогательная структура моделирования:

```
typedef struct
{
    double clock1;      // Таймер 1й очереди
    double clock2;      // Таймер 1й очереди
    double sleep2;      // Время простоя ОА2
    double average_in_q;// Время пребывания заявок в очереди

    size_t oa1_c;       // Количество срабатываний ОА1
    size_t oa2out_c;    // Счетчик вышедших из ОА2
    size_t oa2out_c100; // Счетчик сотен вышедших из ОА2
    size_t ql1;         // Длина 1й очереди
    size_t ql2;         // Длина 2й очереди
    double average_l1;  // Средняя длина 1й очереди * время
    double average_l2;  // Средняя длина 2й очереди * время
} measures_t;
```

Описание функций программы

Функции основного файла программы

```
// Контроллер программы
int controller(int option, bin_tree_t *tree, char **filename);
// Основная функция
int main(void);
```

Операции с очередью на массиве

```
sa_queue_t *saq_init(void); Инициализация
int saq_push(saq_queue_t *queue, applic_t a); Добавление
int saq_pop(saq_queue_t *queue, applic_t a); Удаление
```

Операции с очередью на списке

```
ll_queue_t *llq_init(void); Инициализация
int llq_push(llq_queue_t *queue, applic_t a); Добавление
int llq_pop(llq_queue_t *queue, applic_t *a); Удаление
void llq_free(llq_queue_t *queue);            Освобождение
```

Функции моделирования

```
void saq_model(void);   На массиве
void llq_model(void);   На списке
```

Функции замеров

```
long int measure_saq_add(sa_queue_t *queue, size_t n);
long int measure_saq_del(sa_queue_t *queue, size_t n);
long int measure_saq_mod(void);
long int measure_llq_add(ll_queue_t *queue, size_t n);
long int measure_llq_del(ll_queue_t *queue, size_t n);
long int measure_llq_mod(void);
int measure_data_n(size_t n);
int compare(void);
```

Функции ввода/вывода

```
// Вывод меню программы
void print_menu(void);
// Вывод информации о программе
void print_info(void);
// Чтение строки неопределен. длины из стандартного потока ввода
int get_input(char **str, FILE *f);
// Чтение строки определенной длины из стандартного потока ввода
int get_n_input(char pch[], size_t n);
// Получение выбранного пользователем пункта
int get_option(size_t *option);
// Вывод сообщения, поясняющего ошибку
void print_error_msg(int rc);
// Вывод сообщения, поясняющего ошибку, и возврат того же кода
возврата
int print_error(int rc);
```

Описание алгоритма

Добавление элемента в очередь на массиве: элемент записывается по указателю на конца очереди, указатель увеличивается на размер элемента.

Удаление элемента из очереди на массиве: указатель на начало очереди увеличивается на размер элемента, в указатель на элемент помещается значение «удаленного» элемента.

Добавление элемента в очередь на списке: выделяется память под новый узел списка, в который помещается новое значение. По указателю на на конец очереди списка записывается указатель на созданный элемент, указатель на конец меняется на указатель на указатель следующего элемента нового элемента.

Удаление элемента из очереди на списке: указателю на элемент присваевается значение удаляемого элемента. Указателю начала очереди присваивается указатель на следующий элемента списка, текущий удаляется.

Набор тестов

№	Название теста	Пользовательский ввод	Вывод
1	Некорректный пункт меню	`10`	`Некорректный номер пункта меню. Действия не существует. Ошибка: Действие с введенным пунктом отсутствует.`
2	Некорректный пункт меню	`abacaba`	Ошибка: Некорректный ввод.
3	Корректное использование	`1`	`*Моделирование очереди статическом на массиве*`
8	Выход из программы	`0`	`Выход из программы`

Оценка эффективности

Время в тактах за 100 повторений и память в байтаз

Кол-во элементов	Операция	Очередь на массиве	Очередь на списке
10	Добавление	106	133
	Удаление	100	127
	Память (байт)	840	176
50	Добавление	187	340
	Удаление	163	256
	Память (байт)	840	816
100	Добавление	255	544
	Удаление	178	373
	Память (байт)	840	1616

Как видно, операции с очередью на массиве быстрее операций очереди на списке, т.к. не происходит работы с памятью. При этом, количество памяти занимаемое при заполенности очереди меньше половины у списка меньше, а при заполненности выше половины, больше.

Ответы на контрольные 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 в виде статического списка, единоразово выделяется статическая память размером MAX_SIZE * sizeof(data_t) для MAX_SIZE элементов и 24 байта для 3-х указателей. При реализации списком для каждого элемента по мере необходимости выделяется sizeof(data_t) для данных и 8 байт для указателя на следующий элемент.

3. Каким образом освобождается память при удалении элемента из очереди при её различной реализации? При удалении элемента из очереди в виде массива, перемещается указатель, память не освобождается, т. к. массив статический. При удалении элемента из очереди в виде списка, указатель на «голову» переходит на следующий элемент, текущий удаляется, память освобождается.

4. Что происходит с элементами очереди при её просмотре? При просмотре очереди, головной элемент удаляется, поэтому при просмотре очереди ее элементы удаляются.

5. От чего зависит эффективность физической реализации очереди? От реализации очереди и размера элементов.

6. Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций? При реализации очереди в виде массива, размер очереди ограничен и постоянен. Операции добавления и удаления элементов работают быстрее (при кольцевой реализации). При реализации в виде списка размер ограничен только оперативной памятью и не постоянен, но элементы очереди занимают в 2 раза больше памяти за счет указателей. Если изначально знать размер очереди, то лучше воспользоваться массивом, иначе списком.

7. Что такое фрагментация памяти, и в какой части ОП она возникает? Фрагментация памяти — явление, при котором память выделена не последовательно в одном блоке памяти, а фрагментирована на множество частей. Фрагментация может возникнуть при неэффективном выделении и освобождении памяти, когда участки занятой памяти расположены не последовательно, создавая промежутки между ними.

8. Для чего нужен алгоритм «близнецов». Этот алгоритм обеспечивает высокую скорость выделения и освобождения динамической памяти и минимизации фрагментации памяти.

9. Какие дисциплины выделения памяти вы знаете? Основные дисциплины выделения памяти: «самый подхходящий» - выбирается участок с наименьшим размером из походящих, «первый подходящий» - выбирается первый участок с размером не меньшим требуемоего.

10. На что необходимо обратить внимание при тестировании программы? На возможность переполнения очереди или попытки забрать элемент из пустой очереди, на правильность работы обслуживающих аппаратов, расчета времени моделирования, функций-генераторов случайных значений.

11. Каким образом физически выделяется и освобождается память при динамических запросах? При выделении выбирается свободный участок памяти определенного размера в памяти и помечается как занятый. При освобождении этот участок помечается ка свободный.

Вывод

Операции с очередью на статическом массиве выполняются быстрее, чем операции с очередью на списке, также очередь на массиве, при заполнении больше половины, на статическом массиве занимает постоянное количество памяти и ограничена максимальным размером очереди. Очередь на списке при заполнении меньше половины занимает меньше памяти, чем очередь на статическом массиве, т. к. даже несмотря на испоользование дополнительной памяти на указатели, размер очереди зависит от количества элементов очереди, также очередь на списке, в отличие от очереди на массиве, ограничена только ОП компьютера, а значит может быть использована при заранее неизвестном количестве элементов.

Если заранее известно количество элементов, которое будет находиться в очереди, то лучше использовать реализацию очереди на статическом массиве, т. к. он работает быстрее, а ограниченность в размере не влияет, т. к. количество элементов известно. Если количество элементов не известно, то тогда нужно использовать реализацию очреди на списке, т. к очередь на списке не ограничена.