



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

Гаврилюк Владислав Анатольевич

Группа

ИУ7 – 31Б

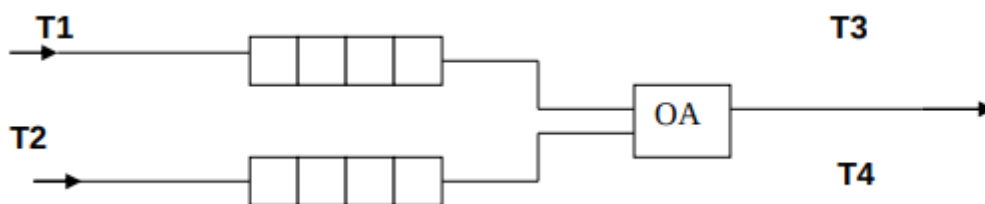
Преподаватель  
Проверил

Барышникова Марина Юрьевна

<b>Описание условия задачи.....</b>	<b>3</b>
<b>Описание структур данных.....</b>	<b>4</b>
<b>Сравнение эффективности (в ед.вр.).....</b>	<b>7</b>
<b>Примеры работы программы.....</b>	<b>7</b>
<b>Ответы на контрольные вопросы.....</b>	<b>13</b>
<b>Вывод.....</b>	<b>14</b>

## 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:

- она вошла в пустую систему;
- перед ней обслуживалась заявка ее же типа;
- 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

### Задачи программы:

1. Изменение установленных временных параметров и необходимого количества обработанных заявок.

2. Вывод установленных параметров на экран
3. Моделирование обработки заявок с помощью очередей, реализованных через массив.
4. Моделирование обработки заявок с помощью очередей, реализованных через связный список.

### **Обращение к программе:**

Команда запуска: `./app.exe`

### **Входные данные:**

Пользователь может изменить временные параметры и количество заявок, которые нужно обработать.

### **Аварийные ситуации:**

- Указан некорректный номер операции
- Данные введены не в указанном формате
- Невозможность выделения памяти для указанной последовательности

Ошибочные ситуации сопровождаются поясняющим сообщением об ошибке.

## **ОПИСАНИЕ СТРУКТУР ДАННЫХ**

В программе представлен следующий набор структур:

1. **node\_t** - узел связного списка.

---

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

---

**char data** - поле, содержащее тип заявки.

**node\_t \*next** - указатель на следующий элемент списка.

2. **time\_range\_t** - структура, содержащая границы временного интервала.

---

```
typedef struct time_range
{
    double min;
    double max;
} time_range_t;
```

---

**double min** - нижняя граница интервала.

**double max** - верхняя граница интервала.

3. **ranges\_t** - структура, содержащая 4 временных интервала.

---

```
typedef struct ranges_t
{
    time_range_t t1;
    time_range_t t2;
    time_range_t t3;
    time_range_t t4;
} ranges_t;
```

---

4. **parameters\_t** - структура, в которой содержится промежуточная и итоговая информация о состоянии очереди.
- 

```
typedef struct parameters
{
    void* p_in;
    void* p_out;
    void* low;
    void* up;
    int max_num;
    int count_request;
    int sum_size;
    int curr_size;
    int out_request;
    int in_request;
} parameters_t;
```

---

**void \*p\_in** - указатель на конец очереди

**void \*p\_out** - указатель на начало очереди

**void \*low** - указатель на начало массива

**void \*up** - указатель на конец массива

**int max\_num** - максимальная длина очереди

**int sum\_size** - сумма всех длин очереди при различных состояниях

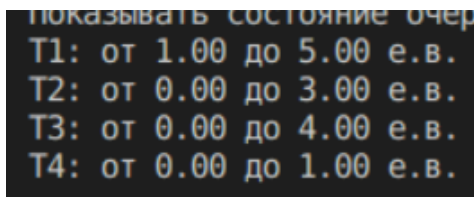
**int curr\_size** - текущий размер очереди

**int out\_request** - количество заявок, покинувших очередь

**int in\_request** - количество заявок, вошедших в очередь

## ПРОВЕРКА КОРРЕКТНОСТИ МОДЕЛИРОВАНИЯ (в ЕД.ВР.)

Эксперименты проводились при стандартных временных параметрах.



показывать состояние очер  
T1: от 1.00 до 5.00 е.в.  
T2: от 0.00 до 3.00 е.в.  
T3: от 0.00 до 4.00 е.в.  
T4: от 0.00 до 1.00 е.в.

Ожидаемое время моделирования  $T_{\text{exр}} = n * \max(T1_{\text{ср}}, T3_{\text{ср}})$

Количество заявок	Время моделирования для массива	Погрешность для массива	Время моделирования для списка	Погрешность для списка	Ожидаемое время моделирования
100	295.74	1.41%	303.72	1.24%	300
500	1523.15	1.54%	1528.30	1.88%	1500
1000	3032.80	1.09%	3057.67	1.92%	3000
2000	6057.78	0.96%	6047.58	0.79%	6000
3000	9048.04	0.53%	9065.05	0.72%	9000
4000	12050.34	0.41%	12106.84	0.89%	12000
5000	15028.61	0.19%	15101.49	0.67%	15000

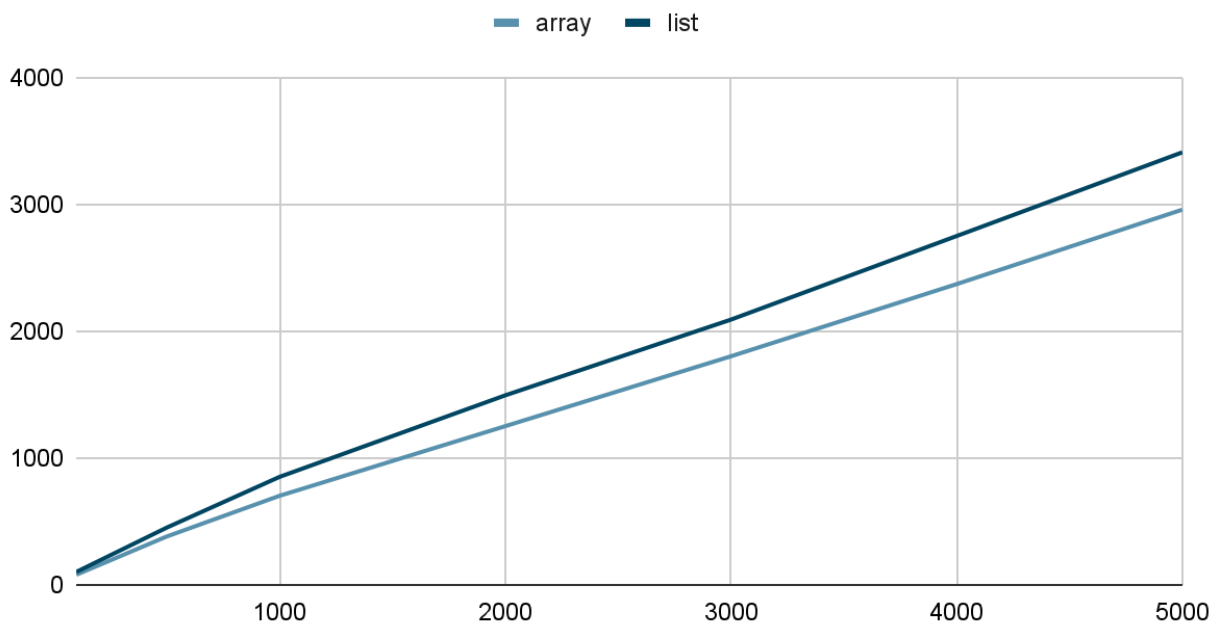
## СРАВНЕНИЕ ЭФФЕКТИВНОСТИ

Время измеряется в микросекундах, память - в байтах.

Количество заявок	Время моделирования для массива	Среднее количество используемой памяти для массива	Время моделирования для списка	Среднее количество используемой памяти для списка
100	80	20016	102	150
500	381	20016	451	736

1000	702	20016	851	1104
2000	1251	20016	1494	1508
3000	1802	20016	2091	1792
4000	2371	20016	2750	2012
5000	2958	20016	3410	2576

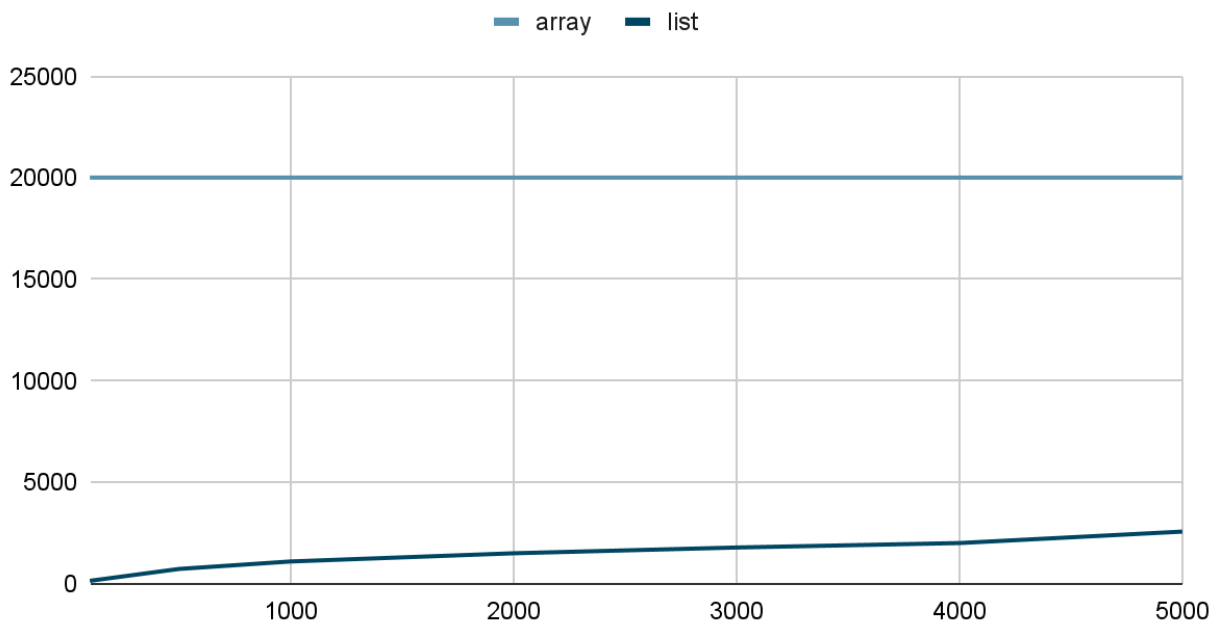
### Сравнение эффективности по времени (мкс)



Можно сделать вывод, что реализация очереди с помощью статического массива эффективнее по времени реализации через связный список примерно на 15-20%.



## Сравнение эффективности по памяти



В силу того, что замеры времени проводились при стандартных значениях, длина очереди стабилизировалась на небольшом значении, и, соответственно, для связного списка выделялся малый объем памяти. А для статического массива количество выделенной памяти постоянно: 10000 элементов (а т.к. очереди две, то объем выделенной памяти увеличивается в 2 раза).

```
показывать состояние очер
T1: от 1.00 до 5.00 е.в.
T2: от 0.00 до 3.00 е.в.
T3: от 0.00 до 4.00 е.в.
T4: от 0.00 до 1.00 е.в.
```

## ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

При запуске программы выводится меню:

```
=====
|      Обработка очередей. Взаимодействие с программой      |
|-----|
| 1. Ввести значения вручную                                   |
| 2. Вывести установленные значения                           |
| 3. Произвести моделирование для массива                     |
| 4. Произвести моделирование для списка                       |
| 0. Выход                                                     |
|-----|
Введите номер операции: █
```

При выборе команды 2 мы получим следующее сообщение, в котором указаны стартовые временные параметры и количество заявок для обслуживания, соответствующие заданию:

```
Введите номер операции: 2
Количество заявок для обслуживания: 1000.
Показывать состояние очередей каждые 100 обработанных заявок.
T1: от 1.00 до 5.00 е.в.
T2: от 0.00 до 3.00 е.в.
T3: от 0.00 до 4.00 е.в.
T4: от 0.00 до 1.00 е.в.
=====
```

Выполним моделирование с указанными параметрами для массива:

1. Вывод промежуточного состояния очередей:

500	на данный момент	в среднем
1-ая очередь	0	2
2-ая очередь	2	11
600	на данный момент	в среднем
1-ая очередь	3	3
2-ая очередь	2	12
700	на данный момент	в среднем
1-ая очередь	5	3
2-ая очередь	10	13
800	на данный момент	в среднем
1-ая очередь	9	4
2-ая очередь	21	17
900	на данный момент	в среднем
1-ая очередь	5	5
2-ая очередь	57	22

## 2. Итоговый результат:

Моделирование для массива		
	Вход	Выход
1-ая очередь	1002	1000
2-ая очередь	2010	1884
- Время моделирования: 2955.145052 тиков (1735 мс)		
- Время простоя: 49.514734		
- Ожидаемое время моделирования: 3000.000000		
- Погрешность: 1.495165%		
- Средний размер: 20016 байт		

Выполним моделирование для связного списка:

1. Вывод промежуточного состояния очередей:

400	на данный момент	в среднем
1-ая очередь	1	4
2-ая очередь	6	17
500	на данный момент	в среднем
1-ая очередь	6	5
2-ая очередь	13	18
600	на данный момент	в среднем
1-ая очередь	2	5
2-ая очередь	18	19
700	на данный момент	в среднем
1-ая очередь	1	5
2-ая очередь	16	19
800	на данный момент	в среднем
1-ая очередь	2	4
2-ая очередь	2	17
900	на данный момент	в среднем
1-ая очередь	1	4
2-ая очередь	9	18

2. Результат:

Моделирование для связного списка		
	Вход	Выход
1-ая очередь	1001	1000
2-ая очередь	2062	2054

- Время моделирования: 3043.741795 тиков (3827 мс)  
 - Время простоя: 87.505188  
 - Ожидаемое время моделирования: 3000.000000  
 - Погрешность: 1.458060%  
 - Средний размер: 352 байт

При моделировании для связного списка можно вывести результаты работы с памятью:

```

Показать результаты работы с памятью? 1/0: 1
Переиспользованных адресов: 2965
Не использовались повторно: 59
Первые 15 из них:
[0x55ee3fd02b00]
[0x55ee3fd02f00]
[0x55ee3fd03020]
[0x55ee3fd030c0]
[0x55ee3fd03040]
[0x55ee3fd02be0]
[0x55ee3fd03160]
[0x55ee3fd02c20]
[0x55ee3fd032c0]
[0x55ee3fd030a0]
[0x55ee3fd02d80]
[0x55ee3fd02c00]
[0x55ee3fd02e00]
[0x55ee3fd02d00]
[0x55ee3fd02ba0]
  
```

**Можно заметить, что при работе со связным списком присутствует небольшая фрагментация памяти.**

Теперь изменим временные параметры:

```

=====
Введите номер операции: 1

Введите количество заявок для обслуживания: 1000

Показывать промежуточное состояние очередей? 1/0: 0

Внести изменения во временные диапазоны? 1/0: 1

Введите T1_min T1_max значения через пробел: 0 4

Введите T2_min T2_max значения через пробел: 0 3

Введите T3_min T3_max значения через пробел: 0 3

Введите T4_min T4_max значения через пробел: 0 1
=====

```

Очевидно, что время моделирования и время простоя тоже изменились:

```

=====
|                                     |
|               Моделирование для массива               |
|-----|-----|-----|
|               |               Вход               |               Выход               |
|-----|-----|-----|
| 1-ая очередь |               1049               |               1000               |
|-----|-----|-----|
| 2-ая очередь |               1336               |               1099               |
|-----|-----|-----|
- Время моделирования: 2055.677573 тиков (875 мс)
- Время простоя: 4.472854
- Ожидаемое время моделирования: 2000.000000
- Погрешность: 2.783879%
- Средний размер: 20016 байт
=====

```

Если изменить временные параметры так, чтобы время поступления заявок было меньше времени их обработки, то мы получим переполнение очереди:

```

=====
Введите номер операции: 1

Введите количество заявок для обслуживания: 1000

Показывать промежуточное состояние очередей? 1/0: 0

Внести изменения во временные диапазоны? 1/0: 1

Введите T1_min T1_max значения через пробел: 0 3

Введите T2_min T2_max значения через пробел: 0 2

Введите T3_min T3_max значения через пробел: 1 5

Введите T4_min T4_max значения через пробел: 1 2
=====

```

```
=====
|      Обработка очередей. Взаимодействие с программой      |
|-----|
| 1. Ввести значения вручную                                   |
| 2. Вывести установленные значения                           |
| 3. Произвести моделирование для массива                     |
| 4. Произвести моделирование для списка                       |
| 0. Выход                                                     |
|-----|
Введите номер операции: 3
Первая очередь переполнена.
```

## ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

### 1. Что такое очередь?

Очередь – последовательный список переменной длины. Включение элементов происходит с «хвоста» списка, исключение – с «головы» списка. Принцип работы: первым вошел – первым вышел (FIFO)

### 2. Каким образом, и какой объем памяти выделяется под хранение очереди при различной ее реализации?

При реализации очереди с помощью связного списка, память будет выделяться динамически по мере необходимости (до определенного установленного предела или пока есть место в ОЗУ). При реализации очереди с помощью статического массива, создается массив, длина которого соответствует максимальной длине очереди.

### 3. Каким образом освобождается память при удалении элемента из очереди при ее различной реализации?

При реализации через связный список сначала смещается указатель на следующий элемент очереди, а затем освобождается память из-под первого элемента (“головы очереди”).

При реализации через статический массив освобождения памяти не происходит, смещается только указатель на следующую позицию.

### 4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди первый элемент будет удаляться, пока очередь не станет пустой.

### **5. Каковы достоинства и недостатки различных реализаций очереди в зависимости от выполняемых над ней операций?**

При реализации очереди массивом не возникает фрагментации памяти, но затрачивается время на сдвиг элементов. Чтобы этого избежать, можно использовать кольцевой массив, но будет сложнее реализовать операции работы с такой очередью. Также важным недостатком статического массива является невозможность его расширения при заполнении. При реализации списком может возникнуть фрагментация памяти, но объем занимаемой памяти ограничен только размерами ОЗУ.

### **6. Что такое фрагментация памяти?**

Фрагментация – чередование занятых и свободных участков памяти при последовательных запросах на выделение и освобождение. Свободные участки могут быть слишком малы, чтобы хранить в них полезную информацию.

### **7. На что необходимо обратить внимание при тестировании программы?**

Необходимо обратить внимание на корректное освобождение памяти при удалении элемента из очереди.

### **8. Каким образом физически выделяется и освобождается память при динамических запросах?**

Программа запрашивает блок памяти необходимого 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, адрес считается освобожденным. При попытке доступа к освобожденной памяти могут возникнуть ошибки.

## **Вывод**

В ходе данной лабораторной работы мы реализовали готовую программу, моделирующую работу системы с чередующимся приоритетом.



Проведя анализ полученных результатов, можно заключить, что время моделирования для обеих реализаций отличается от ожидаемого не более, чем на 3%.

Сравнение двух реализаций показало, что использование статического массива дает 15-20% (процентное) преимущество в скорости, но при этом требует больших объемов памяти, поэтому использование статического массива для очереди не всегда будет оправдано.