|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки 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 (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5  
«Работа с очередью»**

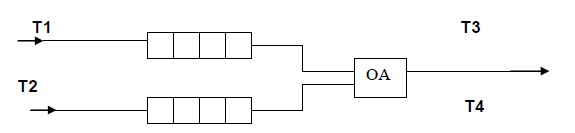
Студент, группа **Пронин А.С. ИУ7-32Б**

*2020 г.*

# Условия задачи

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.

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 **Т1** и **Т2**, 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 **Т3** и **Т4**, 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).

# Техническое задание

## Входные данные

1) Целое число от 0 до 5 (для ввода номера команды из меню)

2) Целое число >0 для ввода (для ввода кол-ва элементов при расчёте занимаемой памяти)

## Выходные данные

Вывод информации о моделировании процесса обслуживания

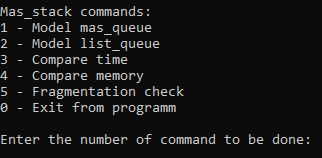
Вывод информации о сравнении времени работы

Вывод информации о сравнении занимаемой памяти

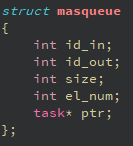
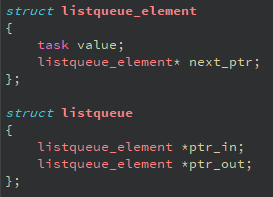
Вывод информации о проверке фрагментации

## Функции программы

Программа представляет из себя консольное приложение исполняющее следующие команды:



## Структуры данных

# Описание алгоритма

Команды:

1 – Моделирование процесса обслуживания с использованием очереди в виде массива

2 – Моделирование процесса обслуживания с использованием очереди в виде линейного списка

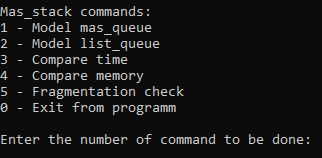
3 – Сравнивает время работы разных реализаций очередей

4 – Сравнивает размер заним. памяти разных реализаций очередей

5 – Показывает информацию для проверки на фрагментацию.

# Тесты

### Вот так выглядит меню:



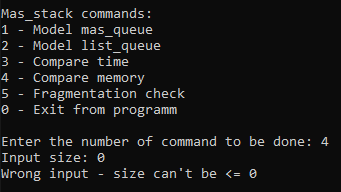
### Некорректный ввод:

1. Некорректная команда



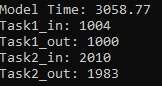


1. Команда 4 (Ввод не подходящего числа)

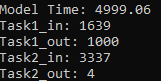


# Расчёт теоретического времени и сравнение с практическим

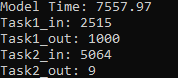
Имея T1 = (1, 5); T2 = (0, 3), T3 = (0, 4) и T4 = (0, 1), теоретическое время будет равно кол-во вышедших элементов (1000) умноженная на среднее время прихода заявок 1ого типа, т.к. T1ср > T3ср => 3000 е.в.

 В данном примере погрешность составляет 58.77/3000 \* 100% = 1.96%

Если же сделать время обработки (T3) больше времени прихода (T1) то общее время работы будет определятся средним значением T3, например при T3 = (0, 10) время должно быть 1000\*5 = 5000:

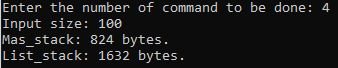
 Погрешность < 1%

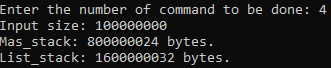
Или при T3 = (5, 10)



# Оценка эффективности

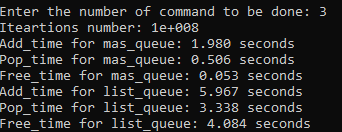
## Память





Поскольку для каждого элемента очереди в виде списка требуется указатель на следующий элемент, а самими элементами стека являются символы – логично что размер очереди в виде массива выигрывает по памяти примерно в 2 раза.

## Эффективность



Как видим по эффективности, очередь в виде массива также переигрывает очередь в виде списка, особенно видна разница при освобождении памяти, т.к. массив освобождается весь сразу, а по списку нужно пройтись, освобождая каждый элемент отдельно.

# Контрольные вопросы

**1. Что такое очередь?**

Очередь - структура данных, для которой выполняется правило FIFO, то есть первым зашёл - первым вышел. Вход находится с одной стороны очереди, выход - с другой.

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

При хранении кольцевым массивом: кол-во элементов \* размер одного элемента очереди. Память выделяется на стеке при компиляции, если массив статический. Либо память выделяется в куче, если массив динамический.

При хранении списком: кол-во элементов \* (размер одного элемента очереди + указатель на следующий элемент). Память выделяется в куче для каждого элемента отдельно.

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

При хранении кольцевым массивом память не освобождается, а просто меняется указатель на конец очереди. При хранении списком, память под удаляемый кусок освобождается.

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

Эти элементы удаляются из очереди.

**5. Каким образом эффективнее реализовывать очередь. От чего это зависит?**

Зная максимальный размер очереди, лучше всего использовать кольцевой статический массив. Не зная максимальный размер, стоит использовать связанный список, так как такую очередь можно будет переполнить только если закончится оперативная память.

**6. В каком случае лучше реализовать очередь посредством указателей, а в каком – массивом?**

Реализовывать очередь посредством указателей лучше только в случае, когда не известен максимальный размер, стоит использовать связанный список, так как такую очередь можно будет переполнить только если закончится оперативная память.

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

При использовании линейного списка тратится больше времени на обработку операций с очередью, а так же может возникнуть фрагментация памяти. При реализации статическим кольцевым массивом, очередь всегда ограничена по размеру, но операции выполняются быстрее, нежели на списке.

**8. Что такое фрагментация памяти?**

Фрагментация памяти - разбиение памяти на куски, которые лежат не рядом друг с другом. Можно сказать, что это чередование свободных и занятых кусков памяти.

**9. На что необходимо обратить внимание при тестировании программы?**

При тестировании программы необходимо обратить внимание на эффективное выделение и корректное освобождение динамической памяти. Помимо этого стоит обратить внимание на корректность реализации кольцевого массива, чтобы не произошло записи в невыделенную область памяти. Еще стоит обратить внимание на возникновение фрагментации памяти.

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

При запросе памяти, ОС находит подходящий блок памяти и записывает его в «таблицу» занятой памяти. При освобождении, ОС удаляет этот блок памяти из «таблицы» занятой пользователями памяти.

# Вывод

Использование связанных списков невыгодно при реализации очереди. Списки проигрывают как по памяти, так и по времени обработки. Но, когда заранее неизвестен максимальный размер очереди, то можно использовать связанные списки, так как в отличии от статического массива, списки ограничены в размерах только размером оперативной памяти. Также в ходе выполнения лабораторной работы выяснилось, что на моём компьютере есть только частичная фрагментация.