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

Студент **Паламарчук А.Н.**

Группа **ИУ7-33Б**

Предмет **Типы и структуры данных**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Паламарчук А.Н.**

Преподаватель _____ **Никульшина Т. А.**

Преподаватель _____ **Барышникова М. Ю.**

2023 г.

Условие задачи

Создать программу работы для работы с типом данных «очередь». Реализовать очередь на основе массива и односвязного списка. Используя созданный тип данных, провести моделирование обработки запросов.

Техническое задание

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 2-го типа может войти в ОА, если в системе нет заявок 1-го типа. Если в момент обслуживания заявки 2-го типа в пустую очередь входит заявка 1-го типа, то она немедленно поступает на обслуживание; обработка заявки 2-го типа прерывается и она возвращается в "хвост" своей очереди (система с абсолютным приоритетом и повторным обслуживанием).

Смоделировать процесс обслуживания первых 1000 заявок 1-го типа, выдавая после обслуживания каждых 100 заявок 1-го типа информацию о текущей и средней длине каждой очереди. В конце процесса выдать общее время моделирования и количестве вошедших в систему и вышедших из нее заявок обоих типов, среднем времени пребывания заявок в очереди, количестве «выброшенных» заявок второго типа. Обеспечить по требованию пользователя выдачу на экран адресов элементов очереди при удалении и ОА добавлении элементов. Проследить, возникает ли при этом фрагментация памяти.

Входные данные

- Команда (число) из меню программы (см. описание алгоритма) – обозначение необходимой операции.

Выходные данные

- Результаты моделирования для очереди, основанной на массиве
- Результаты моделирования для очереди, основанной на списке
- Адреса элементов

Возможные аварийные ситуации

- Некорректная величина времени (при ручном изменении)
- Некорректная команда

- Недостаток памяти
- Переполнение очереди

Описание внутренних структур данных

```
typedef struct item_t
{
    int number;
    size_t priority_queue;
    double time_in;
    double time_proc;
} item_t;

typedef struct node_t
{
    struct node_t *next;
    item_t val;
} node_t;

typedef struct queue_array_t
{
    item_t arr[MAX_SIZE];
    size_t size;
    // head (delete)
    item_t *pin;
    // tail (insert)
    item_t *pout;
    double t_in_max;
    double t_in_min;
    double t_proc_max;
    double t_proc_min;
} queue_array_t;

typedef struct queue_list_t
{
    node_t *head;
    node_t *tail;
    size_t len;
    double t_in_max;
    double t_in_min;
    double t_proc_max;
    double t_proc_min;
} queue_list_t;
```

Константы

```
#define LEN 1000
```

Используемые функции

```
unsigned long long cur_mks_gettimeofday(void);
```

Возвращает текущее время

```
void clean_stdin(void);
```

Очищает stdin

```
int process_array(queue_array_t *a, queue_array_t *b);
```
Проводит моделирование процесса для очередей, основанных на массивах

```
int process_list(queue_list_t *a, queue_list_t *b);
```
Проводит моделирование процесса для очередей, основанных на списках

```
int input_times(queue_array_t *a, queue_array_t *b, queue_list_t *c,
queue_list_t *d);
```
Ввод новых времен

```
void print_addresses(void);
```
Распечатывает задействованные адреса

Меню программы

Программа обрабатывает нужную команду:

```
Комманды:
1 - Смоделировать процесс, используя очереди, основанные на массивах
2 - Смоделировать процесс, используя очереди, основанные на списках
3 - Вывести использованные адреса
4 - Изменить исходные времена (вручную)
0 - Завершить работу программы
```

Замеры

Для массива: 1-2

Для списка: 3-4

```
------------------------------------------------
Начало моделирования
------------------------------------------------
Текущее время моделирования: 300.24
Обработано заявок 1-го типа: 100
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.79
Обработано заявок 2-го типа: 28
Текущая длина 2-ой очереди: 196
Средняя длина 2-ой очереди: 381.57
Время простоя: 19.50
Среднее время обработки заявок: 1.40
------------------------------------------------
Текущее время моделирования: 620.41
Обработано заявок 1-го типа: 200
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.51
Обработано заявок 2-го типа: 89
Текущая длина 2-ой очереди: 414
Средняя длина 2-ой очереди: 424.19
Время простоя: 46.05
Среднее время обработки заявок: 1.12
------------------------------------------------
Текущее время моделирования: 927.48
Обработано заявок 1-го типа: 300
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.45
Обработано заявок 2-го типа: 124
Текущая длина 2-ой очереди: 620
Средняя длина 2-ой очереди: 488.77
Время простоя: 65.14
Среднее время обработки заявок: 1.17
------------------------------------------------
Текущее время моделирования: 1231.63
Обработано заявок 1-го типа: 400
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.44
Обработано заявок 2-го типа: 171
Текущая длина 2-ой очереди: 817
Средняя длина 2-ой очереди: 580.74
Время простоя: 85.59
Среднее время обработки заявок: 1.14
------------------------------------------------
Текущее время моделирования: 1537.61
Обработано заявок 1-го типа: 500
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.42
Обработано заявок 2-го типа: 214
Текущая длина 2-ой очереди: 1019
Средняя длина 2-ой очереди: 655.56
Время простоя: 110.71
Среднее время обработки заявок: 1.13
------------------------------------------------
```

```
------------------------------------------------
Текущее время моделирования: 1822.74
Обработано заявок 1-го типа: 600
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.49
Обработано заявок 2-го типа: 245
Текущая длина 2-ой очереди: 1211
Средняя длина 2-ой очереди: 758.87
Время простоя: 125.60
Среднее время обработки заявок: 1.15
------------------------------------------------
Текущее время моделирования: 2136.77
Обработано заявок 1-го типа: 700
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.47
Обработано заявок 2-го типа: 303
Текущая длина 2-ой очереди: 1405
Средняя длина 2-ой очереди: 845.40
Время простоя: 149.93
Среднее время обработки заявок: 1.12
------------------------------------------------
Текущее время моделирования: 2435.24
Обработано заявок 1-го типа: 800
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.47
Обработано заявок 2-го типа: 347
Текущая длина 2-ой очереди: 1612
Средняя длина 2-ой очереди: 917.35
Время простоя: 166.36
Среднее время обработки заявок: 1.11
------------------------------------------------
Текущее время моделирования: 2728.96
Обработано заявок 1-го типа: 900
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.48
Обработано заявок 2-го типа: 375
Текущая длина 2-ой очереди: 1796
Средняя длина 2-ой очереди: 988.57
Время простоя: 182.91
Среднее время обработки заявок: 1.14
------------------------------------------------
Текущее время моделирования: 3024.97
Обработано заявок 1-го типа: 1000
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.48
Обработано заявок 2-го типа: 409
Текущая длина 2-ой очереди: 1990
Средняя длина 2-ой очереди: 1110.59
Время простоя: 205.67
Среднее время обработки заявок: 1.15
------------------------------------------------
Итоговые результаты (массив)
Общее время моделирования: 3025.69
Время простоя: 205.67
Количество обработанных заявок 1-го типа: 1000
Количество обработанных заявок 2-го типа: 409
```

```
------------------------------------------------
Начало моделирования
------------------------------------------------
Текущее время моделирования: 296.92
Обработано заявок 1-го типа: 100
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.79
Обработано заявок 2-го типа: 116
Текущая длина 2-ой очереди: 198
Средняя длина 2-ой очереди: 107.23
Время простоя: 17.03
Среднее время обработки заявок: 1.38
------------------------------------------------
Текущее время моделирования: 617.99
Обработано заявок 1-го типа: 200
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.50
Обработано заявок 2-го типа: 358
Текущая длина 2-ой очереди: 416
Средняя длина 2-ой очереди: 189.49
Время простоя: 42.94
Среднее время обработки заявок: 1.11
------------------------------------------------
Текущее время моделирования: 924.89
Обработано заявок 1-го типа: 300
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.45
Обработано заявок 2-го типа: 494
Текущая длина 2-ой очереди: 622
Средняя длина 2-ой очереди: 201.02
Время простоя: 62.29
Среднее время обработки заявок: 1.17
------------------------------------------------
Текущее время моделирования: 1226.91
Обработано заявок 1-го типа: 400
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.44
Обработано заявок 2-го типа: 684
Текущая длина 2-ой очереди: 818
Средняя длина 2-ой очереди: 247.90
Время простоя: 82.67
Среднее время обработки заявок: 1.13
------------------------------------------------
Текущее время моделирования: 1534.39
Обработано заявок 1-го типа: 500
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.42
Обработано заявок 2-го типа: 859
Текущая длина 2-ой очереди: 1021
Средняя длина 2-ой очереди: 280.52
Время простоя: 108.11
Среднее время обработки заявок: 1.13
------------------------------------------------
```

```
------------------------------------------------
Текущее время моделирования: 1820.01
Обработано заявок 1-го типа: 600
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.49
Обработано заявок 2-го типа: 981
Текущая длина 2-ой очереди: 1214
Средняя длина 2-ой очереди: 309.63
Время простоя: 123.13
Среднее время обработки заявок: 1.15
------------------------------------------------
Текущее время моделирования: 2133.60
Обработано заявок 1-го типа: 700
Текущая длина 1-ой очереди: 1
Средняя длина 1-ой очереди: 0.47
Обработано заявок 2-го типа: 1214
Текущая длина 2-ой очереди: 1407
Средняя длина 2-ой очереди: 366.20
Время простоя: 147.47
Среднее время обработки заявок: 1.12
------------------------------------------------
Текущее время моделирования: 2432.38
Обработано заявок 1-го типа: 800
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.47
Обработано заявок 2-го типа: 1391
Текущая длина 2-ой очереди: 1613
Средняя длина 2-ой очереди: 398.14
Время простоя: 163.16
Среднее время обработки заявок: 1.11
------------------------------------------------
Текущее время моделирования: 2727.49
Обработано заявок 1-го типа: 900
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.48
Обработано заявок 2-го типа: 1501
Текущая длина 2-ой очереди: 1798
Средняя длина 2-ой очереди: 412.05
Время простоя: 180.12
Среднее время обработки заявок: 1.14
------------------------------------------------
Текущее время моделирования: 3020.03
Обработано заявок 1-го типа: 1000
Текущая длина 1-ой очереди: 0
Средняя длина 1-ой очереди: 0.48
Обработано заявок 2-го типа: 1636
Текущая длина 2-ой очереди: 1992
Средняя длина 2-ой очереди: 453.27
Время простоя: 202.84
Среднее время обработки заявок: 1.15
------------------------------------------------
Итоговые результаты (список)
Общее время моделирования: 3024.97
Время простоя: 203.21
Количество обработанных заявок 1-го типа: 1000
Количество обработанных заявок 2-го типа: 1639
```

Теоретический расчёт
Ожидаемое время моделирования:
(5 - 1) * 1000 = 3000 е.в.
Ожидаемая длина 2-й очереди:
3000 / (3 / 2) = 2000
Погрешность составляет менее 1% от ожидаемого времени моделирования. 3026 ед. времени для массива и 3024 ед. времени для списка соответствуют ожидаемому результату.

Фрагментация

```
0x55beb702bef0
0x55beb702bf20
0x55beb702bf50
0x55beb702bf80
0x55beb702bfb0
0x55beb702bfe0
0x55beb702c010
0x55beb702c040
0x55beb702c070
0x55beb702c0a0
0x55beb702c0d0
0x55beb702c100
0x55beb702c130
0x55beb702c160
0x55beb702c190
0x55beb702c1c0
0x55beb702c1f0
0x55beb702c220
0x55beb702c250
0x55beb702c280
0x55beb702c2b0
0x55beb702c2e0
0x55beb702c310
0x55beb702c340
0x55beb702c370
0x55beb702c3a0
0x55beb702c3d0
0x55beb702c400
0x55beb702c430
0x55beb702c460
0x55beb702c490
0x55beb702c4c0
0x55beb702c4f0
```

Фрагментация не наблюдается.

Выводы по проделанной работе
Очередь, реализованную с помощью списка следует использовать, когда неизвестно количество элементов, в противном случает разумнее использовать очередь, реализованную с помощью массива.
Кроме того, список занимает только столько памяти, сколько нужно в данный момент (сам элемент списка требует дополнительную память для хранения адреса следующего узла).
Если нужно реализовать очередь, следует, ориентируясь на объём доступной

памяти и ожидаемое количество заявок, выбрать для реализации очереди либо массив, когда важна скорость и известно общее число заявок, либо список, когда важна память и неизвестно общее число заявок.

Контрольные вопросы

1. Что такое FIFO и LIFO?

FIFO — FIRST IN FIRST OUT - принцип работы как у очереди. На выход сначала поступают те элементы, которые пришли раньше всего.
LIFO — LAST IN FIRST OUT - принцип работы как у стека. На выход сначала поступают те элементы, которые пришли позже всего.

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

При реализации массивом объем памяти рассчитывается путем перемножения количества элементов на размер одного элемента, размер памяти не меняется.
При реализации списком, под каждый новый элемент выделяется память равна размеру узла. Память выделяется по мере необходимости, размер выделенной памяти меняется.

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

При удалении элемента из очереди в виде массива, перемещается указатель, память не освобождается. Память после работы с динамическим массивом необходимо освободить с помощью функции free().
Память для статического массива освобождается после того, как закончится область его видимости.
При удалении элемента из очереди в виде списка, память, выделенная под данный элемент освобождается сразу. Указатель на «голову» смещается.

4. Что происходит с элементами очереди при ее просмотре?

При просмотре очереди, головной элемент удаляется и указатель смещается.
При просмотре очереди её элементы удаляются.

5. От чего зависит эффективность физической реализации очереди?

При реализации обычным массивом, вставка и удаление элементов будет происходить дольше, что сильно уменьшит преимущества выполнения очереди в виде массива.
При реализации в виде списка легче удалять и добавлять элементы, переполнение памяти может возникнуть только если закончится память на куче, однако может возникнуть фрагментация памяти.
Если изначально знать размер очереди, то лучше воспользоваться массивом, иначе списком.

6. Каковы достоинства и недостатки различных реализаций очереди в

зависимости от выполняемых над ней операций?
При реализации очереди в виде массива может возникнуть переполнение очереди, тратится дополнительное время на сдвиги элементов, не возникает фрагментация памяти.
При реализации очереди в виде списка, проще выполнять операции добавления и удаления элементов, но может возникнуть фрагментация памяти.

7. Что такое фрагментация памяти, и в какой части ОП она возникает?
Фрагментация – размещение данных в оперативной памяти нелинейно.
Фрагментация возникает в куче.

8. Для чего нужен алгоритм «близнецов»?
Скорость выделения памяти можно существенно повысить, если выделять ее блоками заранее определенного размера, при этом может быть одновременно использовано несколько альтернативных размеров блока. Скорость выделения памяти возрастает потому, что не требуется искать в списке подходящий по размеру блок.
Метод близнецов обеспечивает очень высокую скорость динамического выделения и освобождения памяти, и используется в составе многих современных операционных систем для динамического распределения памяти в ядре системы, драйверах или в других ответственных компонентах системы, критичных к скорости работы.

9. Какие дисциплины выделения памяти вы знаете?
Две основные дисциплины сводятся к принципам "самый подходящий" и "первый подходящий". По дисциплине "самый подходящий" выделяется тот свободный участок, размер которого равен запрошенному или превышает его на минимальную величину. По дисциплине "первый подходящий" выделяется первый же найденный свободный участок, размер которого не меньше запрошенного. При применении любой дисциплины, если размер выбранного для выделения участка превышает запрос, выделяется запрошенный объем памяти, а остаток образует свободный блок меньшего размера.

10. На что необходимо обратить внимание при тестировании программы?
При реализации очереди в виде массива необходимо следить за отсутствием переполнения.
При реализации очереди в виде списка необходимо следить за освобождением памяти при удалении элемента из очереди.

11. Каким образом физически выделяется и освобождается память при динамических запросах?
Программа дает запрос ОС на выделение блока 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.
Обращение к этому адресу из программы в дальнейшем - неопределенное поведение.