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»
Вариант 6

Студент Кладницкий А. Б.

Преподаватель

Группа ИУ7 – 32Б

2022 г.

1. **Описание условия задачи**

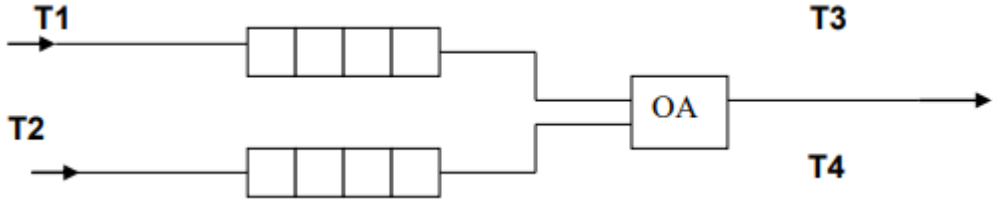
Система массового обслуживания состоит из обслуживающих аппаратов (ОА) и очередей заявок двух типов, различающихся временем прихода и обработки. Заявки поступают в очереди по случайному закону с различными интервалами времени (в зависимости от варианта задания), равномерно распределенными от начального значения (иногда от нуля) до максимального количества единиц времени. В ОА заявки поступают из «головы» очереди по одной и обслуживаются за указанные в задании времена, распределенные равновероятно от минимального до максимального значений (все времена – вещественного типа).

2. **Т3**

Требуется смоделировать процесс обслуживания первых 1000 заявок первого типа, выдавая после обслуживания каждых 100 заявок первого типа информацию о текущей и средней длине каждой очереди и о среднем времени пребывания заявок каждого типа в очереди. В конце процесса необходимо выдать на экран общее время моделирования, время простоя ОА, количество вошедших в систему и вышедших из нее заявок первого и второго типов.

Очередь необходимо представить в виде вектора и списка. Все операции должны быть оформлены подпрограммами. Алгоритм для реализации задачи один, независимо от формы представления очереди. Необходимо сравнить эффективность различного представления очереди по времени выполнения программы и по требуемой памяти. При реализации очереди списком нужно проследить, каким образом происходит выделение и освобождение участков памяти, для чего по запросу пользователя необходимо выдать на экран адреса памяти, содержащие элементы очереди при добавлении или удалении очередного элемента.

Схема:



1. Исходные данные:
   - Пункт меню
   - Измененные времена для обработки (при необходимости) и запрос на их изменение
   - Меню:
     1. Emulate mass service
     2. Check efficiency of queue as list
     3. Check efficiency of queue as array
2. Результирующие данные:
   - Таблица с данными расчета эмулированной системы:
     - Добавлено в очереди
     - Удалено из очередей

        - Средняя длина очередей
        - Текущая длина очередей
    - Адреса памяти
    - Результаты замеров
3. Задача программы:
    - Эмулирование системы обслуживания
    - Логирование действий, связанных с динамической памятью
    - Замер эффективности разных реализаций очереди
4. Способ обращения к программе:
    - Запуск через терминал (./app.exe)
5. Возможные ошибки:
    - Ошибка выделения динамической памяти
    - Переполнение очереди
    - Ошибка чтения из потока
    - Ошибка чтения вещественного числа

3. Описание внутренних структур данных

Тип данных queue_t (очередь на списке):

```
typedef struct queue
{
  node_t *pb;
  node_t *pe;
  size_t len;
  size_t len_max;
} queue_t;
```

Тип содержит в себе указатели на начало и конец очереди, текущий и максимальный размеры.

Тип данных aqueue_t (очередь на массиве):

```
typedef struct aqueue
{
  double *data;
  int beg;
  int end;
  size_t len;
  size_t len_max;
} aqueue_t;
```

Тип aqueue_t содержит в себе указатель на область с данными (сам массив), индексы начального и конечного элемента из очереди, текущий и максимальный размеры.

```
typedef struct node
{
  void *data;
  node_t *next;
} node_t;
```

Тип node_t хранит в себе значения data узла односвязного списка и ссылку на следующий узел (NULL, если конец).

4. **Алгоритм**
    1. Считать пункт меню
    2. При необходимости, для эмулирования изменить заданные времена
    3. Произвести действие по выбору пользователя (эмулирование/замер)
5. **Набор тестов**

	Описание	Входные данные	Результат
1.	Невозможно открыть лог-файл	???	Сообщение о невозможности открытия лог-файла
2.	Неверный пункт меню	q, 11, -5 и т.д.	EXIT_UNKNOWN Сообщение о том, что пункт меню неизвестен
3.	Ошибка выделения динамической памяти	???	EXIT_ALLOCATE
4.	Неверное время (T1/T2/T3/T4)	112.1h8	EXIT_BAD_DOUBLE
5.	Пропущено одно из времен	T1: 1 T2: 1 2 T3: 2 4 T4: 4 8	EXIT_BAD_DOUBLE
6.	Запрос замеров времени	Пункт меню 2/3	Результаты замеров
7.	Эмуляция со стандартным временем	Пункт меню 1	Таблица результатов измерения
8.	Эмуляция с пользовательским временем	Пункт меню 1 Времена T1, T2, T3, T4	Таблица результатов измерения

6. Пример эмуляции

```
Theorethic:
||=================================================================||
||  Input count  ||Output count  ||  Current len  ||     Average len     ||
||===============================================================||
||   Q1   |   Q2   ||   Q1   |   Q2   ||   Q1   |   Q2   ||    Q1     |    Q2     ||
||=======|=======||=======|=======||=======|=======||===========|===========||
||    100|    200||     99|    199||      1|      1||  0.502513|  0.501253||
||    200|    400||    199|    399||      1|      1||  0.501253|  0.500626||
||    300|    600||    299|    599||      1|      1||  0.500835|  0.500417||
||    400|    800||    399|    799||      1|      1||  0.500626|  0.500313||
||    500|   1000||    499|    999||      1|      1||  0.500501|  0.500250||
||    600|   1200||    599|   1199||      1|      1||  0.500417|  0.500208||
||    700|   1400||    699|   1399||      1|      1||  0.500357|  0.500179||
||    800|   1600||    799|   1599||      1|      1||  0.500313|  0.500156||
||    900|   1800||    899|   1799||      1|      1||  0.500278|  0.500139||
||   1000|   2000||    999|   1999||      1|      1||  0.500250|  0.500125||
||   1000|   2000||   1000|   1999||      0|      1||  0.500000|  0.500125||
||=================================================================||

Total time of emulation:   3002.000000
Total time without work:   2.500000
Total count of req #1 in:  1000
Total count of req #1 out: 1000
Total count of req #2 in:  2000
Total count of req #2 out: 1999
```

```
Queue as list:
||=================================================================||
||  Input count  ||Output count  ||  Current len  ||     Average len     ||
||===============================================================||
||   Q1   |   Q2   ||   Q1   |   Q2   ||   Q1   |   Q2   ||    Q1     |    Q2     ||
||=======|=======||=======|=======||=======|=======||===========|===========||
||    100|    185||     99|    185||      1|      0||  0.502513|  0.500000||
||    200|    390||    199|    390||      1|      0||  0.501253|  0.500000||
||    300|    590||    299|    590||      1|      0||  0.500835|  0.500000||
||    400|    778||    399|    778||      1|      0||  0.500626|  0.500000||
||    500|    973||    499|    973||      1|      0||  0.500501|  0.500000||
||    600|   1192||    599|   1192||      1|      0||  0.500417|  0.500000||
||    700|   1393||    699|   1393||      1|      0||  0.500357|  0.500000||
||    800|   1594||    799|   1594||      1|      0||  0.500313|  0.500000||
||    900|   1794||    899|   1794||      1|      0||  0.500278|  0.500000||
||   1000|   2004||    999|   2004||      1|      0||  0.500250|  0.500000||
||   1000|   2004||   1000|   2004||      0|      0||  0.500000|  0.500000||
||=================================================================||

Total time of emulation:   3013.470171
Total time without work:   20.890266
Total count of req #1 in:  1000
Total count of req #1 out: 1000
Total count of req #2 in:  2004
Total count of req #2 out: 2004
```

7. Оценка эффективности

Приведена эффективность реализации на массиве относительно реализации на списке.

Время (мкс):

Кол-во элементов	Push			Pop		
	Список	Массив	Эфф. %	Список	Массив	Эфф. %
50	10	3	333%	3	2	150%
100	10	5	200%	7	5	140%
250	35	12	292%	23	13	177%
500	54	22	245%	30	23	130%
1000	95	43	221%	59	46	128%

Память (байт):

Кол-во элементов	Список	Массив	Относит. эффективность
50	832	432	193%
100	1632	832	196%
250	4032	2032	198%
500	8032	4032	199%
1000	16032	8032	200%

8. Вывод

Реализация очереди на списке уступает реализации очереди на динамическом массиве, как по памяти, так и по времени, для любых размерностей. Использование списка не выгодно.

9. Контрольные вопросы

1. Что такое FIFO и LIFO?
   Разные структуры данных.
   Структура FIFO – first in first out – очередь
   Структура LIFO – last in first out – стек
2. Каким образом, и какой объем памяти выделяется под хранение очереди при различной ее реализации?
   При реализации списком, хранятся узлы списка, а также указатель на начало и конец очереди. Опциональны размеры.
   При реализации массивом, хранятся данные (сам массив), указатель на начало и конец очереди, размер.
   Реализация на массиве занимает в 2 раза меньше памяти.
3. Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?
   Для массива освобождение не требуется, просто сдвигаются указатели.
   Для списка освобождается узел из начала очереди, начало сдвигается на следующий элемент.
4. Что происходит с элементами очереди при ее просмотре?
   Элемент удаляется из очереди
5. От чего зависит эффективность физической реализации очереди?

От выбора структуры для представления очереди. Выгоднее динамический массив.

6. Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций?
   Динамический массив преобладает над списком и по времени, и по памяти. Статический массив уступает списку максимально возможной используемой памятью (его размер ограничен аппаратным стеком).
7. Что такое фрагментация памяти, и в какой части ОП она возникает?
   Разбиение памяти на куски, лежащие друг за другом. Таким образом, нельзя выделить большой цельный кусок памяти.
8. Для чего нужен алгоритм «близнецов».
   Выделения памяти методом «близнецов»: размер кучи берется кратным степени двойки и берется наименьший возможный блок требуемых размеров.
9. Какие дисциплины выделения памяти вы знаете?
   Best fir и First fit
   Первый находит «самый подходящий» по размеру блок памяти, в то время как второй находит и выделяет первый попавшийся, размер которого не меньше запрашиваемого. First fit эффективнее по времени.
10. На что необходимо обратить внимание при тестировании программы?
    Покрыть все аварийные ситуации
    Покрыть все ветки условных операторов
    Желательно полностью покрыть код
11. Каким образом физически выделяется и освобождается память при динамических запросах?
    Хранится информация о выделенных/невыделенных блоках памяти (битовая карта или список). При выделении находится блок подходящего размера (в зависимости от дисциплины) и выделяется. Для освобождения есть несколько вариантов:
    - Явный запрос
    - «Сборка мусора»
      Освобождение, когда свободной памяти не осталось
    - Освобождение, когда блок перестает использоваться
    - «Уплотнение»
      Физическое передвижение блоков памяти с целью сбора свободных блоков в один большой